

Article

Discussion on IoT Security Recommendations against the State-of-the-Art Solutions

Marta Chmiel [†], Mateusz Korona [†], Fryderyk Koziół [†], Krzysztof Szczypiorski  and Mariusz Rawski ^{*}

Institute of Telecommunications, Faculty of Electronics and Information Technology, Warsaw University of Technology, 00-665 Warsaw, Poland; m.chmiel@tele.pw.edu.pl (M.C.); m.korona@tele.pw.edu.pl (M.K.); f.koziol@tele.pw.edu.pl (F.K.); k.szczypiorski@tele.pw.edu.pl (K.S.)

* Correspondence: m.rawski@tele.pw.edu.pl

† These authors contributed equally to this work.

Abstract: The Internet of Things (IoT) is an emerging concept comprising a wide ecosystem of interconnected devices and services. These technologies collect, exchange and process data in order to dynamically adapt to a specific context. IoT is tightly bound to cyber-physical systems and, in this respect, has relevant security implications. A need for IoT security guidelines was identified by the industry in the early 2010s. While numerous institutions across the globe have proposed recommendations with a goal to help developers, distributors and users to ensure a secure IoT infrastructure, a strict set of regulations for IoT security is yet to be established. In this paper, we aim to 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. We gathered recommendations proposed by selected government organizations, international associations and advisory groups, and compiled them into a set of the most common and important considerations, divided into eight categories. Then we chose a number of representative examples from IoT security technologies and evaluated them against these criteria. 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 which satisfies all the requirements of a secure IoT device. Further research on this matter could be beneficial. To the best of our knowledge, this is the first comprehensive survey to evaluate different security technologies for IoT device security against the compilation of criteria based on existing guidelines.

Keywords: cybersecurity; IoT; data protection; SoC



Citation: Chmiel, M.; Korona, M.; Koziół, F.; Szczypiorski, K.; Rawski, M. Discussion on IoT Security Recommendations against the State-of-the-Art Solutions. *Electronics* **2021**, *10*, 1814. <https://doi.org/10.3390/electronics10151814>

Academic Editor: Rashid Mehmood

Received: 6 July 2021

Accepted: 23 July 2021

Published: 28 July 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The present day is a time of unprecedented rapid technology development and the growth of the Internet. The majority of citizens in developed countries are not only smartphone users, but also surround themselves with intelligent devices such as various sensors, smart home appliances or CCTV cameras. These objects, which are capable of collecting, processing and exchanging data via various networks (also without human intervention), make up the Internet of Things (IoT). Researchers estimate that there were 12 billion IoT devices active in 2020 and this number will at least double within five years [1,2]. Unfortunately, people often focus only on the benefits of using IoT devices and tend to underestimate the risks.

Many IoT devices are deployed without sufficient security measures and can be easily exploited by more or less sophisticated attacks [3] with a significant impact on both individuals and society. Remote hijacking of a Jeep on the St. Louis highway is a well-publicized example of a personal IoT security breach [4]. White hat hackers in cooperation with a brave journalist acting as “the victim”, were not only able to manipulate car interiors (display, sound system, air conditioning), but also control the engine and brakes—elevating the severity of the incident from a prank to a potentially fatal attack. On the other end of

the danger spectrum are hostile actions that affect nationwide systems. Distributed Denial of Service (DDoS) attacks from Mirai malware-based botnets (consisting of thousands of compromised IoT devices) that targeted Internet service providers in France and the USA are a good example of this [5–7].

The original IoT paradigm is changing and system architectures are becoming increasingly edge-focused, moving processing of the data collected by sensors from the cloud into closer edge nodes (fog computing [8–10]) in order to reduce latency and required bandwidth. More and more, applications are expecting IoT nodes to be resilient to network connectivity issues, which means that IoT devices have to retain more intelligence and operation capabilities by themselves. The need for advanced data analysis is driving IoT device implementations in the direction of the entire System-on-Chip (SoC) [11], which consists of multiple interfaces, analog/digital circuits, memories and CPUs running highly functional operating systems, such as Linux. All in all, the attack surface of such highly sophisticated and functional IoT devices has increased greatly.

Many papers have been presented on the subject of IoT security. For instance, Mahmoud et al. [12] in 2015 discussed the security of a robust IoT network, with division into layers (perception, network and application). The authors listed a number of threats and attacks to which such a system is susceptible and, more importantly, raised concerns about existing major gaps in addressing basic security, for example, privacy and confidentiality. In 2019, Mohamad Noor and Hassan published a survey on IoT security research in the years 2016–2018 [13]. The conclusion was far from optimistic, since not much had improved over the years. Similarly, layers were insufficiently secured and not enough effort was put into ensuring comprehensive endpoint security. The rapid growth of IoT technologies was inevitably followed by an equally fast-paced growth of attacks. Insufficient focus on security allowed for the development of new, inventive ways of exploitation. Alladi et al. [14] in 2020 published a case study on vulnerabilities present in consumer devices. Their findings indicate that not only do the manufacturers often neglect proper protection of their devices but also that the users are unaware of the threats posed by, for example, a wireless scale that is in their bathroom. Consequently, many organizations noticed the IoT security problem and took steps to tackle it. Over the last decade, a number of them published documents discussing the importance of secure IoT and proposing guidelines. Some of these recommendations present a very detailed approach, from the design process to the user experience; others focus on just a part of the IoT device's life cycle.

Our work concentrates mainly on hardware and software design and some aspects of later stages of IoT device functioning, such as updating or event logging. The goal of this paper is to analyze how existing solutions for trusted computing, especially dedicated for IoT devices, adhere to these recommendations.

The remainder of this paper is organized as follows. Section 2 discusses our motivation and related work on the subject matter. In Section 3, security guidelines for IoT proposed by various organizations are gathered and compiled into a set of the most common and important. Section 4 contains the analysis of existing technologies addressing IoT security. We chose a number of representative examples and evaluated them against the criteria formulated in Section 3. Finally, Section 5 discusses the results and Section 6 concludes this paper.

2. Motivation and Related Work

IoT security can be analyzed from multiple angles and numerous publications on the subject are available. As a term, IoT is sometimes used to describe particular solutions, often from different ends of the technology spectrum requiring a specific approach to the subject, while sometimes it represents a general concept. The discussion ranges from security problems of specific technologies, such as RFID networks [15], through solutions of growing popularity, such as blockchain, machine learning or artificial intelligence [16],

to innovative propositions such as moving target defense [17], aiming at more elaborate structures, such as IoT networks.

National Institute of Standards and Technology (NIST) in 2020 published a document that focuses on defining an IoT device cybersecurity capability core baseline [18]. NIST describes the core baseline as a minimal set of capabilities that an IoT device should be equipped with so that it supports common cybersecurity controls. Advanced security schemes can be built on this basis. However, NIST does not provide advice on how it should be achieved. We decided that this was an interesting perspective and further research on the matter would be beneficial, especially for manufacturers and developers. As a first step, we conducted a literature search and checked whether other organizations provide guidelines regarding IoT device security. Secondly, we examined whether existing, state-of-the-art technologies can be utilized to fulfill the requirements.

We came across multiple survey articles regarding aspects of IoT, from wide-ranging analysis of an entire IoT system [19], to works focusing on protocols [20,21], IoT platforms [22] and frameworks, based on contemporary, commercial examples [23]. In each of these papers, security was considered but the emphasis was rather on existing issues and challenges of discussed solutions, instead of means of protection. To the best of our knowledge, no surveys focusing specifically on IoT *device* security capabilities were available.

Additionally, we researched surveys on IoT security. Our findings showed that the focus of the published work is again more on the challenges than on the solutions. For instance, Macedo et al. in 2019 [24] provided a systematic literature review focused on defining four main aspects of IoT security—authentication, access control, data protection and trust. Their work is addressed to manufacturers, developers, consumer and providers of IoT. Interestingly, the authors recognized a lack of reference architectures to develop secure IoT solutions. Nonetheless, the paper does not present existing guidelines and a link to state-of-the-art technologies. Abdul-Ghani and Konstantas [25] in their work provide an overview of several documents regarding the best practices for securing IoT, published by renowned organizations, such as the Broadband Internet Technical Advisory Group (BITAG) or the IoT Security Foundation (IoTSF). They also recognize the need for standardized security and privacy guidelines for IoT. In contrast to our approach, the analyzed guidelines are not confronted with existing commercial solutions.

Finally, we investigated the availability of articles presenting an overview of contemporary solutions, which could be applicable for securing IoT. In their paper published in 2018, Maene et al. [26] gathered over ten different technologies, dedicated to trusted computing. Similarly to our work, they are compared against a set of criteria. However, these criteria are based on capabilities offered by analyzed solutions, rather than existing guidelines. Even though the trusted computing solutions discussed in this article can, in some cases, be successfully used for IoT applications, there are other technologies catered specifically for this purpose that are in our opinion worth considering.

3. Security Recommendations for Internet of Things

With the number of connected IoT devices growing bigger each year, the question of security has become crucial. The threats and risks related to IoT devices, systems and services are manifold, and evolve rapidly. Hence, it is important to understand what needs to be protected and to develop specific security measures to protect the *things* from cyber threats. While a strict set of regulations on IoT security is yet to be established, a need for guidelines was first identified by the industry in the early 2010s and the discussion has continued since then.

3.1. Existing Guidelines

Numerous institutions across the globe have proposed their recommendations, in order to help developers, distributors and users ensure a secure IoT infrastructure. Government organizations, international associations and advisory groups are aware of the problem and have published many documents on the subject, to name *some* among many more:

- National Institute of Standards and Technology (NIST),
- European Union Agency for Network and Information Security (ENISA),
- GSM Association (GSMA),
- Internet Engineering Task Force (IETF),
- Internet Research Task Force (IRTF),
- IoT Security Foundation (IoTSF),
- ioXt Alliance,
- International Standard Organization (ISO),
- Institute of Electrical and Electronics Engineers (IEEE),
- International Telecommunication Union (ITU),
- Broadband Internet Technical Advisory Group (BITAG),
- Industrial Internet Consortium (IIC),
- Open Web Application Security Project (OWASP),
- Trusted Computing Group (TCG),
- Cloud Security Alliance (CSA),
- GlobalPlatform,
- Internet Society's Online Trust Alliance (OTA).

Our analysis focuses on just a number of them. A time cut-off of 2017 has been adopted for two reasons: rapid IoT industry development might outdate some concepts and, on the other hand, recent publications often reference older ones and align with them in *essential* matters. Furthermore, industry standards issued by renowned organizations or manufacturer associations have been considered over simple brochures or articles. Last but not least, it was important that the given document (or its independent section) primarily concentrates on secure IoT *device* implementation itself, as this is a foundation for deliberations in the next sections.

NIST, part of the U.S. Department of Commerce, in 2020 issued a report, "IoT Device Cybersecurity Capability Core Baseline" (NISTIR 8259A) [18]. The authors define an IoT device cybersecurity capability core baseline, which is a set of device capabilities generally needed to support common cybersecurity features that protect data, systems and ecosystems. The proposed baseline represents a coordinated effort to produce a definition of common capabilities, which is not an exhaustive list. This document highlights activities that aim to improve cybersecurity levels in manufactured products, which in consequence reduces the number of exploited IoT devices.

ENISA created a number of documents on secure IoT development. In 2017, "Baseline Security Recommendations for IoT" [27] was published. The aim of this work was to provide insight into the security requirements of IoT, with a focus on Critical Information Infrastructures. The paper offers a thorough analysis of existing cybersecurity threats, along with a comprehensive set of measures in order to protect IoT systems. The authors developed a series of recommendations based on the results of their research, the views expressed by the experts, and good practices, as well as security measures used in the industry. It is worth noting that this document provides an elaborate list of other security standards regarding IoT, which can be a valuable starting point for further research.

Documents published by the GSMA provide very useful insight and pose questions that IoT designers and network administrators will find useful while discussing system security. The "IoT Security Guidelines" document set [28–30] should especially be considered at the early stages of development, as it asks a series of important questions regarding security which are very helpful during the process. These documents promote a methodology for developing secure IoT services to ensure security best practices are implemented throughout the life cycle of the service. The authors provide recommendations on how to mitigate common security threats and weaknesses within IoT services. The set of documents analyses two ecosystems—service and *endpoint*—but also provides a number of real-life examples.

IETF and IRTF are cooperating, parallel open standards organizations, that focus on short-term and long-term Internet-related research, respectively. They have issued a couple

of highly informative drafts regarding IoT security. In 2017, “Best Current Practices for Securing Internet of Things” [31] was published by IETF. This report collects guidelines for IoT designers and developers, written by engineers from Network Heretics, Mozilla and Arm. It offers valuable remarks on low-level IoT development, by discussing the authentication, encryption, and design of a device and firmware. Even though it is now labelled as expired, we find this document provides valuable input into the discussion. In March 2021, another draft was released—“Security Technical Specification for Smart Devices of IoT” [32]—collecting detailed recommendations from hardware to software level and proposing a secure IoT device model. In April 2019, IRTF published “RFC8576—Internet of Things (IoT) Security: State of the Art and Challenges” [33]. In this document, the authors present a list of already existing guidelines regarding IoT security; they report and predict the development of IoT and point out possible challenges, especially with reference to the nature of resource-constrained IoT devices (e.g., in terms of algorithms and protocols that would allow IoT devices to safely operate in a heterogeneous network with powerful, potentially malicious Internet resources). The aforementioned publications are complementary to each other and were issued by cooperating organizations, therefore conclusions drawn from them are presented together.

The IoTSF and ioXt Alliance are composed of industry leaders, manufacturers and government organizations, dedicated to creating a security and privacy standard for IoT—some of these entities belong to both of these organizations. The first consortium published “Secure Design—Best Practice Guides” [34] at the end of 2019. This document highlights the importance of maintaining a chain of trust throughout the hardware and software layers of IoT device. The ioXt Alliance has recently published “ioXt Pledge: The Global Standard for IoT Security” [35], in which eight core principles are defined and described. It considers a wide range of subjects, such as secured interfaces, proven cryptography, software verification/updates and vulnerability reporting mechanisms. The organization offers a certification program and creates a network of authorized laboratories. It also encourages independent researchers to participate in the certification process, by validating that every security requirement is fulfilled.

It is worth highlighting that ISO is currently working on their own guidelines. As of June 2021, ISO/IEC CD 27400 “Cybersecurity—IoT security and privacy—Guidelines” is still under development [36].

3.2. Evaluation Criteria

In this section, we created a set of recommendations with a focus on SoC hardware and software security, deriving from the documents mentioned in Section 3.1. This selection is later used to analyze the state-of-the-art IoT security technologies. The aim of Table 1, presented later in this section, is to collect the most common and important recommendations from the analyzed literature and to provide a solid overview of what is expected from a well-secured IoT device. As already described in Section 2, our starting point was a security core baseline for IoT devices defined by NIST and, as mentioned in Section 3.1, the main focus during criteria analysis was put on secure IoT device implementation. Therefore, high level concepts, such as, for instance, network structure or its security remained out of scope. On the other hand, topics such as the safety of an industrial or automotive IoT node and its capability to operate in various environmental conditions (e.g., temperature, humidity, contact with harsh chemicals) do concern device architecture, but are mostly related to its reliability instead of security. Only the aspects of physical access to the device relevant at the chip level were considered, because those regarding the product level can be very location or application specific.

The analyzed papers have multiple points in common. In Table 1, we collected the most prevailing suggestions and divided them into the following groups on the basis of key functionalities:

- hardware security,
- trust and integrity management,

- data protection and software design,
- device configuration and software update,
- secure interfaces and communication,
- cybersecurity event monitoring and logging,
- cryptography and key management,
- device identification, authentication and strong default security.

The intent was to mimic the process of constructing a secure IoT device by creating a checklist of requirements it has to fulfill. Almost all of the analyzed documents proposed the functional classification of secure IoT device characteristics with some exceptions. A relevant example is GSMA's document [30], where the requirements were distributed by implementation priority (Critical, High, Medium, Low). Categories presented in this paper are similar to the ones recommended by ENISA [27], but the number of groups was reduced, and an appropriate level of granularity was maintained, which allowed for concise comparison of available secure IoT implementations.

Hardware Security

This category collects recommendations for designing IoT devices based on the Root of Trust (RoT) concept and the characteristics such a component should demonstrate.

A Root of Trust is a unit that consists of a computing engine, low level code and data (e.g., cryptographic keys). It provides security services/features necessary to establish trust and security within the platform it is a part of. Its vital characteristics are *immutability* and *predictability*—the produced results have to be consistent for the same input data. The hardware implementation of an RoT enables the fulfillment of these conditions [37,38].

A Root of Trust can provide the following independent security services—identification, authentication, confidentiality, integrity and measurement (state of the platform), as well as composite services (relying on the independent ones)—authorization, verification, reporting, secure storage and update.

Unsurprisingly, almost every analyzed document provided some guidance on the hardware role in the security of the final product, ENISA [27] and the GSMA [30] being the most elaborate. It proves that protecting an IoT device must start at the hardware level.

Trust and Integrity Management

This section focuses on requirements regarding trust establishment and ensuring integrity, which are fundamental to IoT device security.

An inherently trusted, immutable (hardware) Root of Trust is the trust anchor, from which trust is extended to the whole platform through a secure boot process. A Hardware Root of Trust (HWRoT) utilizes its security services to verify the integrity of subsequently executed software modules (a cryptographic signature of code is checked). Verified software modules then become the next *Chain of Trust* elements. If the integrity verification check fails during one of the secure boot stages, the whole process has to be aborted and the system can only be trusted up to the given *Chain of Trust* level. Depending on which advanced capabilities of the system are available at this point, the device might have to reboot, attempt to return into last known secure state or remain as it is, that is, for reporting purposes.

Apart from NIST and IETF/IRTF publications, every considered paper included recommendations on this matter.

Data Protection and Software Design

This category considers recommendations on data handling and fundamental principles of software architecture. Data confidentiality must be protected through encryption. Additionally, the designer of the system must also ensure that applications processing data operate on minimum privilege and are isolated from each other (e.g., through memory compartmentalization). These subjects were discussed in the majority of analyzed documents.

Device Configuration and Software Update

The question of software updates is highly stressed in the subject matter. All organizations agree on its importance—a lack of identified vulnerability patching and protection against the latest threats is a large security risk in IoT. Thus, keeping the device up-to-date strongly improves its protection. The capability to *securely* update device software by an authorized entity is a must, preferably this process should be automatic and/or remotely available.

Secure Interfaces and Communication

This category analyzes guidance on secure communication, starting from interfaces, through protocols, to data. The usage of device interfaces should be configurable and, by default, only those required should be active. Systems should utilize state-of-the-art, standardized security protocols, with emphasis on using the versions that are intended for IoT, if available. The majority of publications stress the necessity for making only intentional and required connections (preceded by mutual authentication of the peers), which are used to transmit confidential and integral data. Recommendations apply to all layers of IoT devices—hardware, firmware and software.

Cybersecurity Event Monitoring and Logging

Event logging is a key requirement for security management in an IoT device and this category summarizes guidelines on this matter. Adequate level of details aids to solve issues with security incidents, while preventing exposure of sensitive information. The confidentiality and integrity of logs must be protected so that they are reliable—only authorized entities should be able to examine them and they should be unmodifiable.

Furthermore, the GSMA suggests creating a reference model of the IoT device behavior and perform anomaly detection—situations when the device erratically reboots, reconnects to the network or sends multiple poorly-formed messages might indicate a security issue.

Only the ioXt Alliance did not offer guidance on this subject.

Cryptography and Key Management

Each of the evaluated documents provided an insight for this category. It is clearly advised to use standardized, *proven* cryptographic algorithms with secure implementations and sufficient key lengths. Documents highlight the need to support algorithm agility in security protocols—to enable the usage of lightweight cryptography and specialized algorithms for IoT applications, to allow various node types for algorithm and key length negotiation so they can communicate or ensure the ability to change the algorithm or used key length in case it is compromised.

A secure and scalable key management policy must be introduced in the IoT system and every IoT device should be provisioned with a *unique* private key (possibly per application, e.g., device identification, code signature, server communication, etc.). With this approach, every device becomes a separate attack surface and the whole system is secure even if one of them is compromised.

Device Identification, Authentication and Strong Default Security

This section discusses the identity of both the device and its user. The device must be labelled and recognized in a credible way, given that it will operate in a wider network. It should also be protected from undesired access and provide security for user data. As for the software developer and the user, this category includes guidance on password management and general authentication procedures.

Table 1. Synthesized requirements for secure IoT devices.

No.	Requirement	NIST	ENISA	IETF IRTF	GSMA	IoTSF	ioXt
Cat. 1	Hardware Security						
1.1	Utilization of immutable Hardware Root of Trust (HWRoT—Trust Anchor)—trusted component that extends the chain of trust to other HW, FW, SW components	-	X	X [32]	X	X	-
1.2	HW provided security features (memory locking, storage for cryptographic keys, secure boot support, device authentication, communication confidentiality and integrity, ...)	X	X	X[32]	X	X	-
1.3	Measures for tamper protection and detection	-	X	-	X	X	-
1.4	Use of proven/cryptographic quality Random Number Generator (hardware-based if feasible)	-	X	X[31,32]	X	-	-
Cat. 2	Trust and integrity management						
2.1	Secure boot process based on HWRoT. Boot process initializes the main hardware components, verifies executed code (first-, second-stage bootloader, OS) and results in environment determined to be in an uncompromised state	-	X	-	X	X	X
2.2	Software (code, applications) must be signed cryptographically and verified upon installation or execution	-	X	-	X	X	X
2.3	The ability to restore last known secure state (e.g., firmware rollback, when code is verified as damaged or tampered)	-	X	-	X	X	-
2.4	Software installation control in operating systems (OS), to prevent unauthenticated software and files from being loaded onto it	-	X	-	-	-	-

Table 1. Cont.

No.	Requirement	NIST	ENISA	IETF IRTF	GSMA	IoTTSF	ioXt
Cat. 3	Data protection and software design						
3.1	Encryption of data storage medium. Possibility of locking or erasing device contents remotely	X	X	X _[32]	-	X	-
3.2	Ensuring that data is secure prior to use by the process (input data validation)	-	X	X _[32]	-	X	-
3.3	Process privilege minimization - limited permission of allowed actions, applications must operate at the lowest privilege level possible	-	X	X _[31]	X	X	-
3.4	Utilization of memory compartmentalization to prevent rogue or compromised applications from accessing memory areas that they are not authorized to use, process isolation	-	X	X _[31]	X	X	-
Cat. 4	Device configuration and software update						
4.1	The ability to change the device's FW and SW configuration by authorized entities	X	-	X _[32,33]	X	X	-
4.2	The ability to update device's FW and SW through remote or local means by authorized entities	X	X	X _[31,32]	X	X	X
4.3	Update file must be encrypted, signed and device verifies its integrity before application	X	X	X _[31-33]	X	X	-
4.4	Automatic update capability (checking at frequent, but irregular intervals), enabled by default	X	X	X _[31,33]	X	-	X
4.5	Backward compatibility of updates—update should not change network protocol interfaces nor modify user-configured preferences, security and/or privacy settings	-	X	X _[31]	-	-	-

Table 1. Cont.

No.	Requirement	NIST	ENISA	IETF IRTF	GSMA	IoTTSF	ioXt
Cat. 5	Secure interfaces and communication						
Interface security							
5.1	Ability to disable or logically restrict access (e.g., device/user authentication) to any local and network interfaces	X	X	-	-	X	X
5.2	Deployed device should never contain debugging, diagnostic or testing interfaces that could be abused by adversary (JTAG, CLI, Telnet, etc.)	-	X	X _[32]	X	X	X
Protocol security							
5.3	Ensure that communication security is provided using state-of-the-art, standardized security protocols (e.g., TLS for encryption)	-	X	X _[32]	X	X	-
5.4	Mutual authentication of the devices must be performed before trust can be established (prevent man in the middle attack), even when given device leaves and re-joins network	-	X	X _[31]	X	X	X
5.5	Make intentional connections. Prevent unauthorized connections to product or other devices it is connected to, at all levels of the protocols. IoT devices must provide notice and/or request a user confirmation when initially pairing, onboarding, and/or connecting with other devices, platforms or services.	-	X	X _[32]	-	X	X
Data security							
5.6	Guarantee different security aspects of the transmitted data—confidentiality, integrity, authenticity	-	X	X _[31,32]	X	-	-

Table 1. Cont.

No.	Requirement	NIST	ENISA	IETF IRTF	GSMA	IoTSF	ioXt
Cat. 6	Cybersecurity event monitoring and logging						
6.1	The ability to log cybersecurity events from device's HW, FW and SW	X	X	X _[32]	X	X	-
6.2	The ability to record sufficient details for each logged event to facilitate an authorized entity examining the log and determining issue origin	X	-	X _[32]	X	X	-
6.3	The ability to restrict access to the logs so only authorized entities can view them and prevent all entities (authorized or unauthorized) from editing them	X	X	X _[32]	-	X	-
6.4	Avoid security issues when designing error messages. An error message should give/display only the concise information the user needs—it must not expose sensitive information that can be exploited by an attacker, such as an error ID, the version of the web server, etc.	-	X	X _[32]	-	X	-
6.5	Device's behavior should be monitored and compared with model to detect anomalies (e.g., erratic reboots/resets, (dis)connections, different network fingerprint, repeated malformed messages sent, etc.)	-	-	-	X	-	-
Cat. 7	Cryptography and key management						
7.1	Use of Standard Cryptographic Algorithms and Security Protocols, which implementations have been independently reviewed	X	X	X _[31–33]	-	X	X
7.2	Security protocols should support algorithm agility (lightweight cryptography for resource constrained devices, ability to change algorithm or use it only with longer key in case it is compromised)	X	X	X _[31,33]	-	X	-

Table 1. Cont.

No.	Requirement	NIST	ENISA	IETF IRTF	GSMA	IoTSF	ioXt
7.3	Length of the key should provide strong security, make brute-force attack infeasible	-	X	X _[31]	-	X	-
7.4	Every device must be instantiated with unique private key(s)	-	X	X _[31]	X	X	-
7.5	Secure and scalable management of cryptographic keys (generation, storage, distribution)	-	X	X _[31]	X	X	-
Cat. 8	Device identification, authentication, strong default security						
8.1	Device must have a unique physical identifier only authorized entities can access	X	X	-	X	X	-
8.2	Device must be provisioned with unique logical identifier	X	X	X _[32]	X	-	-
8.3	Prepare robust authentication and authorization schemes, so device can prove its identity	X	X	-	X	X	-
8.4	Establish strong, device-individual default passwords that has to be changed by the user during initial setup (weak, null or blank passwords are not allowed)	-	X	X _[31,32]	X	X	X
8.5	Resistance to keyspace-searching, brute-force or other login abusive attacks by limiting number of invalid login attempts or introducing incrementing delays between them	-	X	X _[31,32]	X	-	-
8.6	Availability of two-factor authentication	-	X	X _[31]	X	X	-
8.7	Product security shall be appropriately enabled by default. Strong security controls should be something the consumer has to deliberately disable rather than deliberately enable.	-	X	X _[31]	-	X	X

4. Review of Existing Solutions

In this section, we analyze some of the current popular trusted computing solutions used for IoT devices against the recommendations presented in Section 3. The solutions chosen are either already mature and currently in use for IoT devices or are emerging concepts that might bring new quality to the topic. The selection of technologies for IoT security includes examples that are more hardware-based (e.g., GEON SoC Platform) and more software-focused, such as Intel Software Guard Extensions (SGX). We chose the most popular solutions available on the market, such as Arm's TrustZone or Intel SGX, the most promising open source ones, such as Keystone and OpenTitan and other representative, usually hardware-based, solutions. In the last group, we targeted solutions that can be integrated into a designed SoC in the form of a standalone IP—Rambus RT, similar but more complex solutions, such as Geon SoC Security Platform, and finally, a solution offered as an integrated circuit (NXP EdgeLock SE050). A short description is provided for each one before the analysis of the IoT security in the context of Table 1. We present our findings for each solution in Table 2. The descriptions are formed from the perspective of a designer who wants to understand the requirements for a secure IoT device. We assumed that the analysis is based only on publicly accessible information, no lab testing nor feasible attacks on devices based on the solutions were performed, and that little to no implementation experience for each of the discussed solutions is required.

4.1. Arm TrustZone

TrustZone is a security extension offered by Arm for application processors (Cortex-A family) and microcontrollers (Cortex-M family), which has become popular recently. There is a growing interest in the subject across academia, and a number of commercial products utilize TrustZone's capabilities, for example, Samsung Knox. Arm's solution is based on a Trusted Execution Environment (TEE) concept and enables the system to fulfill Platform Security Requirements [39]—a set of guidelines defined by Arm. This document provides similar guidelines to those gathered in Section 3, with a strong focus on hardware and firmware security. There are some differences between architecture-specific implementations of TrustZone, but the main idea remains the same—on Cortex-A processors, the secure monitor, a privileged software, implements mechanisms for secure context switching between worlds; on Cortex-M processors, there is no secure monitor and hence the change between the secure and non-secure world is handled by a set of core logic mechanisms. Enabling TrustZone for microcontrollers has allowed for more widespread usage in resource-constrained devices, which has an impact on IoT devices. Pinto and Santos [40] provided a detailed explanation of TrustZone's operation with a distinction between Cortex-A and Cortex-M implementation. The general concept of TrustZone's operation on software and firmware levels is presented in Figure 1.

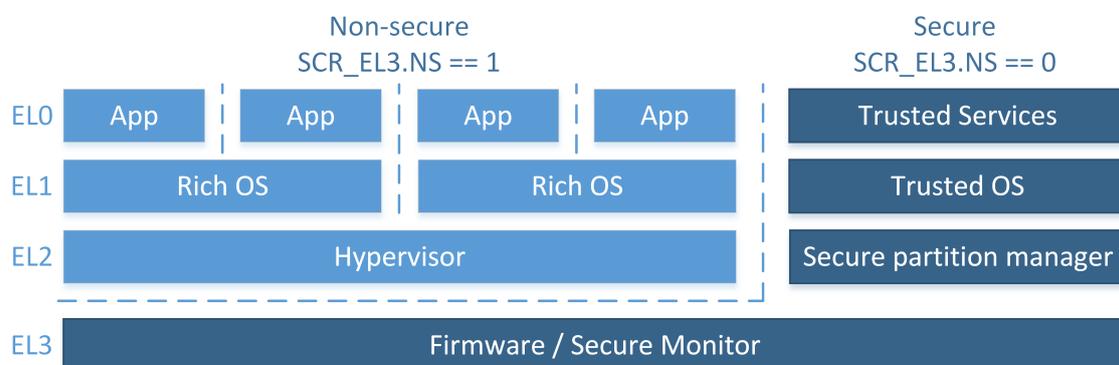


Figure 1. Concept of a TrustZone supported execution environment on software and firmware levels, based on [41]. The division between non-secure and secure worlds, as well as their construction with accentuated levels of operation, are presented. The SCR_EL3.NS is a register bit used to switch between the trusted and not trusted environments. The Secure Monitor, a special processor mode controlling the transition between secure and non-secure states, is presented as a common base for the environments.

TrustZone provides two virtual processors supported by hardware-based access control, resulting in division into secure and non-secure environments. Both execution environments are completely separated in the hardware, thanks to memory isolation and a special processor mode dedicated to monitoring (*secure monitor*). A few additional modules, such as the TrustZone Address Space Controller (TZASC), the TrustZone Memory Adapter (TZMA) and internal interface modifications, are introduced for separation and memory partitioning in TrustZone for Cortex-A. Noticeably, Arm states that the use of TZASC and TZMA is optional in the system. The final decision is left to the designer. Peripherals are flagged as *secure* or *normal*, while the APB-AXI bridge hardware is responsible for access control and rejecting AXI transactions with insufficient permissions. There is no separated trusted path since secure and non-secure transactions are multiplexed on the system bus using the same hardware. Significantly, in TrustZone there are no explicit considerations for Hardware Root of Trust implementation. However, the implementation of Root of Trust has been proposed in the literature by Zhao et al. [42], and a commercial solution, discussed later in this subsection, has also been made available. Some security features are provided by separate hardware modules. For example, remote attestation is achieved by an incorporated hardware component, such as a Trusted Platform Module, responsible for measuring the kernel's integrity and creating unique cryptographic keys. While there is no information on random number generation in TrustZone documents, Arm offers a separate security IP called the True Random Number Generator, advertised as TrustZone compatible. Similarly, secure storage can be implemented by simply denying access to a device from a non-secure world.

From a firmware and software perspective, a privileged instruction called the Secure Monitor Call (SMC) allows for entry to, exit from and general communication between secure and non-secure world applications. This mechanism involves a monitor software with higher privileges than the Rich Execution Environment Operating System (REE OS). Its responsibility is the reliable and protected context switching of the processor [40]. A Non-Secure (NS) bit stored in the Secure Configuration Register (SCR) represents the current context of the processor and the register's state is propagated throughout the entire SoC. Hence, it is possible to use peripheral devices that only allow secure access from the processor. In TrustZone there is no attestation of processes requesting execution in the secure world. In the event of an OS falling into the hands of an adversary, messages that require secure world resources can be crafted and possibly other information may be gleaned from apparently secured processes. A newer TrustZone technology for Cortex-M microcontrollers has some changes to allow for usage in more resource-constrained devices. The division between secure and non-secure worlds is based on memory map partition, and context switching happens due to code exceptions. To support this, a couple of new, dedicated instructions were added.

One could assume that TrustZone alone leaves quite some room for interpretation in terms of security functions implemented in hardware. That being said, Arm proposed the CryptoCell IP-family [43,44], which acts as HWRoT, offering cryptographic acceleration, true random number generator, trusted storage, secure boot support and authenticated debug support, to name a few. CryptoCell-300 is dedicated for Cortex-M implementations, while CryptoCell-700 is Cortex-A oriented [45]. Both families provide broad support for symmetric and asymmetric cryptography, as well as lightweight cryptography [46]. Code integrity and signing, authentication and key management are also mentioned in the documents. Arm claims that this solution is meant for low power, low area designs [43,44]. It may seem that TrustZone, complemented by CryptoCell IP, is a powerful security solution, mostly implemented in hardware.

Analysis for Synthesized Requirements for Secure IoT Devices

Some security flaws inherent to the REE-TEE communication channel have been found in previous years (Black Hat 2014, Black Hat 2015) [40] and have been used in attacks on real devices. Additionally, concerns with cache side-channel attacks arise. The lack of

inherent memory encryption also raises questions. TrustZone in itself does not have any recommendations nor requirements for HWRoT implementation other than the need for the existence of a unique device key [40]. This is paramount for the security of an IoT device and cannot be left to wide interpretation. One could assume that Arm's recommendation is to use IP from the CryptoCell-family, though it is not always feasible. On the software execution front, the secure-world approach is not the same as an enclave and allows for a compromised TEE program to interfere with other programs within the TEE. Secure IoT required functionality, as presented in Section 3, is possible in TrustZone, but with much of the implementation left to the designers—secure deployment might be at risk. TrustZone documentation does not provide input on configuration, update or logging mechanisms. We assume it is at the designer's discretion. Finally, TrustZone is entwined with Arm's technology; it is not a universal solution and porting it to different platforms does not seem feasible. On one hand, this can be limiting for some implementations, as not all IoT devices use Arm's solutions. On the other hand, it would be surprising if one of the leaders in the processor market did not have a compatible security solution available.

We can conclude that TrustZone certainly offers an end-to-end security solution, but it requires a good understanding of the framework, some creativity in implementation and support from external IPs. It is worth repeating that this solution is dedicated for Arm infrastructure. The vast number of TrustZone supporting products speaks for itself. TrustZone fulfills most of the security recommendations, but it would not be possible without supplementary hardware, like CryptoCell or applications. TrustZone alone is not an off-the-shelf, ready-to-use solution.

4.2. Intel Software Guard Extensions (SGX) and Security Essentials

Intel SGX [47] is a set of CPU instructions that allow the creation of isolated software containers called enclaves in which code, data and stack of a program are isolated safely from other processes (even with higher privilege levels) through hardware-based access policy control and memory encryption. Unsurprisingly, this solution is meant for Intel architecture. Application code and the hardware it is running on can be attested by a remote entity [48] by verifying measurements of its code and data (named MRENCLAVE), calculated during enclave creation. This process provides increased confidence that the code is running in an enclave and is unmodified by a third party. It provides a mechanism to run a secure code in an unsecured OS, with support from trusted hardware. After successful attestation, an encrypted channel (based on public key schemes) between the remote party and the enclave can be established in order to exchange sensitive data. Data can be sealed using CPU instructions to generate a key (named MRSIGNER) which allows decryption only by the same enclave created in the future.

Additionally, in the Intel security ecosystem, a provisioning functionality is present, which allows trusted entities to update or add software whilst assuring integrity. It is presented in Figure 2. The provisioning process is described below:

- **Creating the enclave**—an untrusted REE application creates an enclave environment in order to protect the software of the trusted provider, during this process the contents and creation parameters are logged as a measurement,
- **Attestation**—the enclave informs the software provider about its readiness to receive new software and the device presents the measurement from before,
- **Provisioning**—a secure channel between the device and provider is created and the data is sent,
- **Sealing and unsealing**—the enclave uses a cryptographic hardware key for encryption before storing the data in memory. Only an identical enclave will be able to decrypt and use the new program data in the future,
- **Software update**—in this step, a new version of a program may ask an older version to unseal the data. After the update process ends, a new seal is created, which renders the old software version (which could be compromised due to flaws patched in the new version) unable to access the new software version data.

From a hardware viewpoint, Processor Reserved Memory (PRM) is isolated by SGX and protected against all memory accesses from outside an enclave, including kernel, hypervisor and system management mode, as well as DMA accesses requested by peripherals. Enclave's code and data are stored in Enclave Page Cache (EPC), providing pages of a 4KB size [49]. The untrusted OS is in charge of assigning EPC pages to enclaves, which creates a potential exposure. The processor is responsible for assuring that each enclave has only one EPC on hand; it is monitored in Enclave Page Cache Metadata (EPCM).

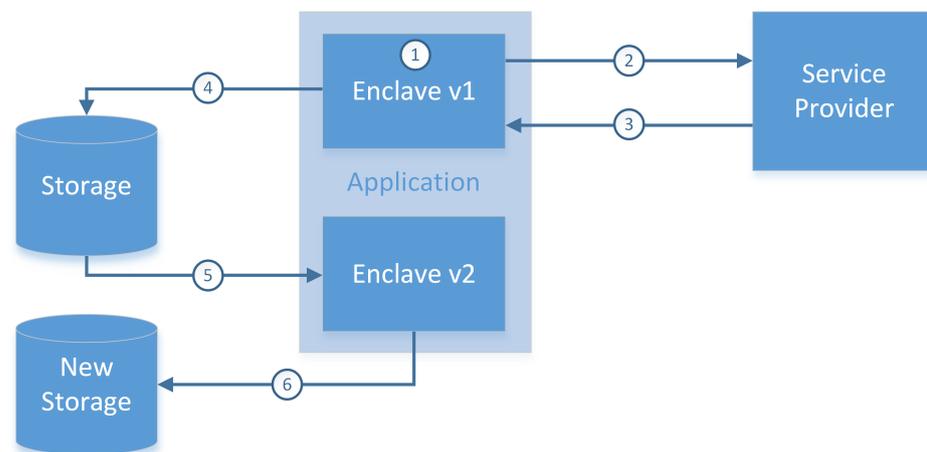


Figure 2. Intel SGX Software Lifecycle; steps are executed in the numerical order, sensitive data are remotely provisioned (3) into the enclave after mutual attestation (2) of the off-platform provider and created enclave (1), after the enclave is destroyed data are sealed in memory (4) in such a way that only an identical enclave can access them again. Software updates (5) are also possible via this scheme and a new seal is created (6) so that an old version of software cannot overwrite the new version. Based on [47].

SGX in itself is generally a TEE implementation, but other Intel modules and technologies allow for wider SoC security such as secure boot capabilities, TPM integration or random number generators [50]. It is advertised that the security extension assists with securing IoT edge device communication [51].

Analysis for Synthesized Requirements for Secure IoT Devices

Intel SGX is a proprietary enterprise solution; therefore all security functionalities are tied in with Intel architecture. This means that other custom application accelerators, interfaces and system bus control are out of the scope for the solution. As such, all SoC security issues have to be solved by custom design decisions, increasing the risk of creating an unsecured device. Costan and Devadas [49] in 2016, one year after the Intel SGX debut, published an in-depth examination of this solution. They exposed a number of potential vulnerabilities to SGX and claimed that “our security analysis reveals that the limitations in SGX’s guarantees mean that a security conscious software developer cannot in good conscience rely on SGX for secure remote computation” ([49], [p. 3]). In the five years that have passed since this article was published, a number of new side channel attacks have been discovered [52–55], and although Intel describes these attacks as very difficult to perform in a data center (i.e., physical access to the platform is required), this changes in an IoT context (device present in a public space) and remains alarming. Due to a security by obscurity situation, it is hard to assess the solutions for many requirements in Table 2 and thus no assurance on their completeness can be given, though an optimistic approach is taken. Interestingly, it is possible to change all device cryptographic keys by changing a single register called OwnerEpoch. This process allows for the fast sealing of the device and serves as additional protection of the enclave content. If the OwnerEpoch value is lost, the device is rendered inaccessible.

SGX and the Security Essentials provide multiple good trusted computing basics but is not a solution entirely catered for protection of IoT devices. This can be seen in the Cat. 6 and Cat. 8 sections of Table 2. However, the security level it provides and the critique around it, raises questions about whether it is a good choice at all.

4.3. Keystone

Keystone [56] is an open-source framework designed for creating TEE environments based on unmodified RISC-V architecture. RISC-V Physical Memory Protection (PMP), arbitrarily securing the physical memory locations, and the programmable machine mode (M-Mode) are used to implement the memory protection scheme. The trusted Security Monitor (SM) program is proposed on M-Mode level and its main task is to manage the secure handling of hardware and context switching between enclaves (Figure 3). It should be executed entirely from on-chip memory. This component satisfies typical TEE requirements such as memory isolation and code/configuration attestation. Keystone does not propose direct resource management. This responsibility lays on the secure enclave application developer side. A runtime (RT) component is an enclave-specific platform for secure applications to be executed on, which also communicates with the SM and manages virtual memory chunks assigned to the enclave. The RT should have the functionality of system plugins, interfaces, *libc* implementations, paging and virtual memory management. It can be successfully reused between enclaves or modified as needed. Additionally, because the rich OS is not a part of a typical enclave, the Trusted Computing Base (TCB) is smaller.

From a hardware perspective, implementing Keystone does not require modifications to CPU cores or memory controllers. However, several requirements for the platform are listed in the publication [56]: trusted boot process support, unique authentication key dedicated for this process and a hardware source of randomness. According to the Keystone designers, the Root of Trust can be realized in hardware, but does not have to, which allows for a variety of implementations, for example tamper resistant software (zeroth-order bootloader) [56].

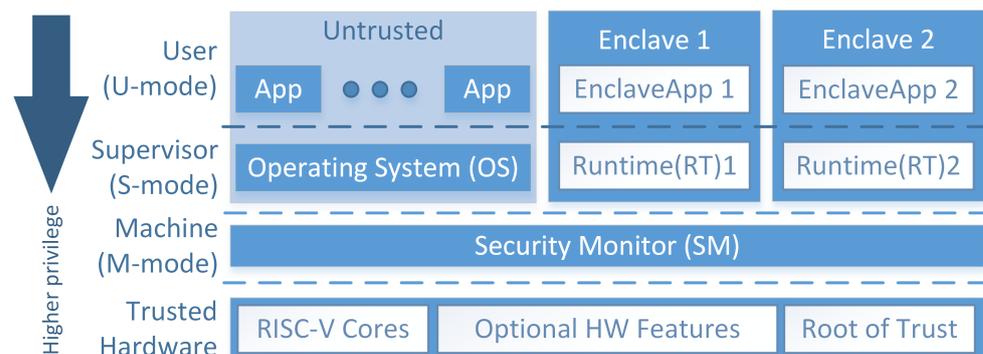


Figure 3. Overview of a Keystone system setup, based on [56]. The distinction between untrusted and trusted execution environments is presented, along with support for multiple secure enclaves and their runtimes. The privilege level for each RISC-V mode is marked. The security Monitor, operating on M-mode, being common for both execution environments, is the manager of context switching. Required trusted hardware is included.

With every CPU reset the Root of Trust executes these steps:

- Measures the SM image loaded,
- Generates a new attestation key based on the randomness source,
- Saves the data in a SM memory location isolated by PMP,
- Sends the cryptographically generated metadata via a public key scheme.

In the RISC-V architecture, only the machine mode has access to hardware resources such as interrupts, system memory, peripherals and devices. The S-mode (supervisor)

is used for the OS kernel and the U-mode (user mode) allows execution of typical REE applications. The machine mode is used as a platform for the SM, because:

- It is programmable,
- It allows control over interrupts and exceptions above the OS level,
- PMP mechanism, which allows the enforcement of access policies.

PMP restricts the physical access to memory locations for the S and U modes. Every record in the PMP table has a set of access flags for the programmed memory segment. Each PMP address register encodes a continuous address region and the configuration bits represent write/read/execute (rwx) permissions for the U/S modes. If a mode attempts to access an unassigned memory area, the access is rejected by the hardware. The PMP areas can be dynamically allocated during device operation. During SM image loading, the first PMP area is configured as the highest priority in order to protect its own resources, such as code/stack/data. By default, the OS has access to any memory location that is not part of an assigned PMP area of higher priority. When an application starts an enclave, the OS finds a continuous area of memory, which does not overlap with any other PMP configured area, and then switches the request to the SM that creates a new PMP record. During a context switch from an enclave to the OS or another enclave, the SM takes away all access permissions, but the enclave's address region is preserved. As a result, an enclave is securely isolated from other processes. Upon creation, the enclave's memory is measured and cryptographically signed. The OS uses an interface within the Linux kernel (/dev/Keystone) for enclave creation.

Analysis for Synthesized Requirements for Secure IoT Devices

The paper that introduces Keystone focuses more on the enclave code execution, while treating key management, software updates, device authentication and other problems such as orthogonal. A Root of Trust (either hardware or software) is stated as a necessity along with a trusted boot process, but it is in the hands of the application designer to make sure that secure IoT requirements are in place. Process and application privilege levels are inherent to the RISC-V implementation. Enclave security is supposed to include resistance to side-channel attacks, mapping attacks and syscall tampering attacks. Keystone's authors state that their solution is an improvement on SGX [56], and the measures they took to mitigating the risks that were exposed in Intel's solution seem to confirm that. However, Keystone, being based on a similar concept to SGX, can still demonstrate similar vulnerabilities that may yet to be discovered.

Using concepts presented in Keystone, or even the entire solution, seems like a good starting point, but it is hardly enough in itself to state that a device is secure in the IoT field. It certainly offers some degree of flexibility. Keystone being an open-source solution can also be an advantage. However, as presented in Table 2, there are many important categories that are left completely at the system designer's discretion and it can result in unintentional exposures of the IoT device.

4.4. OpenTitan

OpenTitan is undoubtedly one of the solutions to secure IoT devices that should attract designers' attention. This open source Hardware Root of Trust implementation [57] is endorsed by leading non-profit, academic or commercial organizations such as lowRISC, ETH Zürich or Google. The project is fully transparent, its sources are available online and can be inspected by the broader community, which should improve its security.

OpenTitan core is currently under development [58] and multiple features are still missing from the early stage top-level [59]. However, the intentions of its creators are well documented and in the complete form it should be a robust solution for various systems' security as an HWRoT module supporting the secure boot procedure and implementing miscellaneous cryptographic primitives.

OpenTitan acts as an immutable HWRoT with secure boot procedure support. It implements multiple cryptographic primitives (AES, RSA, Elliptic Curve Cryptography—

ECC or keyed-Hash Message Authentication Code based on SHA-256 algorithm) that are used to verify the code of subsequent boot stages and authenticate the device during the ownership transfer process. OpenTitan provides FW to control these primitives, so they can be used for securing communication confidentiality and integrity, but it is a responsibility of higher software layers. OpenTitan implements several tamper protection and detection mechanisms as protection codes or scrambling memory regions that contain secrets. Core implements hardware Random Number Generators that are compliant with international standards (e.g., NIST [60,61]).

OpenTitan provides low-level software that is responsible for first stages of the secure boot process (Silicon Creator level—Figure 4). At the beginning, execution is restricted to the ROM region (which cannot be modified after silicon is manufactured) and then ROM_ext part is loaded from flash (provided that integrity check has passed). At this point, execution is transferred to the entry point of the Silicon Owner code and from now on *it is the Owner who is responsible* for further boot stages security. It is also the end user's software duty to ensure isolation of the applications.

This solution offers the means to securely update its firmware—the integrity of the update block is verified prior to the reboot. The capability to update higher layers of software can be implemented using available cryptographic primitives; however, it is up to the end user.

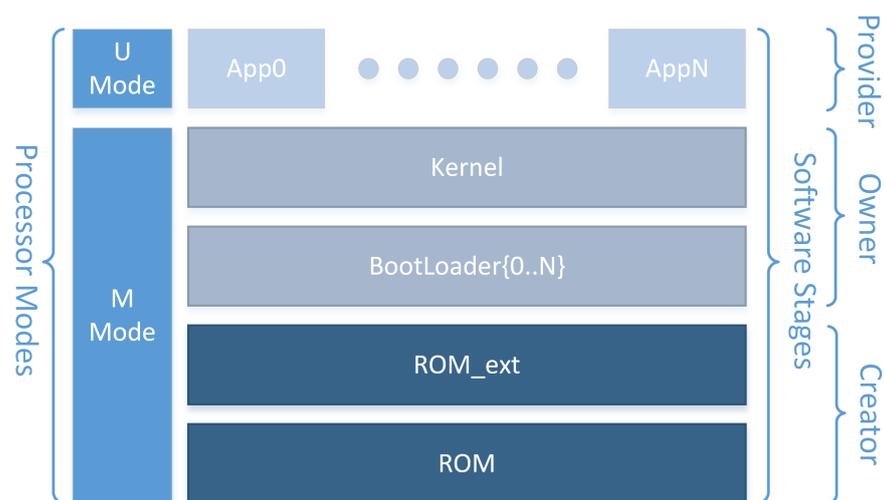


Figure 4. Software stages of the OpenTitan secure boot procedure with respect to particular levels owners, based on [62]. Only low level software layers (ROM, ROM_ext) are provided with the solution (Silicon Creator level), while the end user (Silicon Owner) is responsible for all higher phases of secure boot as well as the isolation of executed applications when the system is already up.

Analysis for Synthesized Requirements for Secure IoT Devices

We attempted to analyze the final set of OpenTitan capabilities against requirements presented in Table 1. The OpenTitan documentation does not mention any means for logging of cybersecurity events, neither in hardware nor in software. Possibly, some mechanisms might be implemented by end user in higher layers of software.

As was already mentioned, OpenTitan offers a rich suite of cryptographic primitives. Algorithm agility may still be improved, especially if it comes to lightweight cryptography, but it might be that OpenTitan does not target resource-constrained devices. Robust key derivation mechanisms and management schemes are available.

In summary, the fully operational OpenTitan core meets the assumptions of its designers—it is a solid foundation for maintaining the trust and integrity of the system, where it is instantiated. However, it still might not be enough to completely secure IoT devices in terms of recommendations from Table 1. It is not an out-of-the-box solution

for all IoT device security aspects and the potential end user has to implement many missing features.

4.5. NXP EdgeLock SE050

NXP EdgeLock SE050 Plug and Trust Secure Element is presented as a “ready-to-use IoT secure element solution” by the manufacturer [63]. It is an auxiliary security device that connects to host. Optional connections to a sensor mode via another I2C interface and native contactless antenna, granting wireless access are also available (Figure 5). Interestingly, EdgeLock SE050 holds Common Criteria’s EAL6+ certificate. It is advertised as suitable for smart cities, smart home, smart industry and smart supply chains. This solution combines a HWRoT with a Java Card OpenPlatform OS (JCOPOS), on which an IoT Applet runs. The Applet supports a wide range of secure functionalities, for example, random number generation, key management, hash operations, Platform Configuration Register (PCR) creation and management [64], and is provided by the manufacturer. In hardware, many configurable cryptographic primitives are included, supporting a wide range of algorithms and operations to choose from: HMAC, CMAC, SHA-1, RSA, ECC, AES. Lightweight cryptography is also available. What distinguishes NXP SE050 from other discussed solutions is the fact that it is a separate integrated circuit in its own packaging. This implicates the need for a dedicated space on the circuit board. It is also the only analyzed technology that supports SmartCard.

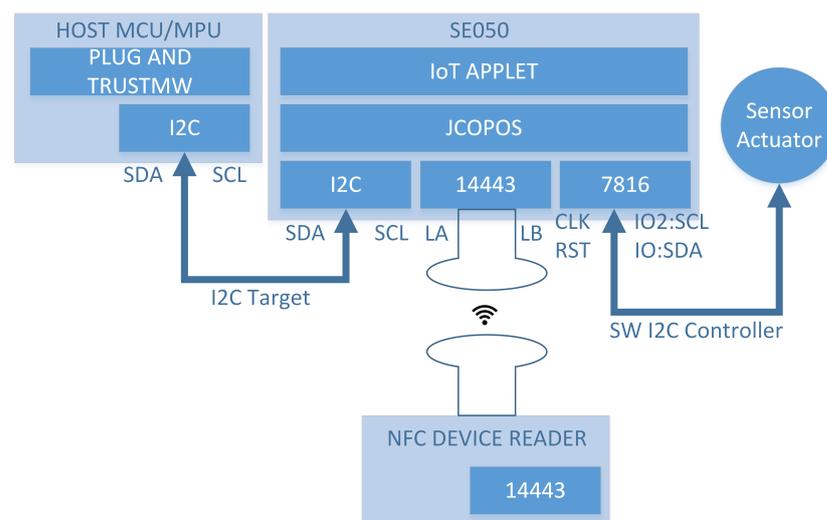


Figure 5. NXP SE050 architecture scheme, based on [63]. The communication between SE050 and Host via I2C bus is presented. The additional interfaces for optional applications, i.e., the antenna and the extra I2C to communicate with Sensor Actuator, are marked. The software/firmware scheme is also present in the form of the IoT Applet, running on Java Card OpenPlatform Operating System (JCOPOS).

Analysis for synthesized requirements for secure IoT devices

Some capabilities required for a secure IoT device hardware are present, but unfortunately as a plug-and-play approach the SE050 can provide a false sense of security, if used inappropriately. No safety is ensured for the host OS, which could in reality be wrongly configured and utterly unsafe with other interface connections that bypass the secure element—each communication channel should be trustworthy to a degree required in a protected IoT device and each communication channel should have the possibility to be disabled if not needed as per Table 1. No approach to software and firmware updating and provisioning is presented. It is hard to assess the safety of the I2C communication channel and possible problems. Additionally, neither software trusted execution nor enclave creation is possible via the applet. It is uncertain whether the host can freely access and use the cryptographic functionalities (including the random number generator) or if they are only available for SE050 internal access. Remarkably, the data sheet states that only

the Pseudo Random Number Generator is available [63]. Tamper detection is mentioned, but there is no information on tamper resistance. Moreover, no information is provided about the execution privileges in the Java OS nor about what information could be gathered through the secure logging of modules inside the secure element. If only the secure element interfaces are used for off chip communication, then they should be safe. Even though a wide range of secure functionalities is presented in the data sheet, a variety of use cases is discussed on NXP's website, cryptographic hardware primitives, coherent with standards and certifications, are implemented—this cannot be considered as a plug-and-play secure IoT solution. In fact, software must be developed carefully to ensure a proper level of security whilst using the SE050. By and large, the solution seems more like an extensively functional HWRoT with communication between two operating systems rather than an IC offering complex security for IoT.

4.6. Beyond Semiconductor GEON SoC Security Platform

The GEON SoC Security Platform manufactured by Beyond Semiconductor is presented as a processor agnostic solution with essentials for hardware IP security. An IoT application is advertised, including Industrial [65]. The SoC contains a suite of security modules that work together to create a secure IoT device but can also be used independently. This includes a customizable Hardware Root of Trust including secure boot functionality (GEON Secure Boot) and even recovery to a vendor software state in the case of a security breach [66], firmware encryption module (GEON Firmware Encryption) for the integrity and confidentiality of software, a module that stores and generates secret keys (GEON Hardware Security Module (HSM)) and hardware cryptographic operations module, including random number generation. A wide range of cryptographic algorithms is supported, including lightweight. Code authentication and cryptography-supported measurement functionalities are available. Interestingly, GEON SoC Secure Platform offers GEON Secure JTAG that is claimed to offer a complete debug analysis without compromising security [67]. As presented in Figure 6, the communication channel for the SoC Security Platform is realized by AMBA interface (AHB/APB), which may impact the final design of the IoT design.

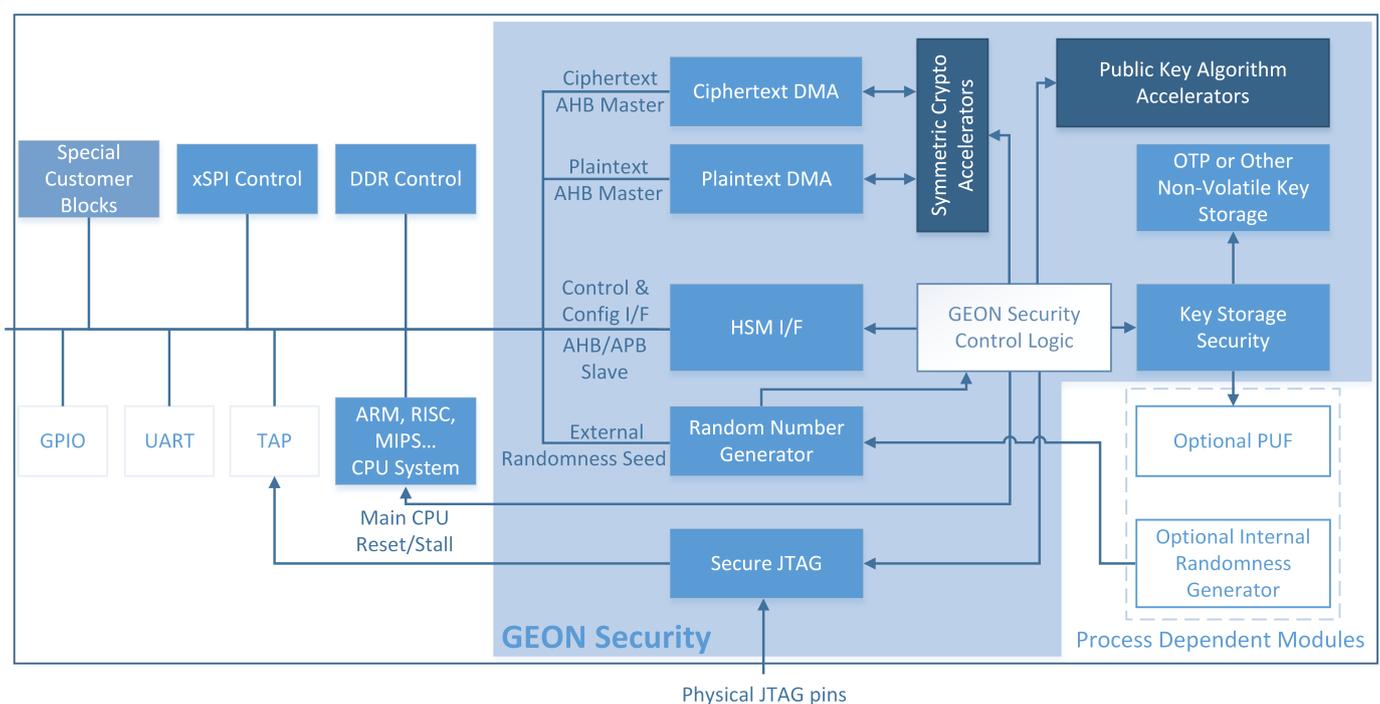


Figure 6. GEON SoC Security Platform hardware architecture, based on [65]. External interfaces and suggested connections to other components of System on Chip are presented.

Analysis for Synthesized Requirements for Secure IoT Devices

Multiple cryptographic ciphers and hash functions are available for use. The debug access is described in some documents mentioned in Section 3 as possibly unsafe and it is advised to disable it after deployment. It should be stressed, though, that these guidelines do not consider a highly secured debug interface and that is the case in the GEON SoC. Agreeably, this functionality can be critical in some applications, and the level of security claimed by Beyond Semiconductors is very promising. Parts of the GEON SoC are described as “flexible” and “disposable” depending on the application, which can obviously be beneficial for system designers, but the lack of awareness can lead to the creation of an unsecured IoT. Additionally, some key secure functionalities could be missing if one of the modules is left behind. No information is provided about cybersecurity event logging and monitoring. In terms of secure updating, firmware downgrade protection is in place; the confidentiality and integrity of software and firmware are also protected. GEON SoC Platform Security does not introduce any interference with other interfaces existing in the system in which it is integrated, mainly no disabling or securing capabilities are present. The security threat of any interface in the IoT device is unaccounted for. It can be considered as a plug in to the main bus of SoC. This solution certainly offers a great variety of hardware-based security features, but the lack of guidance on software and firmware development can be concerning.

4.7. Rambus RT Family

Rambus has a wide range of security IPs in its portfolio. We decided to focus on the Root of Trust IP cores, especially those dedicated for IoT—RT-100 [68], RT-130 [69], RT-140 [70] and RT-260 [71]. It seems that the difference between the aforementioned modules is the supported cryptography and protocols. The rule of operation is the same. These products are advertised as complete Hardware Root of Trust engines, offering cryptographic accelerators (both symmetric and asymmetric, in RT-140 also lightweight), secure boot support, secure asset storage, secure firmware upgrade, device authentication and identity protection, as well as secure debug. RT-140 supports TLS protocols. Remarkably, the Rambus solution is architecture agnostic. It can be integrated in an SoC and is claimed to be compatible with most popular system buses (e.g., AMBA). In order to cooperate with the device, the CPU requires a dedicated API implementation. Figure 7 illustrates the architecture and the location in the system of the Rambus component. Unfortunately, apart from the manufacturer’s materials, there is no literature openly available regarding the RT family. Therefore, our research relies solely on product briefs published on the Rambus website. The analyzed documents indicate that Rambus offers the RoT Secure Boot Toolkit, which may be either a firmware or a software extension to communicate with the instantiated hardware module. It interfaces with protected software stored in non-volatile memory and includes a signing tool and boot library.

Analysis for Synthesized Requirements for Secure IoT Devices

Since this is purely a RoT solution, the development of secure software and firmware is left completely to the designer. One of the greatest advantages is the wide range of supported cryptography offered. However, there is very little information about securing memory and data transfers. In conclusion, the Rambus proposition is a foundation for building a secure IoT, not a complete solution.

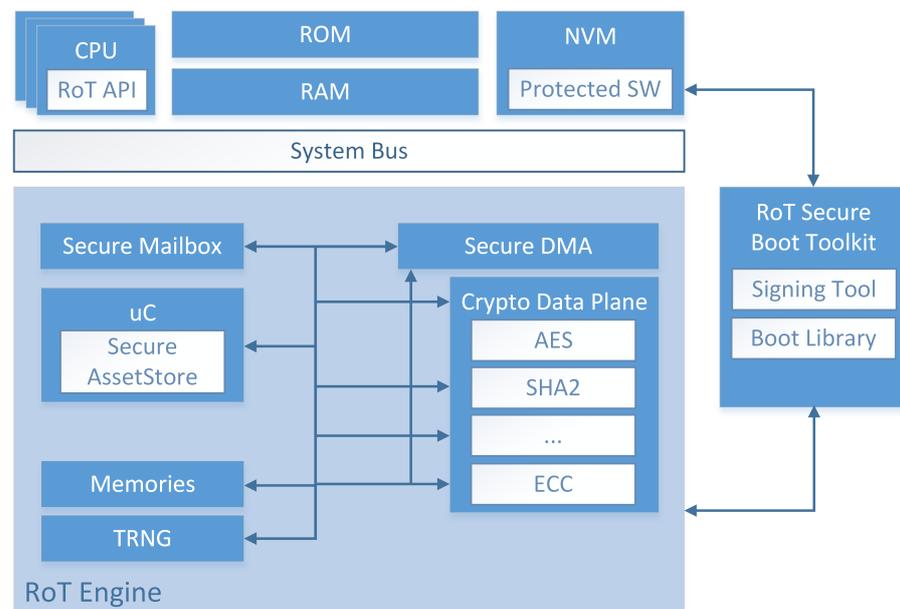


Figure 7. Rambus RT hardware architecture, on RT-100 example, based on [68]. Corresponding software elements (e.g., RoT Secure Boot Toolkit), and the location of this component in a system on chip are presented.

4.8. Summary

Table 2 repeats the requirements from Table 1 and compares the solutions presented in previous sections against them. For clarity, shortened versions of the requirements are presented below; the full description is available in Table 1.

Symbols used in Table 2 have the following meanings:

- ○—difficult to satisfy this secure IoT requirement using a particular solution;
- ◐—can be done, but no explicit recommendation nor solution for use in an IoT device is presented, leaving the design decisions open and potentially unsafe;
- ●—strong recommendation or solution which satisfies the requirement.

For the TrustZone, in some cases, ●* is used. This means that the requirement is fulfilled on the condition that TrustZone is combined with CryptoCell. For NXP the evaluation is based on the use case scenario from the data sheet [63] coined “Plug and Trust”—the integration with a host processing unit.

Table 2. Comparison of the state-of-the-art against the synthesized requirements for secure IoT devices.

No.	Requirement	Arm TrustZone	Intel SGX	Keystone	Open Titan	NXP SE050	GEON SoC	Rambus RT
Cat. 1	Hardware Security							
1.1	Immutable Hardware Root of Trust	●*	●	◐	●	●	●	●
1.2	HW provided security features	●	◐	◐	●	●	●	●
1.3	Tamper protection and detection	●*	●	◐	●	●	◐	◐
1.4	Random Number Generator (hardware-based if feasible)	●*	●	◐	●	◐	●	●
Cat. 2	Trust and integrity management							
2.1	Secure boot process based on HWRoT.	●	●	◐	◐ / ●	◐	●	●
2.2	Software (code, applications) signed cryptographically and verified upon installation or execution	●*	●	◐	◐	◐	●	◐
2.3	Restore last known secure state (e.g., firmware rollback, when code is verified as damaged or tampered)	●*	◐	◐	●	○	●	○
2.4	Software installation control in OS, to prevent unauthenticated software and files from being loaded onto it	●*	◐	◐	◐	○	●	◐
Cat. 3	Data protection and software design							
3.1	Encryption of data storage medium.	◐	●	◐	◐	◐	◐	◐
3.2	Input data validation	◐	●	◐	◐	◐	◐	○
3.3	Process privilege minimization	◐	◐	●	◐	○	○	○
3.4	Memory compartmentalization, process isolation	●	●	●	◐	○	○	○

Table 2. Cont.

No.	Requirement	Arm TrustZone	Intel SGX	Keystone	Open Titan	NXP SE050	GEON SoC	Rambus RT
Cat. 4	Device configuration and software update							
4.1	Changing the device's FW and SW configuration by authorized entities	●*	●	◐	●	◐	●	◐
4.2	Updating device's FW and SW through remote or local means by authorized entities	◐	●	◐	●	◐	◐	◐
4.3	Update file must be encrypted, signed and device verifies its integrity before application	●*	●	◐	●	◐	●	●
4.4	Automatic update capability	◐	◐	◐	◐	◐	◐	◐
4.5	Backward compatibility of updates	◐	◐	◐	◐	◐	◐	◐
Cat. 5	Secure interfaces and communication							
Interface security								
5.1	Ability to disable or logically restrict access to any local and network interfaces	◐	◐	◐	◐	○	○	○
5.2	Deployment without debugging, diagnostic or testing interfaces that could be abused by adversary	◐	◐	◐	◐	◐	●	◐
Protocol security								
5.3	Communication security using state-of-the-art, standardized security protocols (e.g., TLS for encryption)	●*	●	◐	◐	●	◐	●
5.4	Mutual authentication of the devices before trust can be established (prevent man in the middle attack)	◐	◐	◐	◐	◐	◐	◐
5.5	Prevent unauthorized connections to product or other devices it is connected to, at all levels of the protocols.	◐	◐	◐	◐	◐	◐	○
Data security								
5.6	Confidentiality, integrity and authenticity of the transmitted data	●*	◐	◐	◐	◐	◐	◐

Table 2. Cont.

No.	Requirement	Arm TrustZone	Intel SGX	Keystone	Open Titan	NXP SE050	GEON SoC	Rambus RT
Cat. 6	Cybersecurity event monitoring and logging							
6.1	Logging cybersecurity events from device's HW, FW and SW	●	●	●	○/●	○	●	●
6.2	Recording sufficient details for each logged event	●	●	●	○/●	○	●	●
6.3	Restricting access to the logs to authorized entities only and prevent all entities from editing them	●	●	●	○/●	●	●	●
6.4	Avoid security issues when designing error messages.	●	●	●	●	●	●	●
6.5	Device's behavior should be monitored and compared with model to detect anomalies	●	●	●	●	●	●	●
Cat. 7	Cryptography and key management							
7.1	Use of Standard Cryptographic Algorithms and Security Protocols	●*	●	●	●	●	●	●
7.2	Security protocols should support algorithm agility	●*	●	●	●	●	●	●
7.3	Length of the key should provide strong security, make brute-force attack infeasible	●	●	●	●	●	●	●
7.4	Every device must be instantiated with unique private key(s)	●*	●	●	●	●	●	●
7.5	Secure and scalable management of cryptographic keys (generation, storage, distribution)	●*	●	●	●	●	●	●
Cat. 8	Device identification, authentication, strong default security							
8.1	Unique physical identifier only authorized entities can access	●*	●	●	●	●	●	●
8.2	Device must be provisioned with unique logical identifier	●	●	●	●	●	●	●

Table 2. Cont.

No.	Requirement	Arm TrustZone	Intel SGX	Keystone	Open Titan	NXP SE050	GEON SoC	Rambus RT
8.3	Robust authentication and authorization schemes	☐	☐	☐	●	☐	☐	☐
8.4	Strong, device-individual default passwords that has to be changed by the user during initial setup	☐	☐	☐	☐	☐	☐	☐
8.5	Limiting number of invalid login attempts or introducing incrementing delays between them	☐	☐	☐	☐	☐	☐	☐
8.6	Availability of two-factor authentication	☐	☐	☐	●	☐	☐	☐
8.7	Product security shall be appropriately enabled by default.	☐	☐	☐	☐	☐	☐	☐

5. Discussion

In the current IoT security landscape, many institutions and entities are defining security requirements, but no industry-wide standard has been agreed upon. Device designers and vendors have their own proprietary solutions, which address some issues but miss the target in others. To the best of our knowledge, no solution addresses all the requirements presented in Table 1 out-of-the-box which generally leaves the solving of security problems mainly in the hands of the designing entity with the potential to create unsecured IoT devices.

Additionally, there is no industry wide standardization of the *level* of security needed, which would aid the designers of IoT devices. A different security level is required for traffic light control in a smart city than for an “intelligent” and connected toothbrush. This should also be part of the discussion and possibly even have a certification entity. Table 1 could be divided into different security levels and additionally be configurable within some of the requirements, that is, the key strength or chosen encryption algorithm should be implicitly standardized for different security needs.

Some solutions presented in Section 4 are, in theory, processor/architecture agnostic, but others like Arm TrustZone or Intel SGX are tied in with a specific architecture, which makes the security improvements dependent on the designing entity and the agility of their bug-fixing process. Additionally, the solutions in the current environment are generally divided into more hardware or software heavy, even though a combined solution of strong TEE and software process isolation with hardware cryptographic implementations and Hardware Root of Trust functionalities would seem to be the most secure. In our opinion, the discussed open-source solutions, Keystone and OpenTitan, could work together to create a highly secure prototype of a framework for IoT device protection, whilst being publicly attestable and with all the benefits of independence.

Many solutions in the present state-of-the-art fulfill a subset of secure IoT device requirements, but none adheres to all of them. No common approach is present for many application-level IoT requirements (Table 1), such as the return to a secure state, interface disabling in the SoC, automatic update by default and the logging of IoT device security events and others. This is an issue during the design of such devices and could have a real-world impact in the future. All current solutions focus on the wider trusted computing paradigm or on providing a hardware cryptographic acceleration, rather than being dedicated explicitly to IoT devices.

A comprehensive design framework is needed to assist in the design of secure IoT devices. Such a solution should take into consideration the most important security requirements shown in Section 3. We believe that it would, in a way, simplify the technology development and mitigate the problem of time-to-market/functionality trade-off in opposition to security. It should be possible to add, remove or configure IoT security on a block system level in accordance with the target application of the device.

6. Conclusions

The IoT ecosystem poses new security challenges that extend beyond traditional data security. There is a need for IoT security guidelines, and numerous institutions across the globe have proposed their recommendations, aiming to help ensure a secure IoT infrastructure. Device designers and vendors have their own proprietary solutions, which address some issues, but miss the target in others. In this paper, selected recommendations have been analyzed and compiled into a set of the most common and important considerations, divided into eight categories. The evaluation of representative examples from IoT security technologies against these criteria shows that there are solutions with the potential to meet all these recommendations, but at the moment no solution addresses all requirements in an out-of-the-box capacity, which allows for further research in this field.

Author Contributions: Conceptualization, M.C., M.K., F.K. and M.R.; methodology, M.C., M.K. and F.K.; investigation, M.C., M.K. and F.K.; writing—original draft preparation, M.C., M.K. and

F.K.; writing—review and editing, K.S. and M.R.; visualization, M.K.; supervision, M.R.; project administration, M.R.; funding acquisition, M.R. All authors have read and agreed to the published version of the manuscript.

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

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

AC	Air Conditioning
AES	Advanced Encryption Standard
AHB	AMBA High-performance Bus
AMBA	ARM Advanced Microcontroller Bus Architecture
APB	AMBA Advanced Peripheral Bus
API	Application Programming Interface
AXI	AMBA Advanced eXtensible Interface
CCTV	Closed-circuit Television
CLI	Command Line Interface
CMAC	Cipher-based Message Authentication Code
CPU	Central Processing Unit
DDoS	Distributed Denial of Service
DMA	Direct Memory Access
ECC	Elliptic-curve cryptography
FW	Firmware
GSM	Global System for Mobile Communications
HMAC	keyed-Hash Message Authentication Code
HW	Hardware
HWRoT, RoT	(Hardware) Root of Trust
I2C	Inter-Integrated Circuit communication bus
IC	Integrated Circuit
IoT	Internet of Things
IP	Intellectual Property
JCOPOS	Java Card OpenPlatform Operating System
JTAG	Joint Test Action Group
M-mode	RISC-V Machine Mode
NIST	National Institute of Standards and Technology
OS	Operating System
REE	Rich Execution Environment
RFID	Radio-frequency Identification
ROM	Read-only Memory
RSA	Rivest–Shamir–Adleman (public key cryptosystem)
S-mode	RISC-V Supervisor Mode
SGX	(Intel) Software Guard Extensions
SHA-1	Secure Hash Algorithm 1
SoC	System-on-Chip
SW	Software
TCB	Trusted Computing Base
TEE	Trusted Execution Environment
TLS	Transport Layer Security
TPM	Trusted Platform Module
U-mode	RISC-V User Mode

References

1. Ericsson. IoT Connections Outlook. Available online: <https://www.ericsson.com/en/mobility-report/dataforecasts/iot-connections-outlook> (accessed on 24 June 2021).
2. Lueth, K.L. State of the IoT 2020: 12 billion IoT Connections, Surpassing Non-IoT for the First Time. Available online: <https://iot-analytics.com/state-of-the-iot-2020-12-billion-iot-connections-surpassing-non-iot-for-the-first-time/> (accessed on 24 June 2021).
3. Neshenko, N.; Bou-Harb, E.; Crichigno, J.; Kaddoum, G.; Ghani, N. Demystifying IoT Security: An Exhaustive Survey on IoT Vulnerabilities and a First Empirical Look on Internet-Scale IoT Exploitations. *IEEE Commun. Surv. Tutor.* **2019**, *21*. [CrossRef]
4. Greenberg, A. Hackers Remotely Kill a Jeep on the Highway—With Me in It. Available online: <https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/> (accessed on 24 June 2021).
5. Graff, G.M. How a Dorm Room Minecraft Scam Brought down the Internet. Available online: <https://www.wired.com/story/mirai-botnet-minecraft-scam-brought-down-the-internet/> (accessed on 24 June 2021).
6. Ben Herzberg, I.Z.; Bekerman, D. Breaking Down Mirai: An IoT DDoS Botnet Analysis. Available online: <https://www.imperva.com/blog/malware-analysis-mirai-ddos-botnet/> (accessed on 24 June 2021).
7. Green, A. The Mirai Botnet Attack and Revenge of the Internet of Things. Available online: <https://www.varonis.com/blog/the-mirai-botnet-attack-and-revenge-of-the-internet-of-things/> (accessed on 24 June 2021).
8. Bonomi, F.; Milito, R. Fog Computing and its Role in the Internet of Things. *Proc. MCC Workshop Mob. Cloud Comput.* **2012**. [CrossRef]
9. Gupta, N.; Rodrigues, J.J.P.C.; Dauwels, J. (Eds.) *Augmented Intelligence toward Smart Vehicular Application*; CRC Press: Boca Raton, FL, USA, 2020; pp. 7–13. [CrossRef]
10. Park, H.; Zhai, S.; Lu, L.; Lin, F.X. StreamBox-TZ: Secure Stream Analytics at the Edge with TrustZone. In Proceedings of the 2019 USENIX Annual Technical Conference (USENIX ATC 19), Renton, WA, USA, 10–12 July 2019; pp. 537–554.
11. Fournaris, A.P.; Alexakos, C.; Anagnostopoulos, C.; Koulamas, C.; Kalogeras, A. Introducing Hardware-Based Intelligence and Reconfigurability on Industrial IoT Edge Nodes. *IEEE Des. Test* **2019**, *36*, 15–23. [CrossRef]
12. Mahmoud, R.; Yousuf, T.; Aloul, F.; Zualkernan, I. Internet of things (IoT) security: Current status, challenges and prospective measures. In Proceedings of the 2015 10th International Conference for Internet Technology and Secured Transactions, London, UK, 8–10 December 2016; pp. 336–341. [CrossRef]
13. binti Mohamad Noor, M.; Hassan, W.H. Current research on Internet of Things (IoT) security: A survey. *Comput. Netw.* **2019**, *148*, 283–294. [CrossRef]
14. Alladi, T.; Chamola, V.; Sikdar, B.; Choo, K.K.R. Consumer IoT: Security Vulnerability Case Studies and Solutions. *IEEE Consum. Electron. Mag.* **2020**, *9*, 17–25. [CrossRef]
15. Tsiropoulou, E.E.; Baras, J.S.; Papavassiliou, S.; Qu, G. On the Mitigation of Interference Imposed by Intruders in Passive RFID Networks. In *Decision and Game Theory for Security*; Zhu, Q., Alpcan, T., Panaousis, E., Tambe, M., Casey, W., Eds.; Springer International Publishing: Cham, Switzerland, 2016; pp. 62–80.
16. Mohanta, B.K.; Jena, D.; Satapathy, U.; Patnaik, S. Survey on IoT security: Challenges and solution using machine learning, artificial intelligence and blockchain technology. *Internet Things* **2020**, *11*, 100227. [CrossRef]
17. Zhu, Q.; Başar, T. Game-Theoretic Approach to Feedback-Driven Multi-stage Moving Target Defense. In *Decision and Game Theory for Security*; Das, S.K., Nita-Rotaru, C., Kantarcioglu, M., Eds.; Springer International Publishing: Cham, Switzerland, 2013; pp. 246–263.
18. Fagan, M.; Megas, K.N.; Scarfone, K.; Smith, M. IoT device cybersecurity capability core baseline. In *Technical Report*; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2020. [CrossRef]
19. Fortino, G.; Savaglio, C.; Spezzano, G.; Zhou, M. Internet of Things as System of Systems: A Review of Methodologies, Frameworks, Platforms, and Tools. *IEEE Trans. Syst. Man Cybern. Syst.* **2021**, *51*, 223–236. [CrossRef]
20. Tournier, J.; Lesueur, F.; Mouël, F.L.; Guyon, L.; Ben-Hassine, H. A survey of IoT protocols and their security issues through the lens of a generic IoT stack. *Internet Things* **2020**, 100264. [CrossRef]
21. Sinche, S.; Raposo, D.; Armando, N.; Rodrigues, A.; Boavida, F.; Pereira, V.; Silva, J.S. A Survey of IoT Management Protocols and Frameworks. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 1168–1190. [CrossRef]
22. Babun, L.; Denney, K.; Celik, Z.B.; McDaniel, P.; Uluagac, A.S. A survey on IoT platforms: Communication, security, and privacy perspectives. *Comput. Netw.* **2021**, *192*, 108040. [CrossRef]
23. Ammar, M.; Russello, G.; Crispo, B. Internet of Things: A survey on the security of IoT frameworks. *J. Inf. Secur. Appl.* **2018**, *38*, 8–27. [CrossRef]
24. Macedo, E.L.; De Oliveira, E.A.; Silva, F.H.; Mello, R.R.; Franca, F.M.; Delicato, F.C.; De Rezende, J.F.; De Moraes, L.F. On the security aspects of Internet of Things: A systematic literature review. *J. Commun. Netw.* **2019**, *21*, 444–457. [CrossRef]
25. Abdul-Ghani, H.A.; Konstantas, D. A comprehensive study of security and privacy guidelines, threats, and countermeasures: An IoT perspective. *J. Sens. Actuator Netw.* **2019**, *8*, 22. [CrossRef]
26. Maene, P.; Götzfried, J.; De Clercq, R.; Müller, T.; Freiling, F.; Verbauwhede, I. Hardware-Based Trusted Computing Architectures for Isolation and Attestation. *IEEE Trans. Comput.* **2018**, *67*, 361–374. [CrossRef]

27. European Union Agency for Network and Information Security. Baseline Security Recommendations for IoT in the Context of Critical Information Infrastructures. Available online: <https://op.europa.eu/en/publication-detail/-/publication/c37f8196-d96f-11e7-a506-01aa75ed71a1/language-en> (accessed on 30 June 2021).
28. GSM Association. IoT Security Guidelines Overview Document—Version 2.2. Available online: <https://www.gsma.com/iot/wp-content/uploads/2020/05/CLP.11-v2.2-GSMA-IoT-Security-Guidelines-Overview-Document.pdf> (accessed on 30 June 2021).
29. GSM Association. IoT Security Guidelines for IoT Service Ecosystem—Version 2.2. Available online: <https://www.gsma.com/iot/wp-content/uploads/2020/05/CLP.12-v2.2-GSMA-IoT-Security-Guidelines-for-Service-Ecosystems.pdf> (accessed on 30 June 2021).
30. GSM Association. IoT Security Guidelines for Endpoint Ecosystem—Version 2.2. Available online: <https://www.gsma.com/iot/wp-content/uploads/2020/05/CLP.13-v2.2-GSMA-IoT-Security-Guidelines-for-Endpoint-Ecosystems.pdf> (accessed on 30 June 2021).
31. Moore, K.; Barnes, R.; Tschofenig, H. Best Current Practices for Securing Internet of Things (IoT) Devices. Internet-Draft draft-moore-iot-security-bcp-01. *Internet Eng. Task Force* **2017**, Work in Progress.
32. Wang, B.; Liu, S.; Wan, L.; Li, J.; Wang, X.T. Technical Requirements for Secure Access and Management of IoT Smart Terminals. Internet-Draft draft-wang-secure-access-of-iot-terminals-01. *Internet Eng. Task Force* **2021**, Work in Progress.
33. Garcia-Morchon, O.; Kumar, S.; Sethi, M. Internet of Things (IoT) Security: State of the Art and Challenges. Available online: <https://www.rfc-editor.org/info/rfc8576> (accessed on 30 June 2021).
34. Foundation, I.S. Secure Design—Best Practice Guides—Release 2. Internet of Things (IoT) Security: State of the Art and Challenges. Available online: https://www.iotsecurityfoundation.org/wp-content/uploads/2019/12/Best-Practice-Guides-Release-2_Digitalv3.pdf (accessed on 30 June 2021).
35. ioXt Alliance. ioXt Pledge—The Global Standard for IoT Security. Available online: https://static1.squarespace.com/static/5c6dbac1f8135a29c7fbb621/t/5fb43a05bda7c3689ea5bd32/1605646853608/ioXt+Pledge+Book_Secure.pdf (accessed on 30 June 2021).
36. International Organization for Standardization. Cybersecurity—IoT Security and Privacy—Guidelines. Available online: <https://www.iso.org/standard/44373.html> (accessed on 30 June 2021).
37. Trusted Computing Group. Trusted Computing Group Glossary. Version 1.1. Available online: <https://trustedcomputinggroup.org/wp-content/uploads/TCG-Glossary-V1.1-Rev-1.0.pdf> (accessed on 30 June 2021).
38. GlobalPlatform, Inc. GlobalPlatform Security Task Force Root of Trust Definitions and Requirements. Available online: https://globalplatform.org/wp-content/uploads/2018/06/GP_RoT_Definitions_and_Requirements_v1.0.1_PublicRelease_CC.pdf (accessed on 30 June 2021).
39. Arm. Arm® Platform Security Requirements 1.0. Available online: <https://developer.arm.com/documentation/den0106/latest/> (accessed on 30 June 2021).
40. Pinto, S.; Santos, N. Demystifying Arm TrustZone: A comprehensive survey. *ACM Comput. Surv.* **2019**, *51*, 36. [CrossRef]
41. Arm. Learn the Architecture: TrustZone for AArch64. Available online: <https://developer.arm.com/documentation/102418/0100> (accessed on 24 June 2021).
42. Zhao, S.; Zhang, Q.; Hu, G.; Qin, Y.; Feng, D. Providing Root of Trust for ARM TrustZone using On-Chip SRAM. In Proceedings of the 4th International Workshop on Trustworthy Embedded Devices—TrustED '14, Scottsdale, AZ, USA, 3 November 2014; pp. 25–36.
43. Arm. CryptoCell-300 Family. Available online: <https://developer.arm.com/ip-products/security-ip/cryptocell-300-family> (accessed on 24 June 2021).
44. Arm. CryptoCell-700 Family. Available online: <https://developer.arm.com/ip-products/security-ip/cryptocell-700-family> (accessed on 24 June 2021).
45. Wallace, J. Arm CryptoCell-312: Simplifying the Design of Secure IoT Systems. 2016. Available online: <https://community.arm.com/developer/ip-products/system/b/embedded-blog/posts/arm-trustzone-cryptocell-312-simplifying-the-design-of-secure-iot-systems> (accessed on 24 June 2021).
46. Nordic Semiconductor. CRYPTOCELL—ARM® TrustZone® CryptoCell 310. 2021. Available online: https://infocenter.nordicsemi.com/index.jsp?topic=%2Fps_nrf52840%2Fcryptocell.html (accessed on 24 June 2021).
47. Anati, I.; Gueron, S.; Johnson, S.; Scarlata, V. Innovative technology for CPU based attestation and sealing. In *Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy*; ACM: New York, NY, USA, 2013; Volume 13, p. 7.
48. Intel SGX-Remote Attestation. Available online: <https://software.intel.com/content/www/us/en/develop/topics/software-guard-extensions/attestation-services.html> (accessed on 30 June 2021).
49. Costan, V.; Devadas, S. Intel SGX Explained. Available online: <https://eprint.iacr.org/2016/086.pdf> (accessed on 30 June 2021).
50. Intel SGX—Security Essentials Solution Brief. Available online: <https://www.intel.com/content/dam/www/public/us/en/documents/solution-briefs/intel-security-essentials-solution-brief.pdf> (accessed on 30 June 2021).
51. Intel SGX—IoT Edge Devices. Available online: <https://software.intel.com/content/www/us/en/develop/topics/software-guard-extensions.html> (accessed on 30 June 2021).

52. Brassier, F.; Müller, U.; Dmitrienko, A.; Kostianinen, K.; Capkun, S.; Sadeghi, A.R. Software Grand Exposure: {SGX} Cache Attacks are Practical. Available online: <https://www.usenix.org/conference/woot17/workshop-program/presentation/brassier> (accessed on 30 June 2021).
53. Van Bulck, J.; Minkin, M.; Weisse, O.; Genkin, D.; Kasikci, B.; Piessens, F.; Silberstein, M.; Wenisch, T.F.; Yarom, Y.; Strackx, R. Foreshadow: Extracting the keys to the intel {SGX} kingdom with transient out-of-order execution. In Proceedings of the 27th {USENIX} Security Symposium ({USENIX} Security 18), Baltimore, MD, USA, 15–17 August 2018, pp. 991–1008.
54. Chen, G.; Chen, S.; Xiao, Y.; Zhang, Y.; Lin, Z.; Lai, T.H. Sgxpectre: Stealing intel secrets from sgx enclaves via speculative execution. In Proceedings of the 2019 IEEE European Symposium on Security and Privacy (EuroS&P), Stockholm, Sweden, 17–19 June 2019; pp. 142–157.
55. Murdock, K.; Oswald, D.; Garcia, F.D.; Van Bulck, J.; Gruss, D.; Piessens, F. Plundervolt: Software-based fault injection attacks against Intel SGX. In Proceedings of the 2020 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 17–21 May 2020; pp. 1466–1482.
56. Lee, D.; Kohlbrenner, D.; Shinde, S.; Asanović, K.; Song, D. Keystone: An open framework for architecting trusted execution environments. In Proceedings of the Fifteenth European Conference on Computer Systems, Heraklion, Greece, 27–30 April 2020; pp. 1–16.
57. OpenTitan—Open Source Silicon Root of Trust. Available online: <https://opentitan.org/> (accessed on 30 June 2021).
58. OpenTitan—Hardware Dashboard. Available online: <https://docs.opentitan.org/hw/> (accessed on 30 June 2021).
59. OpenTitan—Earl Gray Top Level Specification. Available online: https://docs.opentitan.org/hw/top_earlgray/doc/ (accessed on 30 June 2021).
60. Barker, E.; Kelsey, J.; National Institute of Standards and Technology. NIST Special Publication 800-90A, Revision 1: Recommendation for Random Number Generation Using Deterministic Random Bit Generators. Available online: <https://csrc.nist.gov/publications/detail/sp/800-90a/rev-1/final> (accessed on 30 June 2021).
61. Barker, E.; Kelsey, J.; National Institute of Standards and Technology. NIST Special Publication 800-90C (Second Draft): Recommendation for Random Number Generator (RBG) Constructions. Available online: <https://csrc.nist.gov/publications/detail/sp/800-90c/draft> (accessed on 30 June 2021).
62. OpenTitan—Logical Security Model. Available online: https://docs.opentitan.org/doc/security/logical_security_model/ (accessed on 30 June 2021).
63. NXP. SE050 Plug and Trust Secure Element. Available online: <https://www.nxp.com/docs/en/data-sheet/SE050-DATASHEET.pdf> (accessed on 30 June 2021).
64. NXP. AN12413-SE050 APDU Specification—Application Note—Rev. 2.12. Available online: <https://www.nxp.com.cn/docs/en/application-note/AN12413.pdf> (accessed on 30 June 2021).
65. Beyond Semiconductor. GEON SoC Security Platform. 2021. Available online: <https://www.beyondsemi.com/116/geon-security-platform/> (accessed on 30 June 2021).
66. Beyond Semiconductor. GEON Secure Boot. 2021. Available online: <https://www.beyondsemi.com/117/geon-secure-boot/> (accessed on 30 June 2021).
67. Beyond Semiconductor. GEON Secure JTAG. 2021. Available online: <https://www.beyondsemi.com/119/geon-secure-jtag/> (accessed on 30 June 2021).
68. Root of Trust RT-100. Foundational Security for SoCs and FPGAs. Available online: <https://go.rambus.com/root-of-trust-rt-100-product-brief> (accessed on 30 June 2021).
69. Root of Trust RT-130. Foundational Security for SoCs and FPGAs for IoT Servers, Gateways and Edge Devices. Available online: <https://go.rambus.com/root-of-trust-rt-130-product-brief> (accessed on 30 June 2021).
70. Root of Trust RT-140. Foundational Security for SoCs and FPGAs for Cloud-Connected Devices. Available online: <https://go.rambus.com/root-of-trust-rt-140-product-brief> (accessed on 30 June 2021).
71. Root of Trust RT-260. Foundational Security for SoCs and FPGAs for Secure MCU Devices. Available online: <https://go.rambus.com/root-of-trust-rt-260-product-brief> (accessed on 30 June 2021).