Hardware security for IoT identity assurance

André Cirne*, Patrícia R. Sousa*, João S. Resende†, Luís Antunes*
*Faculty of Science - University of Porto †NOVA School of Science and Technology

***Abstract*—The widespread use of Internet of Things (IoT) devices makes their security a priority. Among the different security challenges, identity and authentication mechanisms rise as the most important. Identity and authentication in IoT are limited by the device's computation capabilities and are more susceptible to physical attacks than common computers and servers. Regardless, identity and authentication mechanisms are essential for a secure system.**

Researchers have pointed out that hardware, the source of these limitations, may also be the solution to overcome these challenges. Systems may include hardware-based cryptographic implementations to overcome computation and energy limitations. On the other hand, the addition of security hardware can increase the resilience of a device against hardware and software attacks.

Our work aims to support these claims by exploring the physical attacks and other challenges that identity and authentication are subject to and analyzing possible technologies that may solve these issues.

For achieving this goal, we preformed a threat analysis to the IoT identity and used it to guided us through the research. For each technology, we identified: known security attacks, employed countermeasures, advantages and disadvantages for identity assurance. Additionally, we surveyed the literature for examples of these technologies supporting the device's identity.

Finally, we were able to create an objective comparison between the different technologies and identified challenges that are hampering the extensive use of hardware-based identity and authentication systems in IoT.

***Index Terms*—IoT, device's identity, identity, hardware-based identity, hardware trust anchors, hardware attacks**

I. Introduction

The Internet of Things is an environment of interconnected devices that uses the Internet to share data. Virtually any device can be connected to the network. Smartphones, wearables, motion sensors, cars, and smart home appliances are just a few examples of connected devices today. The number of Internet of Things (IoT) devices connected to the Internet continues to grow [1] and our daily lives already depend on them. In fact, they are even replacing us in factories, farms, and other jobs [2]. This trend will not decrease and, on the contrary, continues to increase, driven by the emergence of more modern technologies emerging, such as 5G [3], Big Data [4] or Fog Computing [5], that enable better connectivity to end devices, increasing network bandwidth, storage and computing resources. Therefore, an increase in the number of Internet of Things (IoT) devices is foreseeable in the coming years [3].

With the arising of these devices, the need for strong security policies and controls in the IoT life-cycle urges [6]. IoT devices are known to be insecure and the development of security solutions is considered one of its open research challenges [7], [8]. This problem is caused by a generalized lack of security standards, along with the desire for inexpensive systems, which means security is not a priority in their development [9]–[11].

Manuscript received April 19, 2021; revised August 16, 2021.

Among the different security challenges in this field [12], [13], device identity is one of the most fundamental to create a secure system. Authentication refers to the confirmation of the origin of an object or person, in this case, often related to the verification of its identity. Thus, identity management is the basis for secure authentication methods for IoT. Without these premises it is not possible to design a secure IoT system, as we lose the ability to access control resources in a system or guarantee the veracity of information received from a device [14]. Despite these facts, the issues we mentioned earlier limit the implementation of identity and authentication mechanisms, which make them custom, lacking peer review, and using slow or outdated cryptographic algorithms that do not offer the best security.

Furthermore, IoT devices are more susceptible to physical attacks than other devices, which has consequences for the creation of IoT authentication and identity systems. Unlike computers and servers, IoT devices are more likely to be deployed in unprotected locations where the attacker can have unrestricted physical access. A simple example is a smart meter placed inside the customer's home. His electric bill depends on the measurements of that device, which means that he is the person most motivated to tamper with the smart meter and has unrestricted access to it.

Approaching this problem in a general way, we can conclude that protection against physical attacks is fundamental for the production of secure IoT devices. These devices have two parts: software and hardware. The software implements all the logic of the device, while the hardware supports its execution and enables interactions with the physical world, which means that there is a relationship between them, as the software runs on top of the hardware. Regarding vulnerabilities, the industry has many tools to help us find them in software, such as code auditors, fuzzers, debuggers, and static analyzers, but there are fewer for hardware vulnerabilities. For this reason, hardware vulnerabilities are more difficult and slower to solve then their software counterparts [15].

To counterbalance, a root of trust is essentially a security process that starts with an immutable (unchangeable) hardware identity ingrained into the IoT device. A root of trust is an immutable process or identity used as the first entity in a chain of trust. For the most critical applications, a hardware root of trust can be an important building block for more secure IoT devices. Regardless, this creates a hardware trust anchor.

This type of construction has multiple advantages, namely guaranteeing the security of the device's identity from software and physical attacks, and enabling high assurance systems, due to the higher level of resilience it offers.

As these anchors are implemented at the hardware level, there may be the wrong perception that even if the device's software is fully compromised, the attacker will not be able to compromise these primitives [16]. However, if the design can not effectively resist to hardware attacks, hackers can easily obtain the secrets of the entire chip. Attackers can use the secrets to crack identity authentication and data encryption and steal product design know-how, causing application security problems.

Therefore, there is a big demand for the implementation of hardware-level security features [15]–[18].

A. Related Research Overview

Over the years, several literature reviews have been published on hardware security and identity that bear similarities to our work.

Yang et al. [19] reviewed different technologies that can be used to support the identification and authentication of IoT devices. Throughout their review, Yang et al. explored how each technology can be used as a building block for new systems and potential security attacks on those technologies. Our work also explores technologies with the same goal, but we include more technologies like Trusted Execution Environment (TEE)s and secure elements. Furthermore, unlike our work, Yang et al. do not compare the technologies presented.

Shepard et al. [20] reviewed technologies that allow safe and reliable execution according to the needs of IoT systems. During their analysis, they defined a threat model and evaluation criteria, focusing on the IoT use case, which was used to compare the different technologies. While our work follows a similar approach, we explore identity assurance via hardware security, which means that our analysis includes other technologies, different threats, and requirements.

Ehret et al. [21] have more generalized coverage of this topic. His research focused on hardware-based security techniques related to IoT devices. This research goes through the different components of a IoT device and presents their hardware security threats and possible mitigations. Hu et al. [22] followed the same research direction, but generalized further by presenting systems hardware security threats and their countermeasures. In addition, they reviewed security tools that can be used to verify device security, for example, to analyze at the hardware level how information flows within the board or to check if the device implementation respects its intended design. Contrarily to these works, our work analyzes the use of hardware security for a specific purpose, supporting the identification and authentication of IoT devices. In addition, we present a threat analysis specifically for this purpose and relate each threat to its identity assets. We also cover the different countermeasures for each security threat and map them onto the technologies presented.

Compared to these previous works, our research is intrinsically different from the research mentioned in this section. To the best of our knowledge, this research is the first that connects these two realities, identity and hardware security, analyzing threats from physical security to identity, identifying potential countermeasures, and mapping them to technologies that can be used to support the implementation of new solutions.

B. Contributions

Our work aims to provide a comprehensive analysis of hardware trust anchors that can be used to support the implementation of identity systems for IoT devices. In summary, the contributions of this research are:

- Threat analysis of physical risks to the IoT identity, which includes a definition of security assets, goals, threat actors, threats and hardware countermeasures.
- Analysis of different technologies that can be used as hardware trust anchors, containing their advantages and disadvantages, security concerns and examples of their use to support device identity.
- Discussion of challenges for adopting hardware trust anchors and future research directions.

C. Outline

The rest of this article is structured as follows:

The Section I-A describes similar research papers on hardware security and device identity, and how we differ from them. Section II gives a brief introduction to identity and authentication and their state of development regarding IoT. The Section III lists the different research challenges that can be retrieved from the literature related to the device identity and authentication process, and the IV section elaborates a threat analysis focused on physical risks to the device identity. In Section V, we enumerate and analyze the different technologies that can be used to support device identity. The Section VI compares the different technologies and relates them to the research challenges and security threats identified earlier. Finally, Section VIII provides a summary of our research.

II. Traditional vs IoT-based Identity Management

One of the challenges of IoT is identity management and the effective implementation of secure authentication mechanisms. These two are fundamental characteristics to design a system with security by default, as both are security primitives to implement further features [13].

Identity is a specific set of features that allow unique identification of an entity (something or someone). There are countless implementations regarding the field of application [23]. For example, a human identity could be a fingerprint or a social security number. In the case of IoT, it can be a serial number or a cryptographic key.

Authentication is the ability of an entity to prove that it is genuinely the entity it claims to be. Thus, authentication attests to the identity of an entity. In cryptography, authentication has two main classes: data source authentication and entity authentication. Authentication of data origin refers to information exchanges where it is necessary to ensure that the information and its source are immutable. As a result, data source authentication implies data integrity. In contrast, entity authentication is the corroboration of identity and does not include any message other than the claim to be a particular entity [24].

Depending on the literature, entity authentication is synonymous with identification or self-concept. For authors who consider identification an independent notion, identification is claiming a specific identity without presenting irrefutable evidence of this claim [24], [25]. In this work, we will treat authentication and entity identification as synonyms.

Related to identity and identification, we have Identity Management (IdM), which is the process of managing identity information and providing authentication and access control to information systems. IdM systems handle the relation between different parties, entities, Service Provider (SP), and Identity Provider (IdP). The entity is the one that claims an identity. The SP provides a service to the entity, and, finally, the IdP has three main functions, entity registration, identity storage, and authentication, which means it is responsible for the enrollment of new entities and also the authentication process whenever an entity needs to access a service [26], [27]. This fact makes the IdP the core component of an IdM.

On a IdM, an entity can have multiple identities characterized by different identifiers organized in three categories: something that only the entity and the IdP knows, such as a password; something that the entity owns, such as a serial number; and some physical characteristic of the entity, like fingerprints [26].

With the growth of IdM systems, they started to follow the isolated model - the traditional identity model in which SP and IdP functions merge, therefore, identification and authentication are done directly in the SP itself.

Isolated models are a management burden for organizations with multiple services, implying identities for each entity [27]. In the case of human identity, multiple credentials are required, decreasing user usability and weakening overall security. To respond to these problems, IdM systems began to simplify the user experience and its management, introducing the so-called centralized model.

In the centralized model, the SP is separated from the IdP. Herewith, different SPs can use the same IdP for authentication, and the entity has a single identity across multiple servers, which also eases its management.

Despite the advances made by the centralized model and its paradigms, there are still two problems. IdP servers struggle to scale, as the increase in the number of identities implies an increase in the compute and storage requirements. Also, the centralized model does not support inter-domain authentication, which is a usability problem for large enterprises [27].

The federated model solves these issues by integrating multiple IdPs in a single authentication domain - the federated authentication domain. This model is implemented by setting a group of agreements, standards, and technologies that enable a SP to recognize identities from other authentication domain's IdP or the creation of maps between identities from different IdPs [27]–[29].

Based on the federated model principles, multiple protocols support their implementation, such as SAML [30], OpenID connect [31], and full-fledged IdM systems, as Keycloak [32] and Shibboleth [33]. These are just a few examples of implementations, there are other frameworks and systems, but the ones presented here are the most widespread [34].

Security Assertion Markup Language (SAML) [30] is a XML-based protocol to exchange authentication and authorization data between a SP and a IdP, even when they are part of different authentication domains. This protocol relies on the SAML Assertion message, a XML containing all the information required by the SP about the entity and cryptographically signed by the IdP. The SP uses the IdP's public key to check the veracity of the message. Shibboleth [33] leverages the SAML protocol to implement a complete IdM solution, featuring federated Single Sign-On (SSO) capabilities.

OpenID Connect [31] is an authentication and authorization framework, which is implemented on top of the OAuth 2.0 authorization framework [35] by adding an identity layer that allows the exchange of identity information [31]. The protocol uses a REST API to delegate conditional access to entity data. The entity obtains an Access Token from the IdP that SP uses to access its identity information. Keycloak [32] is an example of a IdM system based on OpenID Connect.

The growing number of online users and accounts has driven the evolution of IdM models to be user-centric. Both protocols are examples of this paradigm. The user controls the information exchanged between SP and IdP, which allows users to have different identifiers linked to their identity, which can be shared with SPs according to their consent. Furthermore, this paradigm is being explored to create SSO experiences. The user only needs to authenticate once and will have access multiple services without having to re-enter their credentials [36].

Research continues to follow the user-centric paradigm, addressing privacy issues, such as a lack of control over personal data dissemination [37]. Others move away from the classic centralized model and make decentralized IdMs focusing on preserving user privacy [38]–[40].

With the emergence of IoT devices, the need for specific IoT IdM solutions has also increased [41]. IoT devices are inherently different from humans because there is a lack of identifiers, which makes it difficult to develop solutions.

Lam et al. [42] listed four types of characteristics that can be used to identify a IoT device: *inheritance*, *association*, *knowledge* and *context*; inheritance is the most hardware-dependent and immutable, and context is the most hardware-independent, but changeable.

The *inheritance* category is like human biometrics identifiers. It includes information dependent on the device's hardware and unique for each device. An example of an identifier of this type is a Physically Unclonable Functions (PUF) (more details in Subsection V-F).

The *association* category uses relationships between devices that are critical to their functioning. For example, if a wearable needs a connection to a smartphone to communicate with the Internet, the smartphone can be an identifier for the wearable.

The *knowledge* category is similar to the "something you know" for humans. However, the level of assurance is very different. A human can memorize a password and does not need to save it anywhere else. On the other hand, a device needs a mechanism to store the password securely, which implies risks.

Lastly, the *context* category uses IoT-based sensing data as identifiers. Context-aware computing connects much information found in the real world to the intelligence of the environment. For example, sensor readings produced by GPS sensors can be considered raw sensor data. Since we put the data from the GPS sensors in such a way that it represents a geographic location, we call it context information. However, it is necessary to consider the quality of context, which depends on the quality of the physical sensor, the context data, and the quality of the delivery process. So, these identifiers can have relatively less quality than others and introduce some challenges, such as when a device has an owner and multiple users or when the interactions with the devices change over time. Both produce changes in the identifiers that make them difficult to use [42].

Regardless of the identifiers used to authenticate a device, there is no universal identifier for IoT devices, which is a barrier to developing perfect IoT solutions. Although each resource on the Internet has a unique domain name or a public IP managed by international organizations, this does not exist in IoT, as each manufacturer has its standards and protocols. Therefore, in the short term, it is unlikely that a universal solution to this problem will emerge [43]. However, researchers are developing ways to authenticate and identify IoT devices.

Most IoT systems use cryptographic-based entity authentication, which means that device identifiers are used in conjunction with cryptographic algorithms to enable identity verification [44], [45].

An example of this type of authentication is *Attribute-Based Authentication* schemes, in which device attributes are used to generate a secret key in a public-key encryption scheme. Every time the device needs to authenticate, it encrypts a challenge sent by the server with its key. The server will then decrypt this message using the expected attributes to reproduce the device key, and if it can retrieve its challenge from the encrypted message, the device will be authenticated [42].

There are also systems using other approaches. For example, the use of private keys and Public Key Infrastructure (PKI) certificate for each device [46], or adapted versions of IdM systems to the IoT, mainly when it is necessary to authenticate devices and users [47].

Beyond that, researchers are working on blockchain-based solutions [13], [34]. These aim to create fault-tolerant IdMs, enabling unique identifiers to promote interoperability among different brands of devices [34].

As we stated earlier, most protocols rely on asymmetric cryptography and assume the availability of secure storage, which brings limitations for use in IoT. Asymmetric encryption is too heavy for low-end IoT processors, which makes encryption operations slow and energy-intensive [19], [48]. On the other hand, almost all IoT devices do not have secure storage available due to the inherent cost or expertise required to implement these features [17]. Therefore, if we want to continue to use solutions that rely on standard cryptographic algorithms, IoT must be assisted by hardware components that ease their execution and create the necessary security conditions.

III. Hardware-based IoT Identity Challenges

Since the beginning of IoT, device identity has been pointed out as an open research challenge [12], [49]. Assessing the requirements stated in the Section II, we can highlight three main research opportunities for identity management: **Lightweight Cryptography** [7], [8], [12], [49], **Object Identification** [8], [12], [41], [43], [49], and **Secure Storage** [43], [49], [50].

One of the limiting factors in IoT is its restricted resources, which limits the implementation of identity and authentication mechanisms. Several authors point out that **Lightweight Cryptography** can solve this problem [7], [8], [12] as lightweight cryptography is an encryption algorithm or protocol designed with restricted devices in mind. This type of solution is evaluated according to requirements such as energy consumption, implementation size, RAM, and computational power [51]. Lightweight cryptography does not necessarily imply a trade-off in security efficiency. Some researchers try to develop new approaches to cryptographic problems while respecting device constraints, and others use known algorithms and protocols and try to reduce them to meet the requirements of constrained devices [51].

Before designing any security system, it is necessary to be able to identify each device. An ideal identification solution should reflect the device's characteristics in its identification process [12]. For example, following the idea that IoT devices can connect to the Internet anytime and anywhere, the device identity should reflect these properties [52]. Also, regarding this issue, IoT promises that devices will communicate seamlessly regardless of manufacturer. However, the lack of standardization undermines this idea. Therefore, creating a vendor-independent identifier is a priority to overcome this problem.

Therefore, the challenge of **Object Identification** is being addressed through two different approaches. First, researchers and international organizations are trying to create a global naming scheme that multiple manufacturers can use and be able to identify a device, even when it is not connected directly to the Internet (for example, a sensor that connects to a gateway via Bluetooth) [49], [52]. On the other hand, researchers are exploring how we can define identity by analyzing the necessary resources and available technologies.

For solving the object identification challenge, the need for **Secure Storage** resources increases. As we stated previously in this section, many identities require the storage of cryptographic keys, and independent of the method used, the identity needs to be stored securely on the device. Therefore, researchers are looking for ways to solve this problem, from creating encrypted storage to dynamically generating cryptographic keys using the intrinsic characteristics of the device [25], [50].

In summary, identity in IoT has two ways to overcome this challenge, either by using existing IdM systems and making devices capable of handling computationally demanding cryptographic algorithms or by starting to employ new IdM systems that exploit lightweight cryptography. Either way, hardware can be part of the solution.

Any of the technologies presented in this work may be used to support these research challenges. The relationship between the two is further developed in the Section VI, where we compare the different technologies and analyze their advantages and disadvantages for each research challenge.

IV. Hardware-based IoT Identity Threat Analysis

As stated earlier in this paper, IoT devices are more likely to sustain physical attacks. This trend can be hampered by developing devices resilient to this type of attack, which means that the component and design of devices must be protected from physical attacks.

A threat analysis is a process to identify the security requirements for the component of each device. This analysis is based on various characteristics, such as potential intruders, attacks, and device assets.

Developing a detailed threat profile provides organizations with a clear illustration of the threats they face and allows them to implement a proactive incident management program that focuses on the threat component of risk [53]. This threat analysis starts with defining the context of the IoT environment that will be assessed and will follow a threat profile that includes information about critical assets, actors and threats to evaluate the requirements for secure hardware-based IoT identity management.

A. Context Environment

IoT devices have two major components: hardware and software. Physical attacks target the device's hardware. However, a hardware attack will affect the device's software due to the relationship between hardware and software.

As with any computer, IoT hardware has a CPU that provides computation capabilities, RAM to hold program storage, Read Only Memory (ROM) to store the boot program connected by a CPU bus, and multiple buses to hold peripherals, like persistent storage. However, unlike computers, IoT devices include all these components on a single chip called System On a Chip (SoC) [54], depending on the system's purpose. For example, it may have multiple WiFi and Bluetooth radios or general-purpose buses like I2C and SPI. Finally, but not least, an IoT device has its Printed Circuit Board (PCB), which provides pads to solder different components and offers a reliable electrical connection between them.

IoT devices need firmware to function. Firmware is an embedded-software in a piece of hardware. The first software to run on a device is the bootloader. The bootloader can be in the SoC, on an on-chip ROM - programmed during production - or in external memory. The bootloader is responsible for initializing the different components and transferring the execution to the user image. Depending on the device, the user image can be an embedded Operating System (OS) or a bare-metal application [55]. If the user image is an embedded OS, the operating system will manage the different processes, events, and a hardware abstraction layer. Besides, the embedded OS is also responsible for running the applications. On the other hand, if a bare-metal application is loaded, the application will need to handle and interact with the hardware directly. The choice between an operating system or a bare-metal application will depend on the device's capabilities.

Software and hardware depend on each other, and the device will not function correctly if one of these parts is faulty. If the software stack is compromised, it does not mean the hardware was compromised. Contrariwise, hardware attacks compromise software. Moreover, an existent Over-the-air (OTA) update can always patch software vulnerabilities. On the other hand, hardware vulnerabilities require a new hardware revision to solve them, so there will be devices that will never be fixed [56].

Hardware attacks require specialized knowledge and tools. Therefore, devices that do not require high assurance do not minimize these risks. Furthermore, security certifications that address these threats do not assess whether the risks are fully mitigated but assess whether the device has countermeasures to disrupt and delay the attacker [57].

Any device's components can be an entry point to compromise its identity. However, attacks on software will not directly affect the device's identity but rather the authentication protocol or will require lateral movement and further exploits to attack the device's identity.

As this work focuses on identity assurance through hardware, we will only consider identity threats related to hardware, which means hardware attacks, and attacks that can be mitigated by hardware. These threats will help us to define the security features and capabilities of the components mentioned in Section V. Moreover, during this analysis, we assume the attacker has complete access to the physical device and unlimited time to perform the attack.

B. Assets

In a threat analysis, an asset is something that has value to the company and must be protected. For example, in the case of a pay-TV network, there is a smart card with a decryption key that will control access to the network. In this case, this smart card is an asset as it is crucial to the company revenue [58].

Our threat analysis does not have a specific product, so we will only focus on the technical assets needed to achieve identity and authentication. Any of our assets will require confidentiality and integrity, which means the assets must be kept hidden from attackers, and attackers must not be able to modify them. We are not considering its availability because we assume that the attacker has physical access to the device, which means that for this type of attacker, making a service unavailable is the same as disconnecting it from power.

As we stated at the beginning of Section II, identifiers must support the identity of a IoT device. We have listed four identifiers: context, association, knowledge, and inheritance. Context is a technique that allows identifying a device by analyzing the way the device relates to other devices and the environment that is associated with it. Regarding association, it is to something that device can possess for one or two-factor authentication, such as a hardware token. By contrast, knowledge and inheritance rely solely on the device's characteristics to create an identity. The inheritance category uses hardware characteristics to identify a device and the knowledge category encompass the information that only the device knows, such as an authentication token or cryptographic key. Therefore, these last two identifiers are prone to be compromised by hardware attacks.

For the inheritance identifier, the attacker may try to clone the **Integrated Circuit (IC)** design to steal its identity. Therefore, the confidentiality of the **IC design** must be protected.

On the other hand, attackers may also try to tamper with the device to bypass security features, so the **IC design** must protect its integrity.

In addition to the **IC** design, an attacker needs to have access to the device's **firmware** to clone it. Furthermore, the attacker may reverse engineer the firmware to find software vulnerabilities that can compromise the authentication mechanisms. Therefore, keeping the firmware confidential will difficult the cloning process and the vulnerabilities discovery.

Regarding firmware integrity, the main security goal is to make it impossible for unauthorized code to run on the device. This can be developed into three attack vectors. The attacker must not be able to change firmware when stored on persistent storage. If persistent storage is compromised, the device must not execute any instructions that could have been tampered with. Finally, the attacker must not be able to induce temporary changes in the execution of instructions.

Finally, knowledge identifiers, depending on the device, can be authentication tokens, passwords, cryptographic keys, or other information. During this work, we will refer to them as **identity data**.

In the case of cryptographic keys, it is essential to highlight that even if the system does not use them directly as an identifier, they can be used internally to keep information secure or authenticate data traffic. Therefore, the keys must be protected regardless of their type. For public keys, the goal is to maintain their integrity; for private keys, it is to ensure their confidentiality. Public keys are used to verify signatures and encrypt information. Therefore, if a public key is changed, it can disrupt the signature verification process, leading to various attacks, from Man-In-The-Middle (MITM) to malicious updates of OTA. The private and asymmetric keys are used to identify the device or decrypt important information. Therefore, if an attacker compromises their confidentiality, they could spoof the device's identity or MITM communications.

To conclude, devices must maintain three secure assets: **CI design**, **firmware** and **identity data** to protect the device's identity and authentication capabilities.

C. Actors

Many techniques presented throughout this section require expensive equipment, in-depth system knowledge, and execution time. Therefore, not all are within reach of all threat actors. A threat actor is someone, either a person or a group, who poses a threat [59]. Threat actors can be adapted according to their capabilities and motivation. The capabilities of threat actors are the combination of resources and knowledge available to carry out an attack. The threat actor's motivation is the level of desire to perform some action and the expectations (confidence) of success [60]. The threat actor's motivation level is related to the time it takes to succeed. A less motivated attacker will quickly give up on the target if the device has too many countermeasures that require time to circumvent. IoT devices have five main threat actors: criminal enterprises, industrial competition, nation-states, ethical hackers, and lay attackers [56]. Criminal enterprises are motivated by the financial gain they can get from an attack. Typically, these organizations look for vulnerabilities that enable solid business cases. For example, Remote Code Execution (RCE) vulnerabilities in IoT devices to increase the ability of botnets to perform Distributed Denial of Service (DDoS) attacks. Due to the nature of these organizations, the budget for these attacks is high, but their criminal status limits the human resources they can hire and the equipment they can buy.

Industrial competitors aim to collect information about a competing device, which means they reverse engineer the device. This attacker will employ skilled professionals and have an ample budget to acquire all the resources needed to carry out an attack.

Nation-states have motivations such as espionage, counterterrorism, and sabotage. This group has all the necessary means to carry out complex attacks, with the best and most significant amount of professionals, tools, and time to plan attacks.

Lay attackers are usually individuals or small groups who hack to extort money from companies or gain a reputation. Typically, these groups have limited resources and expertise, so they will look for another target if they find countermeasures.

Ethical hackers are driven by a curiosity about how a system works or by the possibility of monetary reward. Unlike lay attackers, they will not try to make money illegally but through legal initiatives, such as bug bounties. They may have the expertise to carry out complex attacks, but they have a limited budget and time. At the same time, they can gain access to more expensive equipment through universities and hackerspaces. When compared to other attackers, they pose a different kind of risk. Ethical hackers are not trying to harm a company, but if not dealt with properly, they can affect consumers' trust in a company because if a vulnerability of a known company is disclosed before it is resolved or properly explained by the company, defamation can result, and customers may no longer trust the company. The motivation and capabilities of each attacker are summarized in Table I. In this table, the attacker's capability is analyzed using two characteristics, knowledge and resources, classified as low, moderate, and extensive.

The motivation was classified as low, high, and extreme, according to the perseverance level of each aggressor. These characteristics directly impact the attacks that each attacker can carry out.

Threat actors	Capability		Motivation
	Knowledge	Resources	
Criminal enterprise	Moderate	Moderate	High
Industry competition	Extensive	Extensive	High
Nation-states	Extensive	Extensive	Extreme
Ethical hackers	Moderate	Moderate	High
Layperson attackers	Low	Low	Low

TABLE I
SUMMARY OF THREAT ACTORS CHARACTERIZATION

D. Threats

Hardware attacks can be divided into two main categories, non-invasive and invasive attacks, according to the physical impact on the device. Non-invasive attacks do not require any preparation of the device to be performed, which means the attacker can access all required components without any modification of the device and does not leave any tamper evidence. By contrast, invasive attacks require the removal of the chip package and target the components inside. Such attacks require expensive tools, complex techniques, and operating at a miniature scale. Many times these attacks also imply the destruction of the chip.

In addition to these categories, some authors also proposed an intermediary class between the non-invasive and invasive attacks, called semi-invasive attacks [61], [62]. Semi-invasive attacks are a subset of invasive attacks that imply the removal of the chip package but do not require contact with its internal lines, decreasing its complexity. For the simplicity of this work, we will not distinguish these attacks from invasive attacks.

Another way to organize the attacks is in terms of similarities in the objective of the attack. This taxonomy originates three main classes of attack [62]: reverse engineering, fault injection, and side-channel attacks and eavesdropping, each containing multiple attack techniques (Figure 1).

In addition to describing possible threats, during this section, we will also present the countermeasures a device can take to difficult their exploitation. These countermeasures can either be in software or hardware. However, our main focus will be hardware countermeasures.

Reverse engineering

Reverse engineering is analyzing a fully functional system and developing a set of specifications describing the system [63]. In IoT, there are two susceptible targets to reverse engineering, the actual hardware (components and PCB) and its firmware. With these attacks, the attacker's goal is to totally understand the device's inner workings, find vulnerabilities, or clone the device.

The attacker employs the most invasive techniques to reverse engineer the device's hardware. Decapsulation, depackaging and delayering are processes that use chemicals to dissolve the chip package. With decapsulation, the attacker partially dissolves the package keeping the chip functional. Alternatively, depackaging completely removes the chip package and makes it non operational. Depacking enables the attacker to delayering the chip. PCBs and chips have different layers with electrical circuits, so by delayering, the attacker will polish off individual layers of the chip to better analyze them. These techniques are used to allow the reverse engineering of the device. Once a chip is exposed, an attacker can use high-resolution images or Scan Electron Microscope (SEM) to analyze it. Moreover, by applying delayering techniques, an attacker can analyze multiple chip layers [64].

Both decapsulation and SEM can be hampered by active metal shields [65] or a defensive PCB design pattern [66], [67]. An active metal shield is a metal conductive layer in the PCB that shields critical circuit elements. Depending on the shield, it can be a simple conductive layer or a meander of conductive lines with resistance sensors attached that will detect tampering, from physically probing to decapsulation attempts [65]–[68]. In terms of defensive PCB design, there are multiple techniques that can be employed. For instance, critical signals can be routed on deeper layers of the PCB and overlapped by other electrical paths that their destruction would disrupt the device's operation [66], [67].

These techniques can also be applied to reverse engineering embedded memory to extract stored information. The information stored on a masked ROM can be decoded, after decapsulation, with an optical microscope. Moreover, it is also possible to use techniques such as microprobing to monitor buses to extract information or even bypass encrypted buses by reverse engineering the chip design [69].

Micropobing is the act of attaching probes inside a chip to measure (side-channel attacks and eavesdropping) or inject voltage (voltage glitching) into an electrical line. This technique expects a decapsulated chip, which means it is an invasive technique and sometimes requires the creation of probe pads with Focused Ion Beam (FIB) [56]. FIB is a beam of ions that can either remove parts of a chip or deposit material. With this technique, an attacker can cut or reroute wires at a nanometer scale. As microprobing, FIB equipment is expensive and requires special knowledge to be performed. Despite its complexity, when successfully performed, it can circumvent many hardware security measures, such as active shields. FIB can be used to tamper with a chip or support other attacks [56], [70]. Micropobing will always depend on the success of the chip decapsulation. Moreover, a defensive PCB design that makes accessing critical signals difficult can hamper microprobing techniques.

For an attacker that is trying to reverse engineering firmware, he first needs to obtain it from the device. An attacker can leverage its access to the device's hardware and perform a PCB or logical attack.

The PCB interconnects the different components of the device. So, to extract the firmware, the attacker can connect directly to the persistent external storage with probes or desolder the memory and then use a debug tool to read it. If the attacker is able to read the firmware, he probably can write a tampered version of the firmware in the device. Another option

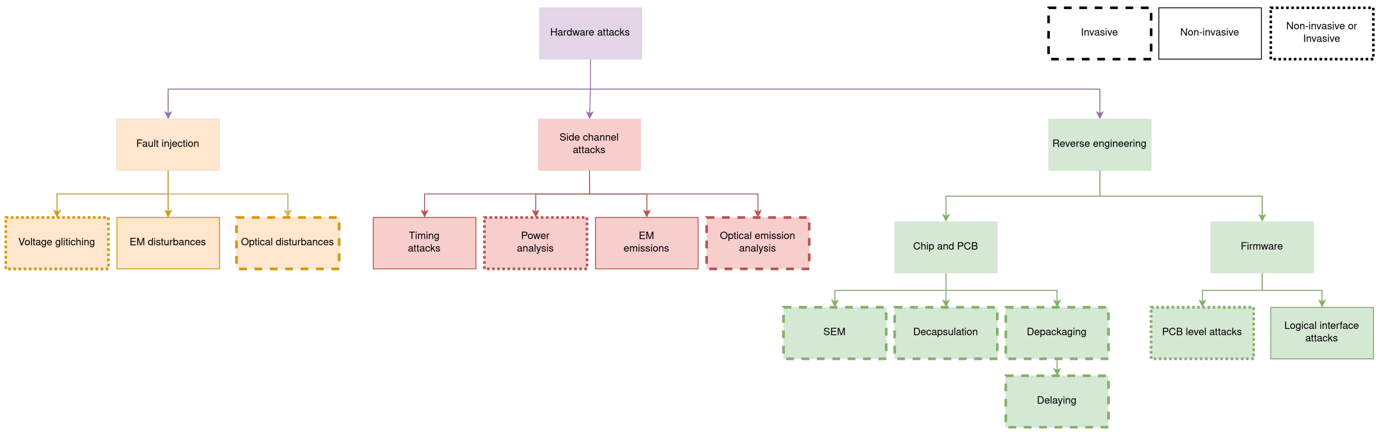

Fig. 1. Hardware attacks taxonomy

can be to analyze the different PCB connections and discover possible features that electrical connections can unlock. SoCs can often be booted with some security features disabled by grounding or pulling up logical pins.

Logical attacks exploit logical interfaces to communicate directly with the device's firmware, bypassing all hardware security features. Commonly, IoT devices have exposed logical ports like Joint Test Action Group (JTAG), Serial Wire Debug (SWD) or Universal asynchronous Receiver/Transmitter (UART), which allow direct interaction with the firmware or even attaching a debugger. These ports can be disabled in software, regardless, the majority of the times they are enabled. If an attacker can communicate successfully with the JTAG or SWD interface, it will be able to extract the firmware and interact with its execution. If it is UART, depending on the exposed software and device's settings, it may require further exploitation to obtain the device's firmware.

After having access to the device firmware, the reverse engineering process is similar to software. The process is the same, but researchers need to know the device architecture and how the firmware interacts with the hardware, which is often overlooked in other forms of reverse engineering due to the abstraction provided by operating systems.

Fault injection

Fault injection is an attack that induces processing errors in a processor, forcing the processor to jump the execution of an instruction or change the content of a register.

There are multiple ways to create these faults. The more common ones are voltage glitching and electromagnetic and optical disturbances (also called laser glitching). These attacks are non-invasive, with the exception of optical disturbances, which are invasive attacks. They all have the same objective; however, the attack vector changes.

Voltage glitching is when an attacker causes a fast change in a device's component to affect its operation. Usually, this attack is executed against a power supply or a clock signal. This attack does not imply any invasive process. It is carried outside the component's package without any physical modification. This kind of attack aims to push the hardware to induce an error in the software. A simple example of this type of attack is documented by Colin O'Flyn [71], which with a paper clip, performed a power glitch to a Philips Hue Bridge 2.0's Electrically Erasable Programmable Read-Only Memory (EEPROM) to interrupt the communication between the memory module and processor. With this fault, he was able to interact with the bootloader shell that is locked. However, the majority of these attacks are more complex. For instance, force the processor to jump the execution of an instruction.

Usually, voltage glitching is a non-invasive technique. However, it can also be performed at a nanometer scale by attacking voltage lines inside a chip (microprobing). As we stated before, this technique makes the attack invasive since it requires decapsulation of the chip.

Electromagnetic disturbances are when an attacker generates electromagnetic (EM) signals and directs them to cause faults. This is possible because changes in a magnetic field near a chip induce alterations in the voltage, which can temporarily cause flips in the logical levels of a data line.

Optical disturbances leverage the fact that when a transistor is illuminated with a photon intense light pulse, it conducts current, which can be used to generate localized faults. This attack requires decapsulation of the component and lasers to emit light pulses.

Fault injection attacks jeopardize the code integrity of the device by executing the code in an unintended way. These attacks are momentary and not persistent by nature but can be leveraged to produce persistent errors, for instance, by attacking the storage interface [72]. Overall, fault injection attacks require knowledgeable and highly motivated attackers since these attacks need to be tuned by experimentation according to the target hardware, which is time-consuming. Moreover, electromagnetic and optical disturbances involve high voltages and lasers, which can hurt the attacker if the necessary safety measures are not taken.

Generally, devices can prevent this type of attack by applying software or hardware countermeasures. In terms of software, there are multiple recommendations that developers may follow. For instance, random delays can be added to the code to deflect exploitation, or critical information may be checked multiple times during execution (for instance, two copies of the same information stored in different memory regions) to detect any exploitation attempt [73]. The duplication principle may also be applied to hardware. For example, the device can have the same function implemented in multiple places and compare their output to detect tampering attempts. Nevertheless, this type of approach is expensive [68].

Besides these general countermeasures, there are also specific techniques for each threat. Devices may have voltage sensors or monitor the clock signal to detect voltage glitching attacks [74], [75]. Power or clock lines are scattered throughout the PCB. Therefore, any attack attempt will propagate over these lines, making the attack easily detectable. On the other hand electromagnetic attacks are localized, which makes them difficult to detect [76]. EM sensors may be used to detect electromagnetic fault injections [75], [77]. However, their placement must be properly analyzed due to their limited range. On the other hand, electromagnetic fault injections can be avoided by using an active metal shield that protects the SoC from electromagnetic waves and optical disturbances [68], [78].

Side-channel attacks

Side-channel attacks are non-invasive attacks that aim to extract secrets from a system by measuring and analyzing physical parameters like time, power or electromagnetic emissions [79]. The majority of these attacks require the plaintext and correspondent ciphertext to be known by the attackers.

Timing attacks exploit data-dependent execution time differences to uncover secret data [79]. For example, in a password check, if a system compares the password inserted by an attacker with the correct password character by character, an attacker could measure the time it takes from the password being inserted to the feedback received. With these measurements, an attacker could reduce the effort of a brute-force attack by brute-forcing each password character thanks to the different execution times depending on each number of correct characters in his input. Depending on the target, this attack could be performed by timing responses on its software interface [80] or measuring CPU cycles [81]. The developer must mitigate this threat at the software level by ensuring the same response time independently of the input correctness.

The power consumption of a processor depends on its current activity, mainly when there are changes in the state of its components. A precise measurement of the power consumption allows an attacker to identify the current instruction and estimate changes of bits in memory [61]. Many power analysis techniques can be used to attack cryptographic systems. However, the two primary techniques are Simple Power Analysis (SPA) and Differential Power Analysis (DPA) [61].

SPA relies on the direct observation of power consumption and requires the attacker's specific knowledge about the cryptographic algorithm implementation to succeed. DPA is a technique that does not require this previous knowledge. It leverages statistical analysis to extract information from a data set of power traces [82]. Despite not requiring too expensive equipment, power analysis requires a skilled attacker.

An example of an attack using this technique was when a group of researchers extracted the cryptographic keys used to encrypt and verify firmware updates of a smart bulb through power analysis, which enabled attackers to upload a malicious OTA update [83].

To prevent these attacks, boards may use voltage regulators to keep their power consumption steady independently of the operations that are running. Nevertheless, attackers may bypass this by probing inside the chip (microprobing) [75]. This process would require decapsulation.

Integrated circuits in operation emit electromagnetic waves. The principle behind electromagnetic analysis is very similar to power analysis. It is possible to identify events by analyzing the electromagnetic signals around a device with electromagnetic probes. Furthermore, in the case of electromagnetic analysis, we can also identify the location of a specific activity by locating the source of that radiation, which is not possible in power analysis. Finally, similar to power analysis, there are multiple techniques to analyze these measures. The main ones are Simple Electromagnetic Analysis and Differential Electromagnetic Analysis, which are similar to their power analysis counterparts [79].

Optical emission analysis studies emitted photons by transistors that change state. Once again, there are two main techniques, simple and differential analysis [84], which can be used to retrieve cryptographic keys. Additionally, optical emission analysis allows attackers to locate the emission source of this photon, which means it can support reverse engineering efforts. Finally, these attacks require direct observation of the different components of the chip. Therefore the chip needs to be decapsulated. Moreover, these attacks require custom build tools to be performed, increasing the knowledge required to execute this kind of attack [56], [85].

A possible mitigation for electromagnetic and optical emission analysis is to use an active metal shield to protect critical components. In the previous subsection, we explained that these shields hinder EM and optical injections. Additionally, this type of shield prevents emissions generated inside the package from propagating to the outside, thus, preventing leakages [68], [86].

Similar to injections, some software measures, such as random delays during execution, can be implemented to decrease the risk of successful side-channel attacks [73].

Side-channel attacks mainly affect the confidentiality of identity data. The actors that perform these attacks have extensive knowledge of statistical analysis and the target's cryptographic implementations. Regarding resources, both time, power, and electromagnetic emission analysis require the same equipment, an oscilloscope, and the necessary probes. By contrast, optical emission analysis requires specialized equipment and decapsulation of the chip.

Eavesdropping

Depending on the device's design, an attacker could eavesdrop on information from the buses connecting the different device's components. This technique could be invasive or non-invasive, depending on the attack scale and if the buses are exposed or not. In a non-invasive form, this attack only requires a logic analyzer, which is not expensive depending on the number of probes. If an invasive approach is required, an attacker must use microprobing [87]. Either way, the attacker needs to be able to reverse engineering the signals and convert them to meaningful information.

The simpler forms of this attack can be prevented by a defensive PCB design, which means critical signals should not be routed on the top or bottom layer of the PCB. This way, the attacker is forced to be invasive and perform more complex techniques such as decapsulation and microprobing to reach the same goal, eavesdropping on a signal. Nevertheless, if the signal is critical to the device, the designer should evaluate its encryption [67].

E. Summary

Multiple attacks based on hardware can compromise the device's identity and authentication capabilities. Throughout this section, we analyzed the different security objectives to keep the IoT identity and authentication secure. At the same time, we listed the attacks that may be used to compromise these devices and detailed their requirements regarding knowledge and resources. These facts have a direct repercussion in which attack a specific threat actor may employ.

Table II summarizes the different attacks by mapping the requirements needed by the attacker to be successfully. Also, it identifies the compromised assets and outlines the hardware countermeasures. In this table, we included microprobing and FIB as they can be applied to multiple attacks and will impact its requirements.

During Section V, we will use the countermeasures identified in this section to assess the level of security of each technology against hardware attacks.

V. Hardware Trust Anchors Technologies

We have identified six technologies that can be used as building blocks to overcome today's open research challenges IoT, True Random Number Generator (TRNG)s, ROMs, crypto accelerators, secure elements, TEEs, and PUFs. For each technology, we look at its advantages and disadvantages for supporting device identity, the existing security attacks and countermeasures that are typically implemented, and finally, the systems where they have already been used to support device identity.

Before analyzing each one, we must reflect on their nature and how they relate. We can differentiate two distinct groups in these technologies: basic and composite building blocks.

Base blocks provide elementary resources, and composite blocks provide multiple resources, which means they can be broken down into smaller building blocks. For example, a TRNG is a base block as it only provides random numbers. On the other hand, a secure element is a composite building block because it provides various features such as cryptographic operations and random number generation, which are provided by base blocks, namely a cryptographic coprocessor and TRNG.

A. True Random Number Generator

Encryption is the foundation of identity and authentication solutions. These systems rely on unpredictable and unreproducible key streams to generate cryptographic keys and produce authentication challenges, among other things. There are two random number generators types: True Random Number Generator (TRNG) and Pseudorandom Number Gnerator (PRNG). However, for cryptographic operations, PRNGs are not recommended because of their lack of entropy [88].

TRNG is a random number generator that can generate random numbers, without any periodicity, from physical sources. The TRNGs are distinguished from the PRNGs, used in most systems, by the quality of the generated numbers. A PRNG uses algorithms to generate a sequence of numbers that depends on the initial seed given to the algorithm. The numerical sequence of a PRNG is deterministic, which means that an attacker can calculate the PRNG sequence if the seed is known. Therefore, the seed must be random.

On IoT devices, seed generation have limited entropy sources, and attackers can have physical access to the device, allowing them to disrupt these sources. Therefore, TRNG can be used to overcome these problems by extracting entropy from the device environment, such as electronic noise [19]. Examples of these electronic noises are the variations of signals generated by electronic oscillators, which can be sampled, filtered for possible interference and quantified as digital bits [89]. However, this construction requires several oscillators to produce a high quality number [88], which increases production cost and energy consumption [19].

Electronic noise can suffer external interference, affecting the quality of the generated numbers. Therefore, researchers have been using quantum theory to support new TRNG constructs to solve these problems. In quantum mechanics, each choice is random and independent of the other. Based on this, researchers have been using light pulses and analyzing each photon's choice or using the time between an element's radioactive decay to create TRNGs [90].

1) Security attacks and countermeasures:

TRNGs, as a component, is normally embedded in a device. Therefore, it does not include security countermeasures and delegates them to the device. Depending on the type of TRNG, some attacks leverage the environment bias of these components to generate weak numbers. For instance, Ring Oscilator (RO)s based TRNGs can be biased with EM fault injections [90], at the same time, attackers can perform EM side-channel analysis to retrieve information about its internal state [91].

2) Advantages and disadvantages for identity assurance:

IoT suffers from a lack of available entropy. Therefore TRNGs can be the solution to provide high entropy numbers in an IoT platform.

On the other hand, TRNGs have several disadvantages. They increase the cost of the device and its power consumption. Moreover, common TRNGs that do not use quantum physics are susceptible to environmental biases, which means threat actors with physical access to the device can take advantage of this fact and attack the generated numbers.

3) Implementations:

TRNGs are included as a base component of other more complex. For instance, any identity system that uses a Trusted Platform Module (TPM) or Secure Element (SE) will inher-

Threats	Identity assets						Required capabilities			Hardware Countermeasure
	Identity data		Firmware		IC design		Knowledge	Resources	Motivation	
	Confidentiality	Integrity	Confidentiality	Integrity	Confidentiality	Integrity				
Voltage glitching	No	No	No	Yes	No	No	Moderate	Low	High	- Voltage sensors - Clock signal sensors
EM disturbances	No	No	No	Yes	No	No	Moderate	Moderate	High	- Multiple voltage sensors - Active metal shield - EM sensor
Optical disturbances	No	No	No	Yes	No	No	Extensive	Extensive	High	- Active metal shield
Timing attacks	No	Yes	No	Yes	No	No	Moderate	Low	High	
Power analysis	Yes	No	No	No	No	No	Moderate	Low	High	- Voltage monitoring
EM emissions analysis	Yes	No	No	No	No	No	Moderate	Low	High	- Active metal shield
Optical emission analysis	Yes	No	No	No	No	No	Extensive	Extensive	High	- Active metal shield
SEM	No	No	No	No	Yes	No	Extensive	Extensive	High	- Active metal shield - Defensive PCB design
Decapsulation	No	No	No	No	Yes	Yes	Extensive	Extensive	High	- Active metal shield - Defensive PCB design
Depackaging	No	No	No	No	Yes	Yes	Extensive	Extensive	High	- Defensive PCB design
Delaying	No	No	No	No	Yes	Yes	Extensive	Extensive	High	- Defensive PCB design
PCB level attacks	Yes	Yes	Yes	Yes	No	No	Low	Low	Low	
Logical interface attacks	Yes	Yes	Yes	Yes	No	No	Low	Low	Low	
Eavesdropping	Yes	Yes	Yes	Yes	No	No	Moderate	Low	Low	- Defensive PCB design
Microprobing	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	Extensive	Extensive	High	- Defensive PCB design
FIB	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	Extensive	Extensive	High	

TABLE II
RELATION BETWEEN THREATS, ASSETS AND REQUIRED CAPABILITIES

ently use a TRNG since TRNGs are fundamental part of these components.

An example of the explicitly use of TRNGs in identity systems is the work of Yang Su et al., in which a decentralized machine identifier for electric vehicles was developed employing a TRNG module for the generation of the vehicle identification [92].

B. *Masked ROM and OTP memories*

Non-Volatile Memory (NVM) is a type of memory used in devices to store information persistently. There are multiple families of NVM which differentiate themselves by the employed technology and the number of times they can be rewritten. Among the different types, the ones that only allow a single write operation may be used in a system to support security operations. For instance, it can be used to implement a Root-Of-Trust (RoT) or store public keys [93].

There are four types of one-time writable memories, masked ROMs, floating-gate One-Time-Programmable (OTP)s, fuses OTPs, and anti-fuses OTPs. These last three are called OTP memories because they can be programmed in the field, offering more flexibility than a masked ROM, which can only be set during fabrication [94].

Masked ROMs have the information hardwired in the chip design, which means the data to store needs to be known before the production of the component since its electrical connections are rearranged to represent the information that needs to be stored. The significant advantage of this type of memory is its low production cost when a large amount of memories storing the same information is required. At the same time this is a disadvantage for small deployments [93], [95].

Floating-gate memory is a type of memory that leverages floating-gate transistors to store information, allowing them to be reprogrammed and erasable in the field [96].

Floating-gate memories can be erased by exposing them to a source of Ultraviolet (UV) radiation. This is possible since the packages of these memories have a quartz window that can be uncovered to irradiate the floating-gates [93]. Floating-gate OTPs follow the same working principle of floating-gate memories but have the floating-gates shielded to ensure they are not reset by radiation.

Eletric-Fuse (eFuse) OTP memory is constructed with a set of fuses that are blown to represent data. This operation is done by applying a high voltage to the fuse, which can be done in the field [97]. The working principle behind an eFuse is electromigration, the process by which material is gradually transported in a conductor. eFuses are constituted by conductive metal lines that, when exposed to high voltages, resistance increases and makes the circuit open due to electromigration [98]. However, in terms of data retention, electromigration introduces disadvantages. eFuses are susceptible to re-growth issues, where metal lines unintentionally connect, changing the stored data [99].

Anti-fuse OTP memory is similar to the Fuse OTP but uses an anti-fuse on its construction. An anti-fuse is an electronic component that, when unaltered, does not conduct electrical current, but after being exposed to a high voltage, it becomes conductive. This change is used to store information permanently in memory [97].

1) *Security attacks and countermeasures:*

The main objective of read-only memories is to make data unchangeable. With the exception of floating gate OTPs, there are no known attacks to this part. As we stated before, since floating gate OTPs are based on floating-gate memories, attackers may try to bypass its shield and expose it to UV radiation to be able to write it again [100]. eFuses suffer from the re-growth issues, but we have not found any literature exploiting this fact to create an attack to compromise the OTP.

On the other hand, since these memories have information that can be interesting to an attacker, it is essential to assess how easy it is to retrieve the stored information. The presented memories have very similar protection levels. To retrieve the information, an attacker must depackage and delaying the chip and then use an high-resolution optical microscope to read its information manually [69], [100]–[102]. Another option is to use an SEM to retrieve the information. However, this is not possible on anti-fuse OTPs [98]. All these techniques are invasive, requiring expensive equipment and manual work to retrieve the memory data.

To mitigate these risks, device designers employ active shields protecting the memories that will erase their content or damage the device if any tampering attempt is detected. Nevertheless, if not backed up by batteries, this kind of protection will only work if the device is power-on. Therefore it will not stop offline attacks.

2) Advantages and disadvantages for identity assurance:

Masked ROM and OTP memories are cheap options to store information that can only be read. Despite that, they also have several security risks which cannot be ignored for high assurance deployments. An attacker with physical access to the device can replace these with similar ones but with different content since devices do not have mechanisms to ensure the integrity of the memory itself. Moreover, for each memory type, specific attacks can be leveraged to change the content of these memories [101], [103]. Therefore, masked ROM and OTP memories bring advantages for systems that require a low-security assurance level. However, if these components are used for other assurance levels, their security risks need to be mitigated.

3) Implementations:

Masked ROM and OTP memories are used as building pieces for more complex systems. The most common use of these memories is to provide a RoT, which is why they are included in most SEs (Subsection V-D) [104]. More recently, researchers have been using these components as RoT in the Arm Trust Zone system (Subsection V-E) as they do not provide a default secure way to store information [105]. Another use for these memories is to enable and disable features in a device. For instance, many SoCs use fuses to disable debug capabilities, such as a JTAG or UART port [17].

C. Crypto accelerators

Crypto accelerators offer a high throughput in cryptographic operations. These components may or may not include countermeasures against known attacks and are mentioned in the literature with multiple names, such as a custom processor and crypto array [106]. We will follow the Bossuet et al. [106] taxonomy and present four types of components: General Purpose Processor (GPP)s with crypto acceleration, hardware crypto coprocessors, crypto processors, and crypto arrays.

GPPs with crypto acceleration are Central Processing Unit (CPU)s that have dedicated instructions for cryptographic operations. These specialized instruction sets allow programs to take advantage of dedicated hardware for cryptographic operations, which is faster than running these operations on a general-purpose hardware. However, this type of solution does not offer any other feature, which means it depends on the GPP for secure storage and preventing hardware attacks.

The crypto acceleration in these GPPs is usually implemented using specialized Arithmetic Logic Units inside the GPP to provide a low overhead connection to the GPP using internal buses.

Examples of GPPs with crypto acceleration are the Intel CPUs with the AES-NI instruction set [107] and ARM CPUs with the ARMv8 Cryptography Extension [108].

Hardware crypto coprocessors are a logic devices or hardware modules dedicated to executing cryptographic operations. These coprocessors cannot be programmed are entirely dependent on a processor to operate, to the point of not having any storage to store secrets. However, these components are more flexible than GPPs with crypto acceleration, enabling reconfiguration of the algorithms, since they are typically implemented on top of Field-Programmable Gate Array (FPGA)s [106].

A crypto processor is an independent processor that is specialized in cryptographic operations. In contrast to GPPs, these processors protect their secret keys. The keys are typically generated inside the processor, stored in a dedicated memory, and transported in a dedicated bus. All these measures are implemented to ensure that the system can only interact with the key by performing a cipher or decipher operation [106], [109]. Nevertheless, countermeasures against more intrusive attacks will vary according to the mode.

An example of a crypto processor that can be found in the majority of the computers is the TPM.

Hardware TPMs are secure crypto processors that implement a specification that the Trusted Computing Group (TCG) created to establish trust in a computation system. Namely, hardware TPMs must have the appropriate hardware protections to provide three RoTs, storage, measurement, and reporting [110], [111].

TPMs have registers to store measurements of each software that runs during the boot process of a system. These measurements provide a chain of trust, allowing detection of any tampering attempt of the boot process. In addition to that, they have a set of asymmetric key pairs which can be used for encryption and signing [110]

These two features allow the production of signed reports of the system's software configuration and, at the same time, decipher data only when the system matches a specific state, which is the base for Trusted Computing.

TPMs offer a secure random number generator and cryptographic engines according to their version. For instance, the most recent version, 2.0, offers RSA, ECC and AES cryptography engines. All these features are backed up by tampering-resistant hardware. [112]

Finally, a crypto array is a crypto accelerator where we have multiple cryptographic processing elements that work together with a GPP to provide a fast parallel computation of cryptographic algorithms. The primary use of these components is VPNs, which require handling multiple ciphered connections simultaneously [106]. Therefore, they are neither directed nor used in IoT since they do not need to handle multiple connections simultaneously.

1) Security attacks and countermeasures:

Generally, crypto accelerators do not offer security countermeasures to prevent hardware attacks, since they are included in more complex systems. Thus, this group of components has been targeted by multiple attacks. For example, it is known that Intel AES-NI is vulnerable to voltage glitching [113] and side-channel attacks [114]. An example of an attack against crypto coprocessors is the power fault injection attack against the coprocessor presented in the PlayStation Vita [115].

Regardless, some of these components may offer security features. For instance, usually, TPMs use the same processors of SEs. Therefore they inherit features like active metal shields and voltage monitors.

Despite this, it is important to remember that the level of protection of TPMs will vary. By definition of the TCG, TPMs need to be certified with Common Criteria Evaluation Level 4, which means the component is methodically designed, tested, and reviewed against the TCG security profile [116]. The TCG security profile defines that TPMs must be protected against physical hacking attempts. In addition to the Common Criteria, TPMs are certified with the FIPS 140-3 certification [117]. This certification has multiple assurance levels. Level 1 of this certification does not require any specific physical security mechanism and level 2 requires that the component shows evidence of any tampering attempt on the plaintext cryptographic keys and critical security parameters. Mechanisms to detect and respond to physical attacks are only enforced in level 3 of the certification [117]. Typically, TPMs are certified with level 1 or 2, which means that TPM designers do not need to have protections against invasive attacks but instead need to make the component tamper evident.

2) Advantages and disadvantages for identity assurance:

One of the limitations of implementing security features in IoT devices is that we have a limited development budget and minimal hardware in terms of computation capabilities. Therefore, crypto accelerators can be the solution when devices do not have the capability to run cryptographic algorithms.

GPPs with crypto acceleration are the cheapest and easiest way to include crypto acceleration in an IoT device since it is not a separated component but instead embedded in the device's main CPU of the device. To benefit from these capabilities, developers only need to ensure that their operating system and cryptographic libraries use these specialized instruction sets [17]. Thus, making it easy to integrate at the software level with the rest of the system.

Another way, but requiring more adaptation, is to use crypto processors. These components are independent components that must be added to the device and connected to a general-purpose bus. At the software level, these components typically provide a software stack that can be used to leverage their capabilities.

Finally, cryptographic coprocessors are the one that brings more disadvantages. It requires low-level integration with access to GPP's internal buses. However, it has the advantage of being reconfigurable if needed.

3) Implementations:

GPPs with crypto acceleration are common in commercial processors. As we stated before, AES-NI and ARMv8 Cryptography Extension are examples of this. Thus, there are many IoT identity systems that, intended or not, already leverage this extension to accelerate their operations.

Crypto-coprocessors and crypto processors are used to accelerate cryptographic operations when limited controllers are used. For example, Pearson et al. used a microchip cryptographic coprocessor to accelerate authentication and encryption operations with the cloud. As a bonus, this coprocessor also offers storage for cryptographic keys, which was used to store the authentication keys [118]. Another example is the use of TPMs. Due to its widespread, multiple solutions include this technology as a way to attest its boot and store its device's cryptographic keys [119]–[121]

D. Secure elements

A Secure Element (SE) is a tamper-resistant component that offers a set of security primitives, such as managing secrets or running applications securely [122]. This type of component cannot be easily forged or copied and has a unique identifier [123].

The concept of SEs was introduced by the GlobalPlatform, an initiative of different industry stakeholders to create specifications and standardization for secure components [122]. These components are also commonly called smart cards. In this work, we will use these two nomenclatures interchangeably.

There are three main form factors for SEs: Universal Integrated Circuit Card (UICC), which can be found in credit cards and sim cards, in a microSD form factor, and an embedded chip in the device, the so-called embedded Secure Element (eSE) [124].

A SE is a SoC with an independent CPU, RAM, EEPROM, and ROM in a small form factor. For reference, inexpensive smart cards have 12 to 144 kilobytes of EEPROM storage, 6 kilobytes of RAM, and 200 kilobytes of ROM [125]. Moreover, current smart cards have different interfaces to interact with the world. Initially, they used serial communication, but nowadays, with the proliferation of smartphones is common to find smart cards with Near-Field Communication(NFC) or even Bluetooth interfaces [126].

However, SEs differ from common embedded systems as they have multiple layers of defense. For instance. an attacker trying to decapping the SE's package would find an active current-carrying layer, that in the case of being break would destroy the information carried by the card [104]. To mitigate probing attacks, SE employs ciphered bused between the different components, and many times the, PCB paths are scrambled to difficult any attempt of reverse engineering [127].

SEs are susceptible to Side Channel and Fault Injection attacks. The preferred attack vector is power analysis, for side channel attacks in SEs is power analysis [128].

Usually, SEs mitigate these attacks in software, designing cryptographic algorithms that have constant execution time or introducing random delays in execution (for instance, in a stream of bytes when XORed with a key, the operation is not performed in a sequential order but in a random order) [104]. On the other hand, fault injection attacks can be mitigated by detecting or hampering the attacks. SEs employ sensors to detect fault conditions, like unusual events in the voltage or clock supplied to the card. In addition to these, they can also implement some software measures. Checksums prevent induced changes in memory. Random delays can hinder attacks, and execution or variable redundancy, where we have multiple copies of the same information in multiple places, can be used to detect tamper attempts. Each SE's manufacturer employs the measures he considers necessary to achieve the required assurance level.

In the past, as SEs were mainly used for a single application, the application and operating system were developed together and stored in ROM. However, this approach makes the development difficult since developers need specific knowledge about the smart card intrinsics. At the same time, the final product would be dependent on the specific smart card model and could not receive updates because it was stored on ROM. Nowadays, smart cards deploy the operating system and the applications independently to overcome these limitations. Smart card operating systems are minimal, provide hardware abstraction to applications, and are typically deployed in ROM, which means they cannot be changed after production. On the other hand, applications use the Application Programming Interface (API)s provided by the operating system and are stored in Electrically-Erasable Programmable Read-Only Memory (EERPOM), which means they can be changed over time. Moreover, many smart card operating systems support virtualization, enabling the deployment of multiple applications in the same smart card independently [129].

The GlobalPlatform standards have contributed significantly to creating multi-application smart cards, promoting security and inter-operability independently of the operating system. From the wide range of standards, it is important to highlight the GlobalPlatform Card Specification. This specification defines a set of logical components that enable secure multi-application smart cards and the different procedures and APIs to manage and install applications in the card. Moreover, the GlobalPlatform Card Specification details how we can manage the communication from the outside world to a specific application in multi-application environments [129].

Smart card models offer a set of security primitives backed up by specialized hardware coprocessors. Normally, this list includes asymmetric and symmetric cryptographic algorithms, hash functions, and a true random number generator. The list of available algorithms will vary depending on the smart card model [104].

1) Security attacks and countermeasures: Even though SEs have limited computing power and memory capacity, their security requirements are high. SE's threat model expects the information stored inside the card is kept secure even if the attacker has unlimited access. Thus, smart cards employ all the countermeasures we presented in Subsection IV-D.

SEs use multiple anomaly sensors to detect unintended execution conditions such as temperature, voltage, clock, and electromagnetic variations. If an unusual condition is detected, the SE will react to it. Depending on the devices and the sensor in question, the device can automatically reset itself or halt its execution until the working condition are regularized [104].

Over the years, multiple side-channel attacks have been registered against smart cards. Because of this, SE's software is designed with multiple lines of defense to prevent information leakages, such as constant execution time for cryptographic operations, memory masking when critical information is present in memory, and a randomize manipulation of data given by the user. In addition to software countermeasures, SE also have active metal shields that hinder some side-channel attacks, as explained in Subsection IV-D [104].

In addition to these measures, smart cards are designed to discourage reverse engineering. Critical parts of the SE internal design are randomized, and the active metal shield also hamper invasive attacks or SEM [104].

The security features of SEs are guaranteed by two certifications, the Common Criteria and FIPS 140-3 [117]. Normally, SEs are certified with the assurance level 6 of the Common Criteria, which implies a semiformally verified and tested design with a security profile created for this purpose [130], and level 3 of the FIPS 140-3, which enforces tampering protection and response to attacks [117].

2) Advantages and disadvantages for identity assurance:

Including SEs in IoT devices enables very constrained devices to overcome their hardware limitations to perform security operations. SE offers high-security assurance with low energy consumption. Moreover, depending on the application, there are different types of smart cards. In a more complex setup, a device can leverage multiple application cards, to have various applications inside the SE. In simpler systems, smart cards that only offer a limited set of features can be used.

No matter the choice, smart cards are not expensive. Multiple application cards are sold for two dollars at the time of this writing.

On the other hand, depending on the system design, the device may need to store a PIN to unlock smart card functionalities. Therefore, in a production system, using a SE will require other security mechanisms to solve this problem. Moreover, even with different form factors, adding a SE means adding another component, increasing the device's size and complexity. Finally, if a SE application needs to be created, the development team will need to get familiarized with a new technology and development kit.

3) Implementations:

SEs have been used by multiple researchers and solutions to securely store a cryptographic key that identifies devices and offload cryptographic operations [131]–[133]. For instance, in academia, Jeon et al. [132] proposed SEs in LoRaWAN nodes to prevent leaks of communication keys used in the LoRaWAN protocol. At the same time, in the industry, for example, Bosch security cameras use SE to store cryptographic keys and handle updates securely [133].

Other researcher have leveraged the fact that some SEs have Near Field Communication (NFC) interfaces to develop solutions that support remote identification and local identification [134], [135]. An operator can physically identify the device using a smartphone with NFC, mitigating problems related to labels that could be tampered with or even eavesdropped on by an attacker [135].

The use of SEs in military Unmanned Aerial Vehicle (UAV)s has also been studied. Any information stored in a military UAV must be kept secure, even if the enemy captures the UAV. With the current development of autonomous UAV fleets, researchers have proposed the inclusion of a SE in each drone from the fleet and use it to store any information that would compromise its mission or the rest of the fleet [136].

E. Trusted Execution Environment

A Trusted Execution Environment (TEE) is a set of software and hardware that provides isolated execution and storage environments from the main operating system. Its primary objective is to assure information security and privacy even if the device is compromised [137].

A fundamental concept in TEEs is the Trusted Computing Base (TCB), which is the set of software and hardware components that are explicitly trusted to ensure the security properties expected from a TEE [138], which means the TCB is the RoT of these platforms. The TCB defines two environments, TEE and Rich Execution Environment (REE). The TEE is the execution environment provided by the TCB, and REE is the execution environment provided by the untrusted components [123].

There are five security features that TEEs must implement: isolated execution, secure storage, remote attestation, secure provisioning, and trusted path [138].

Isolated execution allows applications to run in complete isolation from other code, which means they have their own address space and system resources. This can be implemented in multiple ways, from isolation at the OS, using a hypervisor or a parallel environment with separated components.

Secure storage provides confidentiality and integrity to the data, even when the device is powered off. Once again, depending on the assurance level required, the isolation can be assured by the OS. However if the OS is compromised, this isolation is not guaranteed. Therefore, strong constructions of secure storage use a separate component that ensures access control independently of the OS. Due to the small amount of storage available in the RoT, it usually stores cryptographic keys that are then used to decipher the rest of the data. With this, as the data is encrypted, it can be stored in an untrusted storage. Nevertheless, with this kind of solution, it is critical to prevent the rollback of data to a previous version.

Remote attestation allows the remote verification of the message origin. At the same time, it attests that the TEE loaded correctly. This means that remote attestation will only attest if the TEE's firmware is correctly loaded. Hence protecting the device against persistent treats but does not defend against a run-time compromise or does not inform if the device is not working properly.

Secure provisioning is the capability of sending data to a specific TEE, maintaining secrecy and integrity in the communication. This mechanism is normally used to offer secure updates or change settings in the device, leveraging remote attestation and cryptographic keys unique to each device.

Finally, trusted path enables secure access to physical peripherals. For instance, if an application running inside a TEE requires access to a keyboard for user interaction, it must be impossible to interfere in the connection between the TEE and keyboard in any way, including any attempt of eavesdropping the connection.

Similar to SEs, the GlobalPlatform initiative has a crucial role in establishing TEE standards. They propose TEE's architectures, define APIs to communicate from REE's applications with applications running inside the TEE and promote the development of TEE applications that can run independently of the underlying TEE implementation [123]. Finally, they also introduced the concept of Trusted User Interface API. As we mentioned before,TEE's applications often require user input, and that is why trusted paths are included in the architecture of a TEE. The GlobalPlatform tries to evade this problem by promoting an input/output peripheral inside the TCB [139].

GlobalPlatform TEE System Architecture specification [137] proposes three different architectures: a scheme with shared memory with the REE and an isolated component inside the SoC where the TEE operates; an architecture where all resources are shared with the REE but there is an isolation level between the two environments; an architecture where an External Security SoC is introduced in the device and communicates with the main SoC to provide the TEE capabilities.

1) Hardware-based TEE enabling technologies:

TEE is a general concept. Thus, there are multiple implementations, each one with its caveats and approaches. This subsection analyzes two TEE-enabling technologies, ARM TrustZone [140] and Intel Software Guard Extensions (SGX) [141], which were chosen due to their availability in the market and openness to third-party development.

However, it is essential to highlight that at the time of writing, the version of the Intel SGX detailed here is deprecated in consumer-grade CPUs. Only server-grade CPUs will continue to support it [142]. Moreover, there are rumours of a new version of Intel SGX, but details are scarce, and its future is uncertain [143].

ARM TrustZone

ARM TrustZone [140] is a set of hardware security extensions in a wide range of Arm processors, from the cheaper and less capable processors to the expensive ones. This technology allows an application to run either in a secure state (TEE) or a non-secure state(REE). The processor executes exclusively in one of these states at a given time. The underlying system assures a secure context switch between the two states and controls access to its resources. This construction does not have separate hardware for each environment. Nevertheless, the secure monitor guarantees its separation at the hardware level.

The secure monitor is a component that manages the context switch between the two worlds, REE and TEE. Depending on the processor generation, the secure monitor can be an independent component inside the processor or implemented directly on the processor logic [144].

Even though ARM TrustZone is intended as a security technology, it can also work as a virtualization technology supported by hardware [144]. This means that each execution environment may have its OS. This flexibility brings advantages for the developer. Depending on the application, the developer may create a software library that resides in the TEE and is called from the REE or use a Secure OS for the TEE [123]. In these cases, the TEE's OS is specially designed for this purpose, having a reduced set of features to keep the TCB as small as possible. On the other hand, the REE uses a common OS.

Developers typically do not implement applications directly on ARM TrustZone but instead use a TEE that leverages the ARM TrustZone, and acts as a development framework for their application. The GlobalPlatform standards boost this process since ARM TrustZone implements multiple GlobalPlatform standards, which enable interoperability among different TEEs [145]. Examples of TEEs used with ARM TrustZone are the OP-TEE [146], SierraTEE and Open-TEE [147].

Nevertheless, sharing the same hardware components bring risks. REE applications can try to interfere with TEE applications. Applications may try interfering with the TEE execution by creating interrupts to force context switches, creating a denial-of-service (DoS) or performing side-channel attacks to the shared CPU cache.

To mitigate DoS attacks, ARM TrustZone allows the configuration of interrupt prioritization to rank first the secure world's interrupts. Unfortunately, the developer must activate this feature specifically [144].

ARM TrustZone CPUs share their cache between applications from both worlds, which means both compete for the use of cache. Even though a non-secure world application cannot access to a cache assigned to a secure-world application due to a tag bit that signs to which world the cache is assigned. Regardless, this can yield different attacks, from a rootkit that evades introspection [148], to multiple side channel attacks that by monitoring cache activity are able to retrieve cryptographic secrets from the rich world [149]–[151].

Arm TrustZone secure world has full access to the memory of the untrusted world. This fact introduced a new class of attacks called BOOMERANG attacks [152]. These vulnerabilities enable an application, from the non-secure world, to exploit a TEE application to access a portion of memory which it does not have access.

Finally, ARM TrustZone neither specifies a RoT for their TEEs not a secure storage method. Therefore, the system designer is responsible for bringing a solution to these problems. This difficult the development of solutions based of TrustZone applications. At the same, it may produce a lack of authenticity and integrity guarantees in devices that do not offer separate hardware modules for these functions [144], [153].

Intel Software Guard Extensions

Intel SGX is a set of Intel CPU instructions that provide integrity and confidentiality to computation, even when an attacker compromises privileged software such as the kernel or the hypervisor [141]. The base of Intel SGX is a trusted container, also called an enclave, that is protected by trusted hardware, and its integrity can be attested remotely. The enclave will only install applications signed by a trusted party, which is currently Intel. Each enclave can have multiple applications, and a CPU may have multiple enclaves.

The enclave data and code are stored in the Processor Reserved Memory (PRM), a subset of DRAM restricted to enclaves. Inside the PRM, there is an Enclave Page Cache (EPC), which is divided into multiple pages. Each page can be assigned to a single enclave. An enclave has multiple EPC pages and cannot read pages assigned to others. A page can have multiple types, from ones mapped to the enclave address space to metadata used in its lifecycle [141].

If an EPC needs to be stored in an untrusted memory, SGX will cipher and sign it, to assure its confidentiality and integrity. The only time an untrusted application can write to the PRM is during the loading stage of an application to the enclave, when the application is copied and EPC pages are allocated. This process is cryptographically hashed and then used for software attestation.

The enclave's virtual memory is not the only memory that lives inside the EPCs. The enclave has the option to have memory mapped from the outside in its virtual memory. This allows enclaves to use existent libraries from the non-secure world or act as a library for processes outside the enclave. In these cases, non-enclave software cannot access PRM memory [141].

In any context switching from an enclave to an application outside the enclave, the CPU, to avoid data leakage, saves its state to a predefined area and cleans its registers before transferring execution.

Intel SGX enclaves run at the lowest privilege level possible (user mode). Therefore, enclave application development is similar to non-enclave applications. The developer has a set of libraries that can use and a Software Development Kit (SDK) to compile and deploy the application [154]. Nevertheless, multiple SDKs work on the Intel-provided SDK to facilitate the development of secure SGX applications [155], [156].

The security restrictions applied to SGX enclaves are the same as non-enclave applications. Enclaves will not be able to interact directly with computer devices [141]. Regardless, having software that no one can access is also a security problem. Current anti-virus scan executables, files, and memory looking for patterns that indicate malicious activity. Therefore, SGX technology can evade these analyses, which means if a malicious actor lives inside an enclave, he will not be detected.

In terms of physical security, even though the SGX threat model excludes physical attacks targeting the CPU chip and side-channel attacks, it considers attacks to the DRAM, its bus, and debugging ports [157].

Attacks on CPU chips are complex and require expensive equipment. Therefore they are less common. The intrinsic characteristics of SGX are not publicly known. However, researchers have analyzed patents related to the Intel SGX and concluded that some countermeasures exist to increase the difficulty of attacks against the CPU chip [141]. For instance, is possible that existent keys are hardcoded with fuses in the CPU circuit, or PUFs are used to generate them.

Intel SGX is affected by multiple side-channel attacks, some of which are specific to Intel processors. Enabling these attacks, we have two processor features, hyper-threading, and speculative execution, that try to optimize the processor execution.

Hyper-threading divides physical cores into multiple logical cores, which share resources like cache and execution units [158]. Speculative execution is an optimization technique for optimizing an instruction pipeline.

Multiple steps must be run sequentially to execute instructions in a CPU, each taking one clock cycle. The processor improves the performance of these steps by doing them in parallel for different instructions since they are executed in different parts of the CPU. If a branch instruction is encountered, the processor will only know the next instruction after executing it, which would cause a performance hit, by stalling the pipeline. With speculative execution, the processor will choose one of the execution flows and start loading its instructions into the pipeline. After executing the branch instruction, the processor will discard or use the loaded instructions, according to the branch result [159].

These optimizations were leveraged with architecture problems to create multiple attacks against SGX. Branch prediction attacks exploit information leaks in the component responsible for predicting the execution flow before a branch instruction. Lee et al. [159] was the first to introduce this attack, and other authors [155] used their research to extract a private key that was stored inside an enclave.

Also, related to the speculative execution, in 2018, two vulnerabilities were discovered affecting applications outside the enclave, Spectre [160] and Meltdown [161]. In addition to the speculative execution, these vulnerabilities exploited out-of-order execution to execute unintended instructions. Initially, the vulnerabilities were not able to affect SGX enclaves, but with further research and adaptation, researchers attacked SGX enclaves with the same principles [162], [163].

Finally, in 2019, Microarchitectural Data Sampling(MDS) was introduced, a new class of attacks enabling the bypass of common security boundaries like processes, virtual machines, and enclaves by exploiting flaws in undocumented buffers to leak information [164]–[166].

Fortunately, the vulnerabilities presented here can be fixed through a microcode update to the processor [156].

SGX sets a RoT that ensures the confidentiality and integrity of the TEE. Only enclaves properly signed can be installed. At the same time, SGX provides attestation capabilities. Regardless, these two features are not supported by hardware. Each SGX-enabled CPU has a privileged enclave, called *Quoting Enclave*, installed by Intel that is responsible for measuring the data and code loaded to each enclave and offers remote attestation capabilities. The measurements provided by the *Quoting Enclave* are very similar to the ones provided by TPMs. Despite that, the signature algorithms are different and are not implemented in tamper-resistant hardware [167].

2) Security attacks and countermeasures:

As we stated before, generally, TEEs share their hardware resources with the rest of the computation environment, which brings security risks. In this subsection, we will analyze these consequences, that affect both TEEs implementations presented before.

First of all, TEEs do not protect against software vulnerabilities, which means if the TEE or SDK implementation has a security flaw, this will impact the security of this TEE. Since the introduction of TEE's, both technologies have been affected by software vulnerabilities. Bulck et al. [168] discovered multiple vulnerabilities affecting Intel SGX SDKs. Regarding Arm TrustZone TEE's, researchers have discovered several software vulnerabilities [144]. When writing this article, the NIST vulnerability database had 73 vulnerabilities related to Arm TrustZone [169]. This amount of vulnerabilities is a consequence of the level of knowledge required to develop a TEE. As we stated before, Arm TrustZone is a bare-metal technology that developers use to create their TEE implementations, which means many of the features are implemented by the developer. Therefore, most vulnerabilities compromise TEE's implementations and not the Arm TrustZone directly [170].

On the hardware side of the TEE, side-channel attacks are the main class of attacks affecting TEEs. Among them, cache-based attacks appear as a specific attack against TEEs. Side-channel attacks on the processor cache are made by a malicious application running in the untrusted world, sharing the same processor core with the TEE application. This malicious application fills the processor cache with its data and waits for the TEE application to run and evict its data from the cache. Afterward, the malicious application will access the same data and measure the time it takes to access the information. As access to information stored in the CPU's cache is faster than RAM, the application can figure out accesses patterns of the TEE, and with that and knowledge about the TEE's code, it can extrapolate information about the TEE's execution [156]. Using these principles, multiples attacks were developed, jeopardizing both Intel SGX and Arm TrustZone [149]–[151], [171], [172].

Despite that, TEEs are also vulnerable to other types of side-channel and physical attacks. The literature states that Intel SGX and Arm TrustZone do not have protection against EM, power analysis attacks, or fault injections [141], [144]. For both technologies, we found research performing in practice these attacks. Bukasa et al. [173] analyzed EM attacks against Arm TrustZone, and Chen et al. [174] performed a voltage glitch attack on an Intel SGX enclave. Both researchers were able to recover cryptographic keys from a TEE.

Intel SGX threat model excludes physical threats to the device security due to the inherent cost of hardening a general-purpose CPU. Despite that, the threat model includes threats against the bus connecting the Random Access Memory (RAM) to the CPU due to the likelihood of eavesdropping attacks. Therefore, any data that SGX needs to store in RAM is properly ciphered and signed. Regardless, information inside the internal CPU buses is transmitted in clear [141], [175].

Therefore, in essence, TEEs offer interesting security features to increase resilience against software attacks. Even so, TEEs do not have protection against the majority of the physical attacks, and, if compromised, they can be used as a persistence method for attackers [138].

3) Advantages and disadvantages for identity assurance:

The significant advantage of TEE for identity assurance is the fact that it allows the creation of a secure execution environment without additional hardware. Therefore, less cost and power consumption. Nevertheless, as we saw earlier, Arm TrustZone requires further adaptation to assure RoT, which may include additional hardware.

TEEs can offer a greater performance when compared with solutions such as SEs [176], in terms of CPU, RAM, and storage capabilities. For instance, Arm TrustZone enabled

CPUs can offer RAM in the gigabyte range , in comparison, SEs have RAM in the kilobyte range.

Similar to other technologies, developers need to dominate a new software stack to develop solutions using TEEs.

Finally, it is imperative that developers precisely understand this technology's limitations, as it does not defend against every physical attack and introduces new risks due to the sharing of components with the non-secure world.

4) Implementations:

TEE solutions have been mainly used to store cryptographic keys, provide remote attestation, identity, and ensure the security and resilience of the application even if the device is compromised. Most of the solutions are based on public key cryptography, in which each device has a private key used to identify the device.

There are multiple solutions following these principles. For instance, Ling et al. [177] developed a system that provides secure boot and remote attestation for IoT devices, leveraging Arm TrustZone. This research uses ROM and eFuses, to overcome the secure storage limitation of the Arm TrustZone and ensure the security of a RoT.

Another example is from Lesjak et al. [176], which implemented two authentication systems, one using an Arm TrustZone enabled CPU and the other with a SE, to analyze the advantages and disadvantages of each construction. Finally, they propose a hybrid system that combines Arm TrustZone and SE to overcome the security risks of the Arm TrustZone.

Wang et al. [178] implemented a similar solution with Intel SGX to create a lighter solution for remote attestation compared with TPMs, with the added befit of offering a secure environment to run applications.

Durand et al. [179] developed a lightweight communication system backed up by hardware. However, since Intel SGX CPUs are too expensive for most IoT devices, they used a secure element in the device and an Intel SGX enclave in the server to receive the device communications securely.

More recent researchers are trying to integrate device identity, based on TEEs, with blockchain technology [180], [181].

F. PUFs

PUFs are physical systems that, given a specific input (a *challenge*), produce a string of bits, the response [182]. The response is unique and unpredictable because it depends on the unique hardware characteristics of each device that are a consequence of the physical world variations during the manufacturing process [183]. The challenge and the corresponding response are called a Challenge- Response-Pair (CRP) [184]. Depending on the type of PUF, it can be configurable or not, which means that any change in the challenge will induce a change in the response [16].

The concept of PUF is generic to include systems from different application fields. Regardless, there are still constructions that are not called PUF because they were designed outside the field of hardware security engineering [16].

Many authors consider PUFs as one of the technologies that can help IoT overcome the device identity challenges [185]–[187] due to the promise of a cheaper and safer way to generate and store cryptographic keys compared with EEPROM solutions that offer the same assurance level [188].

PUF's constructions are evaluated according to two metrics, intra-distance and inter-distance. Intra-distance is the hamming distance between two responses from the same PUF instance using the same challenge. Inter-distance is the hamming distance between two responses from different PUF instances when the same challenge is applied [16]. On top of these metrics, researchers can also analyze the PUF's reproducibility and uniqueness.

The distribution of the response intra-distance gives us the PUF reproducibility. PUFs are not mathematical functions because a single input (challenge) can generate multiple outputs due to physical environment changes and random noise in the response generation [189]. Therefore, reproducibility is an essential characteristic of a PUF-based system. Furthermore, PUF solutions employ fuzzy extractors to handle these variations and return a stable response [189].

The distribution of the response inter-distance analyzes the uniqueness of a PUF. When challenged with the same input, different PUFs should produce distinct responses. The responses' inter-distance should ideally be 50% to be considered a true random generator [188].

A PUF that is reproducible and unique is also identifiable. This means that using a PUF response to identifying a device is feasible because its response is unique and stable [16].

In addition to these characteristics, another set of features defines PUFs, namely being tamper-resistant, unclonable and unpredictable.

Tamper-resistance is the capability of resisting attempts of unauthorized physical modifications to leak information or bypass some security protection. Because PUF constructions rely on measurements of physical features, any slight variation provokes a change in its response. Therefore any attempt to tampering a PUF would cause a noticeable change in its CRPs, ultimately originating a new PUF instance [16].

PUFs are also unclonable because of these precise measurements. In any cloning attempt, the attacker would not be able to reproduce all the PUF characteristics since they result from physical variations during the manufacturing process [16].

Another characteristic that differentiates PUFs, from a security standpoint, is their predictability. Based on this characteristic, there are two types of PUFs, Weak and Strong PUFs. This property depends on the resilience of a system against an attacker that tries to predict all CRPs [190], [191]. Strong PUFs are the ones that, even when exposed for an extended period to an attacker, it is impossible to predict their responses to a challenge. All the other constructions that do not meet this requirement are considered Weak PUFs [16].

Strong PUFs have a large set of CRP, which prevents the creation of a database with all possible pairs. Even if attackers know a large subset of CRPs, they cannot predict unknown ones. Moreover, if the attacker physically possesses the PUF, the Strong PUF security is not compromised [188], [192]. By contrast, Weak PUFs may have a single CRP. Thus, the security system is compromised if an attacker obtains its response [188], [193]. Moreover, most Weak PUFs have a single CRP, allowing cloning of the device [192].

Strong PUFs offer higher security guarantees than Weak PUFs. However, some Strong PUF constructions have become vulnerable to modeling attacks, due to the emergence of new technologies. These attacks require a considerable amount of CRPs to model the PUF response. Researchers have known about these attacks since the beginning of this research field. Nevertheless, with the advancement of machine learning techniques, Strong PUF constructions that were known as secure, are now considered vulnerable to these attacks [194]–[198].

These attacks have been the focus of recent research. This leads some authors to suggest that Strong PUFs are still an open research challenge and require further development to meet the expected security features that are being jeopardize by these attacks [16], [199], [200].

1) Security attacks and countermeasures:

PUFs promise a security improvement in the storage of cryptographic keys since it eliminates potential offline attacks when the information is not being used. PUFs do not have countermeasures against physical attacks, given that any attempt to inject a fault or tamper with the device would induce changes in the PUF's response. Despite that, depending on the construction, PUFs may be vulnerable to side-channel attacks or reverse engineering with the objective of modeling the PUF's response [201]. For instance, Different Delay-based PUFs, such as the Arbiter PUF, which explore delays between two competing signal paths, can be vulnerable to power-side channel analysis [201] and optical side-channel attacks [202], and RO PUFs are vulnerable to electromagnetic side-channel attacks [203].

It is argued that PUFs resist to reverse engineering attempts due to their complexity [16]. Still, Nedospasov et al. [204] attacked SRAM PUFs with SEM, a reverse engineering technique, as a way to model its responses. SRAM PUFs exploit that after a power-on/power-off cycle, a memory cell has the same probability of being set as a 0 or 1. At the same time, each cell tends to keep the same behavior over power cycles [205]. As the state of each cell can be observed with SEM, its CRPs may be retrieved. SRAM PUFs have the advantage of being a low-cost solution and simple to implement [206]. On the other hand, these advantages also ease the attacker's work, given that it decreases the PUF's complexity.

Even though we listed several attacks against common PUFs, these can be overcome by an improved design or countermeasures. For instance, Nedospasov et al. [204] propose that new SRAM PUF constructions have an asynchronous reset mechanism of its memory cells to decrease the information exposure, and Merli et al. [203] suggest multiple changes in the RO PUF design to decrease the emanation of electromagnetic radiations.

In summary, while the inherited dependency on physical characteristics protects PUFs against tampering attempts, it does not protect against side-channel attacks. Therefore, side-channel attacks must be mitigated with additional countermeasures.

2) Advantages and disadvantages for identity assurance:

In general, PUFs are considered a possible solution for IoT key management due to being a cheaper solution when compared with other methods to store cryptographic keys like antitampering EEPROM, at the same time, it requires less circuit space and energy to operate [188].

Moreover, Strong PUFs, when are able to resist to modeling attacks, provide lightweight authentication and identification protocols that not rely on heavy cryptographic algorithms [207].

PUFs promise a way to securely manage device identity, protected against invasive attacks like key extraction or any attempt of tampering. However, as we saw earlier, both Secure and Weak PUFs have potential vulnerabilities, as we stated before. Thus, the system using this technology should beware of these risks and accept them.

In addition, most systems based on Strong PUFs assume the server, where the CRPs are stored, is secure [208]. However, for some use cases, this risk is unacceptable. If the server is compromised, the confidentiality of all CRPs pairs is jeopardized, which means the attacker can authenticate itself as any device.

3) Implementations:

In the field of identity assurance, we can see the use of PUFs in two fields, generation of secure keys and authentication. Namely, Strong PUFs are used for authentication and Weak PUFs for key generation [188].

For authentication, the typical protocol has the following construction, after a PUF is manufactured, it is submitted to multiple Challenges in a secure environment to ensure CRP confidentiality. Known CRPs are then securely stored and used for device authentication. Each challenge is only used once, to mitigate replay attacks [200], [209].

This type of authentication protocol has been used since the rise of this technology. Early examples are Gasend et al.' Silicon PUF [210] and Lim et al.' Arbiter-Based PUF [211]. However, the threat model of these early implementations was too strict, and because of that, they were vulnerable to modeling attacks [212].

To hamper modeling attacks, the next iteration of PUF constructions started using one-way functions applied to the PUF response to prevent direct access to the PUF CRP [209], [213]–[215]. This type of PUF is generally known as Control PUF. Once again, further research discovered vulnerabilities in this type of construction due to reversible one-way functions and pattern matching [195], [216].

Current research is working on improving existent constructions, making them resilient to modeling attacks. The main approach is to close the PUF interface by implementing mutual authentication [217]–[220].

There is also research trying to prevent modeling attacks, at the same time, it improves the authentication protocol.

Chatterjee et al. [221] and Qureshi et al. [208] developed solutions that don't assume a secure CRP database, which means that even if the server responsible for the authentication is compromised, the CRPs are not because they are not stored in clear text.

Ebrahimabadi et al. [222] go further to mitigate the eavesdropping of CRPs and modeling attacks. They created an authentication protocol that scrambles and divides the communications between the server and a node (the device with a PUF that needs to be authenticated) into multiple packets. These packets are sent through multiple nodes to obscure the actual destination of the message. With that, the authors expect that attackers cannot relate the CRPs with a specific device, so it will be impossible to perform a modeling attack.

Hang et al. [223] use the classic authentication protocol with Strong PUFs. However, they combine multiple PUFs on the same device to create a fingerprint that changes if any of the device's components is tampered. This construction uses a Configurable RO PUF as a hardware security primitive and a latch structure to extend the key space of the responses, increasing its resilience. This solution uses the same style of authentication protocol, where we have a server that stores multiple CRPs and then queries them.

On the other hand, Weak PUFs used in IoT systems do not try to replace traditional authentication systems. But instead, they attempt to improve the storage of cryptographic keys on a cheap IoT device. A Weak PUF generates the same secret key every time the device is running. Therefore, this key does not need to be stored, which mitigates the risk of an attacker extracting the key when the device is not running.

For this generation process to be reliable, the secret key extraction from a Weak PUF response requires an extra step, a *helper data algorithm* [1]. Any change on the cryptographic key is not acceptable for most encryption protocols. Thus, this algorithm allows the extraction of a secret key from a noisy or not uniform response [224], [225].

The generation process of a secret key from a Weak PUF normally has two phases, the enrollment and the reconstruction. The enrollment happens when we want to create a new secret key, and the reconstruction is when we need to obtain the same secret key after its creation.

During the enrollment phase, from a PUF response, it is generated a secret key and helper data. The key has to be kept secret, but the helper data does not, and can be stored in non-secure NVM. The reconstruction phase uses the helper data, that was obtained before, to generate the same secret key given a another PUF response from the same PUF [226], [227].

The first practical work using PUFs to securely manage cryptographic keys was done by Škorić et al. [226] and Suh et al. [228]. Over the years, new research on this topic emerged. Some of this research used new types of PUFs to generate secrets [205], [229], and others focused on creating re-configurable PUFs that enable the change of secret keys over time [227], [230], [231]. In this type of research, we can also see the use of Strong PUFs due to the amount of CRPs that allow the generation of multiple keys on the same PUF instance [224].

Lastly, researchers have been developing PUF-based systems capable of generating a shared key among different resource-constrained devices to enable multiparty communication [232].

In summary, there are two main approaches to enhance authentication and identity on IoT devices using PUFs, lightweight authentication protocols using CRP databases and secret key generation. Authentication protocols with CRP databases depend on the resilience of the PUF against security attacks. Nevertheless, when correctly employed, this technology answers the needs of constrained IoT devices when an energy-efficient way to authenticate a device is necessary without requiring intensive computation. On the other hand, PUF-based key generation works a security primitive for existent identity and authentication systems. Therefore, it does not solve the need for heavy computation, but it offers a cheap and secure way to generate and store unique secret keys.

[1]A *fuzzy extractor* is a specific type of *helper data algorithm* [224].

VI. Discussion

The different hardware technologies presented in this research can be used to help the development of IoT identity systems. However, they are very different in the way they can support them. Furthermore, regardless of its usefulness for identity systems, the designer needs to analyze its strengths and weaknesses to decide which technology should be included in a device.

The main concerns during this assessment are how the technologies can benefit the system and contribute to overcoming the IoT identity challenges, the advantages and disadvantages of each technology and, finally, what security countermeasures they have. In this section, we will analyze and compare the technologies presented earlier. We will focus on these points to assist any system developer who wants to create a system with an identity based on hardware primitives.

In Section III, we list three challenges to developing an identity system: lightweight encryption, object identification, and secure storage. The technologies analyzed can be used to overcome these challenges or at least circumvent them.

Each encryption accelerator, SEs and TEEs can provide ways to get around the need for lightweight encryption algorithms. All these technologies can run cryptographic algorithms optimally, making these operations faster and more energy efficient. Among these technologies, it is essential to highlight the cryptographic instruction sets (a subtype of the cryptographic accelerator), as they have the potential to normalize the implementation of cryptographic algorithms in hardware. Strong PUFs and TRNGs, not so directly, also contribute to the research challenge in lightweight cryptography. TRNGs do not provide a way to run cryptographic algorithms optimally. Instead, they provide a way to get high entropy numbers, which is essential for any asymmetric encryption scheme. PUFs can be exploited for lightweight authentication systems, easing the need for cryptographic algorithms.

Regarding object identification, Intel SGX and SEs offer an isolated execution environment and secure storage capabilities that can be exploited to create object identification systems. Another technology with similar possibilities is TPMs. However, the developer is limited to the features offered by the TPM specification. Taking a different approach, we have strong PUFs, which have CRPs entirely dependent on the device's hardware characteristics, which can be used to uniquely identify a device. Also, strong PUFs self-destruct on any tampering attempt.

Finally, masked ROM and OTP memories provide read-only memories that can store RoTs. Regardless, they do not

Technologies		Lightweight Cryptography	Object Identification	Secure Storage
TRNG		●		
Masked ROM and OTP memories	Masked ROM			●
	Floating gate OTP			●
	Fuse OTP			●
	Antifuse OTP			●
Crypto accelerators	Crypto Instruction Sets	●		
	Crypto Coprocessors	●		
	Crypto Processor	●	●	●
Secure element		●	●	●
TEE	Intel SGX	●	●	●
	Arm TrustZone	●		
PUF			●	●

TABLE III
TECHNOLOGIES AND RESEARCH CHALLENGES

protect data at rest, which other technologies like Intel SGX and SEs do. TPMs and PUFs can also ease the challenge of secure storage. However, they offer a limited set of features. TPMs can only store cryptographic keys, and weak PUFs are even more limited and cannot import cryptographic keys that are generated outside of PUF. Table III summarizes all this information.

Each of the technologies that may support the device's identity will relate to identity assets and, as a consequence, are critical to achieve any security goal (Subsection IV-B). For instance, we have stated that masked ROM and OTP memories do not provide countermeasures to protect data at rest. This means that identity data and firmware have their confidentiality at risk, given that an attacker can read the stored information when the device is turned off [69], [98], [100]. The other four technologies featuring secure storage (TPM, SE, Intel SGX, and PUF) have countermeasures to protect data at rest and some even during its execution.

Intel SGX ciphers the information every time it needs to leave the CPU package, but besides that, it does not have any further security countermeasure to prevent physical attacks against confidentiality and integrity of the identity data and firmware, which means it is vulnerable to fault injection attacks [233] and side-channel attacks.

By contrast, TPMs and SEs have several countermeasures to prevent fault-injections and side-channel attacks. SEs are known to have the most complete set of security features, and TPMs benefit from this fact since their design is many times based on SE's design but with some compromises considering that the security assurances of a TPM are not as demanding as the ones of a SE [234].

PUFs distinguish themselves from other technologies in terms of security, considering they do not have physical security countermeasures. The risk of fault injections is neglectable because any attempt would alter the PUF's response, making it unusable. However, side-channel attacks and reverse engineering attempts would not affect its response. Researchers consider that PUFs are too complex to be vulnerable to these attacks, which in many cases is true, but it can also be a careless assumption for some constructions [204].

Looking at the remaining technologies, ARM TrustZone, crypto accelerators, and TRNGs do not have any security countermeasures by default. In any case, it is important to mention that with the exception of ARM TrustZone, these components are base building blocks, which means the device designer is responsible for installing of physical security countermeasures.

Despite the security countermeasures, physical security is not perfect, mainly against reverse engineering attempts. If the attacker spends enough time, he will be able to bypass the implemented countermeasures. The majority of these countermeasures do not eliminate risks but rather increase the attack complexity and time to succeed.

Table V summarizes the different countermeasures that are commonly available in the presented technologies.

Each of the technologies that may support the device's identity will report to identifying assets and, as a consequence, are critical to achieving any security goal (Subsection IV-B). For instance, we have stated that masked ROM and OTP memories do not provide countermeasures to protect data at rest. This meaManyns that identity data and firmware have their confidentiality at risk, given that an attacker can read the stored information when the device is turned off [69], [98], [100]. The other four technologies featuring secure storage (TPM, SE, Intel SGX, and PUF) have countermeasures to protect data at rest and some even during its execution.

Intel SGX ciphers the information every time it needs to leave the CPU package, but besides that, it does not have any further security countermeasures to prevent physical attacks against the confidentiality and integrity of the identity data and firmware, which means it is vulnerable to fault injection attacks [233] and side-channel attacks.

By contrast, TPMs and SEs have several countermeasures to prevent fault injections and side-channel attacks. SEs are known to have the most complete set of security features, and TPMs benefit from this fact since their design is many times based on SE's design but with some compromises considering that the security assurances of a TPM are not as demanding as the ones of a SE [234].

PUFs distinguish themselves from other technologies in terms of security, considering they do not have physical security countermeasures. The risk of fault injections is neglectable because any attempt would alter the PUF's response, making it unusable. However, side-channel attacks and reverse engi-

Attack \ Technology	TRNG	Masked ROM OTP memories	Crypto accelerators			Smart cards	TEE	PUF
			Crypto Instruction Sets	Crypto Coprocessors	Crypto Processor			
Voltage sensors	No	N/A	No	No	Yes*	Yes	No	No
Multiple voltage sensors	No	N/A	No	No	No	Yes	No	No
Clock signal sensor	No	N/A	No	No	Yes*	Yes	No	No
Active metal shield	No	No	No	No	Yes*	Yes	No	No
EM radiation sensor	No	N/A	No	No	No	Yes	No	No
Voltage monitoring	No	N/A	No	No	No	Yes	No	No
Defensive PCB design	No	No	No	No	Yes*	Yes	No	Yes

N/A = Not applicable;
* = Depends on the model

TABLE IV
COMMON COUNTERMEASURES OF EACH TECHNOLOGY

neering attempts would not affect its response. Researchers consider that PUFs are too complex to be vulnerable to these attacks, which in many cases is true, but it can also be a careless assumption for some constructions [204].

Looking at the remaining technologies, ARM TrustZone, crypto accelerators, and TRNGs do not have any security countermeasures by default. In any case, it is important to mention that except for ARM TrustZone, these components are base building blocks, which means the device designer is responsible for installing physical security countermeasures.

Despite the security countermeasures, physical security is not perfect, especially against reverse engineering attempts. If the attacker spends enough time, he can bypass the implemented countermeasures. Many countermeasures do not eliminate risks but rather increase the attack's complexity and time to succeed.

Table V summarizes the different countermeasures that are commonly available in the presented technologies.

Finally, we can examine each technology's features and advantages, and disadvantages. To start, TRNGs offer a high-quality entropy source with the cost of adding dedicated hardware to the device, which increases the device's power consumption. Moreover, due to the physical exposure of devices to attackers, TRNGs may be vulnerable to environment bias.

There are four types of ROMs, masked ROM, floating-gate OTP, eFuse OTP, and anti-fuse OTP. These memories can be programmed a single time, after manufacturing, except for masked ROM, which can only be programmed during manufacturing since it stores the information hardwired in its design. Despite this, floating-gate OTPs and eFuse OTPs have some disadvantages. Floating-gate OTPs are vulnerable to optical attacks and resetting the memory, and eFuse OTPs have a limited duration of data retention.

As stated before, Crypto accelerators offer optimized cryptographic operations through specialized hardware. Therefore, except for cryptographic instruction sets, every type of crypto accelerator is an additional component that needs to be added to the device and properly integrated. Crypto-processors and TPMs typically offer libraries to facilitate their integration with the rest of the system. By contrast, crypto-coprocessors are integrated at the CPU level.

In addition to providing optimized cryptographic operations and secure storage, SEs offer a secure execution environment independent of the primary system. SEs have a high assurance level but limited computation power and storage.

TEEs also provide an execution environment. However, their security assumptions are different. Usually, these environments share their hardware with the rest of the system, which introduces advantages and disadvantages. TEEs are a feature of the main CPU, which means it does not require dedicated hardware. Regardless, this fact also increases their security risks.

Moreover, each of the TEEs presented has its own positive and negative points. Intel SGX enabled processors provide a RoT and secure storage but are more expensive than Arm Trust-Zone enabled processors, which are cheaper but do not provide secure storage or RoT. On top of that, Arm Trust-Zone enabled CPUs work at the system level, allowing developers to run an operating system for each execution environment, increasing the system flexibility and development complexity. In contrast, Intel SGX trusted environments are developed on top of SDKs that abstract the complexity of low-level operations.

Finally, PUFs provide a way to generate and store keys tiddly dependent on the device's characteristics. Additionally, Strong PUFs have the benefit of enabling lightweight cryptography authentication algorithms. However, independently of the PUF type, there are disadvantages. In their majority, PUFs depend on dedicated hardware, which requires low-level integration with the system. Moreover, with the exception of some commercial off-the-shelf products, the majority of the time, system designers need to build their own PUFs on the board, which increases the burden of using this technology.

Table VI summarizes the different advantages and disadvantages of the technologies presented.

VII. Limitations and Future Directions

During this research, we examine many technologies that can be used to support identity and authentication operations in IoT. As we have seen, most of them have working prototypes of identity and authentication systems, and some have already been deployed in production. Regardless, we do not see widespread use of these technologies in IoT devices.

Over the years, the cost of adding specialized hardware has been cited as the reason for the lack of adoption of these solutions. However, as other researchers have pointed out [17], this is a misconception. Our analysis confirms that there are solutions in different price ranges, even for smaller budgets. Therefore, it is necessary to consider other reasons for the low adoption of these technologies.

Attack \ Technology	TRNG	Masked ROM OTP memories	Crypto accelerators			Smart cards	TEE	PUF
			Crypto Instruction Sets	Crypto Coprocessors	Crypto Processor			
Voltage sensors	No	N/A	No	No	Yes*	Yes	No	No
Multiple voltage sensors	No	N/A	No	No	No	Yes	No	No
Clock signal sensor	No	N/A	No	No	Yes*	Yes	No	No
Active metal shield	No	No	No	No	Yes*	Yes	No	No
EM radiation sensor	No	N/A	No	No	No	Yes	No	No
Voltage monitoring	No	N/A	No	No	No	Yes	No	No
Defensive PCB design	No	No	No	No	Yes*	Yes	No	Yes

N/A = Not applicable;
* = Depends on the model

TABLE V
COMMON COUNTERMEASURES OF EACH TECHNOLOGY

Technologies		Features	Advantages and disadvantages
TRNG		High quality entropy source	- Increased power consumption - Risk of environment bias - Dedicated hardware
Masked ROM and OTP memories	Masked ROM	Read-only memory	+ Hardwired data - Programmed during manufacturing
	Floating-gate OTP	Read-only memory	+ Programmable after manufacturing - Susceptible to UV attacks
	eFuse OTP	Read-only memory	+ Programmable after manufacturing - Limited data retention
	Anti-fuse OTP	Read-only memory	+ Programmable after manufacturing
Crypto accelerators	Cryptographic Instruction Sets	Optimized cryptographic operations	+ No additional hardware + Easy integration
	Crypto-coprocessors	Optimized cryptographic operations	- Complex integration - Dedicated hardware
	Crypto-processor	Optimized cryptographic operations	+ Easy integration - Dedicated hardware
	TPM	Optimized cryptographic operations Secure storage and RoT	+ Easy integration - Dedicated hardware
Smart cards		Optimized cryptographic operations Secure execution environment Secure storage and RoT	+ High assurance security level - Dedicated hardware - Limited computation power
TEE	Intel SGX	Optimized cryptographic operations Secure storage and RoT Trusted execution environment	+ No additional hardware - Deprecated in Intel Core processors - Expensive TEE option
	Arm TrustZone	Optimized cryptographic operations Trusted execution environment	+ No additional hardware + Cheap TEE option + Flexibility - Secure storage or RoT nonexistent - Development complexity
PUF		Generation of keys tidly dependent on device's characteristics Lightweight cryptography authentication algorithm	- Complex integration - Dedicated hardware

TABLE VI
SUMMARY OF ADVANTAGES AND DISADVANTAGES OF EACH TECHNOLOGY

The disregard for security can be one of the reasons for the lack of adoption of hardware-based solutions. However, some devices implement adequate software-level protection but do not employ hardware RoTs. Therefore, in these cases, your threat model accepts that the risk of physical attacks or the cost of hardware solutions is not worth it when compared to the value of the information in question.

Each technology has its library or software stack to interact with it. These are low-level APIs, so if we want to use them to support an identity mechanism, we need to build our solution using these libraries.

Device designers often do not design a device from scratch, but instead, use an existing SoC as a foundation for adding their features. These SoCs include SDKs to make software development easier, which means that if there is no support for this kind of technology in existing SDKs, it will discourage most software designers from devices. Furthermore, even though the SDK supports the technology in question, it also needs to provide an authentication and identity framework that uses these technologies and hides the complexity of a hardware RoT. If that does not exist, designers will prefer traditional systems over implementing their custom ones with hardware security, which requires experienced staff and is error-prone.

Therefore, if the promotion of hardware security depends on the adoption of these mechanisms by SDKs, the lack of standardization in IoT also affects this challenge. Assume there is no standard hardware-based identity and authentication framework. In this case, each vendor will implement a framework, increasing development diversity and effort, as the developer's knowledge of a framework does not apply to frameworks from other vendors.

Finally, the limitations of some technologies may be the lack of ready-to-use components. For example, in the case of PUFs, from the market analysis, we did at the time of writing, few off-the-shelf components include or provide a PUF. This technology is not new and holds many promises for IoT. However, without the availability of off-the-shelf components, device designers are forced to implement PUFs from scratch, which is a barrier to their proliferation.

So the cost of hardware security is not necessarily the additional hardware added to the device, but the cost of integrating with the rest of the device. To combat this trend, development SDKs should include hardware-based authentication and identity frameworks to facilitate the integration of hardware RoTs into new systems. In addition, the standardization of these frameworks should be a priority so as not to create diversity between the manufacturers' frameworks and increase the learning curve for the use of these technologies.

VIII. Conclusion

IoT devices interact with our personal life and manage critical infrastructures. Thus, keeping them secure is a priority.

Identity and authentication play a vital role in the security of these devices. Without it, it is impossible to guarantee the device's security since we would be unable to assure the veracity of any information. Nevertheless, identity and authentication are considered open research challenges in IoT. Resource-constrained devices, a lack of standardization and exposure to physical attacks are only some of the reasons that make identity and authentication in IoT so challenging.

Over the years, multiple researchers have advocated using hardware to undermine these challenges. Regardless, widespread adoption of hardware technologies supporting identity and authentication has not been seen.

During our work, we focused on hardware trust anchors and their security features that can be exploited to develop new identity and authentication systems.

We analyzed physical risks for IoT identity and identified possible countermeasures. We retrieved that hardware trust anchors must employ protections, like multiple sensors, active metal shields and a defensive PCB design, to protect themselves against physical risks. Besides that, we also explored how challenging these risks are since we cannot mitigate them completely but rather increase the difficulty of an attack.

With these security features and identity challenges in mind, we reviewed technologies available to designers to develop new identity and authentication systems. In this analysis, we included the following technologies: TRNGs, masked ROMs and OTP memories, crypto accelerators, secure elements, TEEs and PUFs.

We concluded that there are multiple candidate technologies that might support new identity and authentication systems, aiming at different price points. Indeed, these technologies can overcome some of the challenges holding back identity and authentication in IoT by enabling the use of common cryptographic algorithms in low-power devices and offering resilience against physical attacks. Unfortunately, the complex integration process of some of these technologies and the required knowledge to effectively use them continue to halt the widespread use of hardware trust anchors in IoT.

References

[1] A. Nordrum, "(2016). popular internet of things forecast of 50 billion devices by," 2020.

[2] D. Hanes, G. Salgueiro, P. Grossetete, R. Barton, and J. Henry, *IoT fundamentals: Networking technologies, protocols, and use cases for the internet of things*. Cisco Press, 2017.

[3] D. Wang, D. Chen, B. Song, N. Guizani, X. Yu, and X. Du, "From iot to 5g i-iot: The next generation iot-based intelligent algorithms and 5g technologies," *IEEE Communications Magazine*, vol. 56, no. 10, pp. 114–120, 2018.

[4] M. Marjani, F. Nasaruddin, A. Gani, A. Karim, I. A. T. Hashem, A. Siddiqa, and I. Yaqoob, "Big iot data analytics: Architecture, opportunities, and open research challenges," *IEEE Access*, vol. 5, pp. 5247–5261, 2017.

[5] M. Iorga, L. Feldman, R. Barton, M. J. Martin, N. Goren, and C. Mahmoudi, "Fog computing conceptual model," Tech. Rep., mar 2018.

[6] N. Yousefnezhad, A. Malhi, and K. Främling, "Security in product lifecycle of IoT devices: A survey," vol. 171, p. 102779, dec 2020.

[7] A. R. H. Hussein, "Internet of things (iot): Research challenges and future applications," *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 6, pp. 77–82, 2019.

[8] H. U. Rehman, M. Asif, and M. Ahmad, "Future applications and research challenges of iot," in *2017 International conference on information and communication technologies (ICICT)*. IEEE, 2017, pp. 68–74.

[9] S. A. Al-Qaseemi, H. A. Almulhim, M. F. Almulhim, and S. R. Chaudhry, "Iot architecture challenges and issues: Lack of standardization," in *2016 Future Technologies Conference (FTC)*. IEEE, 2016, pp. 731–738.

[10] M. Nawir, A. Amir, N. Yaakob, and O. B. Lynn, "Internet of things (iot): Taxonomy of security attacks," in *2016 3rd International Conference on Electronic Design (ICED)*. IEEE, 2016, pp. 321–326.

[11] A. Cirne, P. R. Sousa, J. S. Resende, and L. Antunes, "Iot security certifications: Challenges and potential approaches," *Computers & Security*, vol. 116, p. 102669, 2022.

[12] Z.-K. Zhang, M. C. Y. Cho, C.-W. Wang, C.-W. Hsu, C.-K. Chen, and S. Shieh, "Iot security: ongoing challenges and research opportunities," in *2014 IEEE 7th international conference on service-oriented computing and applications*. IEEE, 2014, pp. 230–234.

[13] X. Zhu and Y. Badr, "A survey on blockchain-based identity management systems for the internet of things," in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. IEEE, jul 2018.

[14] K. Chen, S. Zhang, Z. Li, Y. Zhang, Q. Deng, S. Ray, and Y. Jin, "Internet-of-things security and vulnerabilities: Taxonomy, challenges, and practice," *Journal of Hardware and Systems Security*, vol. 2, no. 2, pp. 97–110, 2018.

[15] S. Sidhu, B. J. Mohd, and T. Hayajneh, "Hardware security in iot devices with emphasis on hardware trojans," *Journal of Sensor and Actuator Networks*, vol. 8, no. 3, p. 42, 2019. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8761062

[16] M. Roel, "Physically unclonable functions: Constructions, properties and applications," *Katholieke Universiteit Leuven, Belgium*, 2012.

[17] B. Pearson, L. Luo, Y. Zhang, R. Dey, Z. Ling, M. Bassiouni, and X. Fu, "On misconception of hardware and cost in iot security and privacy," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, 2019, pp. 1–7.

[18] I. Butun, A. Sari, and P. Österberg, "Hardware security of fog end-devices for the internet of things," *Sensors*, vol. 20, no. 20, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/20/5729

[19] K. Yang, D. Blaauw, and D. Sylvester, "Hardware designs for security in ultra-low-power IoT systems: An overview and survey," *IEEE Micro*, vol. 37, no. 6, pp. 72–89, nov 2017.

[20] C. Shepherd, G. Arfaoui, I. Gurulian, R. P. Lee, K. Markantonakis, R. N. Akram, D. Sauveron, and E. Conchon, "Secure and trusted execution: Past, present, and future - a critical review in the context of the internet of things and cyber-physical systems," in *2016 IEEE Trustcom/BigDataSE/ISPA*. IEEE, aug 2016, pp. 168–177.

[21] A. Ehret, K. Gettings, B. R. Jordan, and M. A. Kinsy, "A survey on hardware security techniques targeting low-power soc designs," in *2019 IEEE High Performance Extreme Computing Conference (HPEC)*, 2019, pp. 1–8.

[22] W. Hu, C.-H. Chang, A. Sengupta, S. Bhunia, R. Kastner, and H. Li, "An overview of hardware security and trust: Threats, countermeasures, and design tools," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 6, pp. 1010–1038, 2021.

[23] ITU-T, "Y.2720 : Ngn identity management framework," Tech. Rep., 2009. [Online]. Available: https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-Y.2720-200901-I!!PDF-E&type=items

[24] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. CRC Press, dec 2018.

[25] R. Maes, *PUF-Based Entity Identification and Authentication*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 117–141. [Online]. Available: https://doi.org/10.1007/978-3-642-41395-7_5

[26] P. Angin, B. Bhargava, R. Ranchal, N. Singh, M. Linderman, L. B. Othmane, and L. Lilien, "An entity-centric approach for privacy and identity management in cloud computing," in *2010 29th IEEE Symposium on Reliable Distributed Systems*. IEEE, oct 2010.

[27] Y. Cao and L. Yang, "A survey of identity management technology," in *2010 IEEE International Conference on Information Theory and Information Security*. IEEE, dec 2010.

[28] M. Gaedke, J. Meinecke, and M. Nussbaumer, "A modeling approach to federated identity and access management," in *Special Interest Tracks and Posters of the 14th International Conference on World Wide Web*, ser. WWW '05. New York, NY, USA: Association for Computing Machinery, 2005, p. 1156–1157. [Online]. Available: https://doi.org/10.1145/1062745.1062916

[29] D. W. Chadwick, *Federated Identity Management*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 96–120. [Online]. Available: https://doi.org/10.1007/978-3-642-03829-7_3

[30] S. Cantor, J. Moreh, R. Philpott, and E. Maler, "Metadata for the oasis security assertion markup language (saml) v2. 0," 2005.

[31] N. Sakimura, J. Bradley, M. Jones, B. De Medeiros, and C. Mortimore, "Openid connect core 1.0," *The OpenID Foundation*, p. S3, 2014.

[32] D. Divyabharathi and N. G. Cholli, "A review on identity and access management server (keycloak)," *International Journal of Security and Privacy in Pervasive Computing (IJSPPC)*, vol. 12, no. 3, pp. 46–53, 2020.

[33] S. Cantor and T. Scavo, "Shibboleth architecture," *Protocols and Profiles*, vol. 10, p. 16, 2005.

[34] P. R. Sousa, J. S. Resende, R. Martins, and L. Antunes, "The case for blockchain in IoT identity management," *Journal of Enterprise Information Management*, vol. ahead-of-print, no. ahead-of-print, jun 2020.

[35] D. Hardt, "The OAuth 2.0 Authorization Framework," RFC 6749, Oct. 2012. [Online]. Available: https://www.rfc-editor.org/info/rfc6749

[36] A. Jøsang and S. Pope, "User centric identity management," in *AusCERT Asia Pacific information technology security conference*. Citeseer, 2005, p. 77. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.60.1563&rep=rep1&type=pdf

[37] J. Werner, C. M. Westphall, and C. B. Westphall, "Cloud identity management: A survey on privacy strategies," *Computer Networks*, vol. 122, pp. 29–42, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128617301664

[38] S. Y. Lim, P. T. Fotsing, A. Almasri, O. Musa, M. L. M. Kiah, T. F. Ang, and R. Ismail, "Blockchain technology the identity management and authentication service disruptor: a survey," *International Journal on Advanced Science, Engineering and Information Technology*, vol. 8, no. 4-2, pp. 1735–1745, 2018.

[39] A. Mühle, A. Grüner, T. Gayvoronskaya, and C. Meinel, "A survey on essential components of a self-sovereign identity," *Computer Science Review*, vol. 30, pp. 80–86, 2018.

[40] Q. Feng, D. He, S. Zeadally, M. K. Khan, and N. Kumar, "A survey on privacy protection in blockchain system," *Journal of Network and Computer Applications*, vol. 126, pp. 45–58, 2019.

[41] P. Mahalle, S. Babar, N. R. Prasad, and R. Prasad, "Identity management framework towards internet of things (iot): Roadmap and key challenges," in *International Conference on Network Security and Applications*. Springer, 2010, pp. 430–439. [Online]. Available: https://sci-hub.se/https://link.springer.com/chapter/10.1007/978-3-642-14478-3_43

[42] K.-Y. Lam and C.-H. Chi, "Identity in the internet-of-things (iot): New challenges and opportunities," in *International Conference on Information and Communications Security*. Springer International Publishing, 2016, pp. 18–26. [Online]. Available: https://link.springer.com/content/pdf/10.1007/978-3-319-50011-9_2.pdf

[43] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE internet of things journal*, vol. 3, no. 5, pp. 637–646, 2016.

[44] T. Nandy, M. Y. I. B. Idris, R. Md Noor, L. Mat Kiah, L. S. Lun, N. B. Annuar Juma'at, I. Ahmedy, N. Abdul Ghani, and S. Bhattacharyya, "Review on security of internet of things authentication mechanism," *IEEE Access*, vol. 7, pp. 151 054–151 089, 2019.

[45] M. El-hajj, A. Fadlallah, M. Chamoun, and A. Serhrouchni, "A survey of internet of things (iot) authentication schemes," *Sensors*, vol. 19, no. 5, 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/5/1141

[46] M.-O. Pahl and L. Donini, "Giving iot services an identity and changeable attributes," in *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE, 2019, pp. 455–461.

[47] R. Román-Castro, J. López, and S. Gritzalis, "Evolution and trends in iot security," *Computer*, vol. 51, no. 7, pp. 16–25, 2018.

[48] K. Zhao and L. Ge, "A survey on the internet of things security," in *2013 Ninth International Conference on Computational Intelligence and Security*, 2013, pp. 663–667.

[49] R. Roman, P. Najera, and J. Lopez, "Securing the internet of things," *Computer*, vol. 44, no. 9, pp. 51–58, 2011.

[50] H. A. Abdulghani, N. A. Nijdam, A. Collen, and D. Konstantas, "A study on security and privacy guidelines, countermeasures, threats: Iot data at rest perspective," *Symmetry*, vol. 11, no. 6, p. 774, 2019.

[51] M. Katagi, S. Moriai *et al.*, "Lightweight cryptography for the internet of things," *Sony Corporation*, vol. 2008, pp. 7–10, 2008. [Online]. Available: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.227.8445&rep=rep1&type=pdf

[52] Z.-K. Zhang, M. C. Y. Cho, Z.-Y. Wu, and S. W. Shieh, "Identifying and authenticating iot objects in a natural context," *Computer*, vol. 48, no. 08, pp. 81–83, 2015.

[53] R. F. Rights, "Global information assurance certification paper," *GIAC*, 2003.

[54] M. Wolf, *Computers as components: principles of embedded computing system design*. Elsevier, 2012.

[55] C. Gu, *Power On and Bootloader*. Berkeley, CA: Apress, 2016, pp. 5–25. [Online]. Available: https://doi.org/10.1007/978-1-4842-1919-5_2

[56] C. O. Jasper van Woudenberg, *The Hardware Hacking Handbook*. Random House LCC US, Dec. 2021. [Online]. Available: https://www.ebook.de/de/product/31189064/jasper_van_woudenberg_colin_o_flynn_the_hardware_hacking_handbook.html

[57] "Security requirements for cryptographic modules," Tech. Rep., may 2001.

[58] K. Markantonakis *et al.*, "Enhancing the conditional access module security in light of smart card sharing attacks," *Presentation, Information Security Group Smart Card Centre, Royal Holloway, University of London. Accessed at on Oct*, vol. 20, 2008.

[59] C. S. Johnson, M. L. Badger, D. A. Waltermire, J. Snyder, and C. Skorupka, "Guide to cyber threat information sharing," Tech. Rep., oct 2016.

[60] C. L. Smith and D. J. Brooks, "Chapter 3 - security risk management," in *Security Science*, C. L. Smith and D. J. Brooks, Eds. Boston: Butterworth-Heinemann, 2013, pp. 51–80. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780123944368000035

[61] S. P. Skorobogatov, "Semi-invasive attacks: a new approach to hardware security analysis," 2005.

[62] M. T. Rahman, Q. Shi, S. Tajik, H. Shen, D. L. Woodard, M. Tehranipoor, and N. Asadizanjani, "Physical inspection & attacks: New frontier in hardware security," in *2018 IEEE 3rd International Verification and Security Workshop (IVSW)*. IEEE, jul 2018, pp. 93–102.

[63] M. G. Rekoff, "On reverse engineering," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-15, no. 2, pp. 244–252, 1985.

[64] R. Torrance and D. James, "The state-of-the-art in ic reverse engineering," in *Cryptographic Hardware and Embedded Systems - CHES 2009*, C. Clavier and K. Gaj, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 363–381.

[65] R. C. Gilberg, R. M. Knowles, P. Moroney, and W. A. Shumate, "Secure integrated circuit chip with conductive shield," Jun. 12 1990, uS Patent 4,933,898.

[66] S. H. Weingart, "Physical security devices for computer subsystems: A survey of attacks and defenses," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2000, pp. 302–317.

[67] S. Manich, M. S. Wamser, and G. Sigl, "Detection of probing attempts in secure ics," in *2012 IEEE International Symposium on Hardware-Oriented Security and Trust*. IEEE, 2012, pp. 134–139.

[68] M. Nagata, "Exploring fault injection attack resilience of secure ic chips," in *2022 IEEE International Reliability Physics Symposium (IRPS)*. IEEE, 2022, pp. 11C–1.

[69] S. Skorobogatov, "How microprobing can attack encrypted memory," in *2017 Euromicro Conference on Digital System Design (DSD)*. IEEE, 2017, pp. 244–251.

[70] A. Mohammadi, M. Ebrahimi, A. Ejlali, and S. G. Miremadi, "Scfit: A fpga-based fault injection technique for seu fault model," in *2012 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2012, pp. 586–589.

[71] C. O'Flynn, "Getting root on philips hue bridge 2.0," 2016.

[72] N. Timmers, A. Spruyt, and M. Witteman, "Controlling pc on arm using fault injection," in *2016 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*. IEEE, 2016, pp. 25–35.

[73] M. Witteman and M. Oostdijk, "Secure application programming in the presence of side channel attacks," in *RSA conference*, vol. 2008.

[74] S. Endo, Y. Li, N. Homma, K. Sakiyama, K. Ohta, and T. Aoki, "An efficient countermeasure against fault sensitivity analysis using configurable delay blocks," in *2012 Workshop on Fault Diagnosis and Tolerance in Cryptography*. IEEE, 2012, pp. 95–102.

[75] M. Nagata, T. Miki, and N. Miura, "Physical attack protection techniques for ic chip level hardware security," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 30, no. 1, pp. 5–14, 2021.

[76] L. Zussa, A. Dehbaoui, K. Tobich, J.-M. Dutertre, P. Maurine, L. Guillaume-Sage, J. Clediere, and A. Tria, "Efficiency of a glitch detector against electromagnetic fault injection," in *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Mar. 2014, pp. 1–6, iSSN: 1558-1101.

[77] N. Miura, D. Fujimoto, D. Tanaka, Y.-i. Hayashi, N. Homma, T. Aoki, and M. Nagata, "A local EM-analysis attack resistant cryptographic engine with fully-digital oscillator-based tamper-access sensor," in *2014 Symposium on VLSI Circuits Digest of Technical Papers*, Jun. 2014, pp. 1–2, iSSN: 2158-5636.

[78] Y. Araga, M. Nagata, H. Ikeda, T. Miki, N. Miura, N. Watanabe, H. Shimamoto, and K. Kikuchi, "A Thick Cu Layer Buried in Si Interposer Backside for Global Power Routing," *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 9, no. 3, pp. 502–510, Mar. 2019.

[79] S. Bhunia and M. Tehranipoor, "Chapter 8 - side-channel attacks," in *Hardware Security*, S. Bhunia and M. Tehranipoor, Eds. Morgan Kaufmann, 2019, pp. 193–218. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128124772000137

[80] D. Brumley and D. Boneh, "Remote timing attacks are practical," *Computer Networks*, vol. 48, no. 5, pp. 701–716, 2005.

[81] J.-F. Dhem, F. Koeune, P.-A. Leroux, P. Mestré, J.-J. Quisquater, and J.-L. Willems, "A practical implementation of the timing attack," in *International Conference on Smart Card Research and Advanced Applications*. Springer, 1998, pp. 167–182.

[82] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Annual international cryptology conference*. Springer, 1999, pp. 388–397. [Online]. Available: https://link.springer.com/content/pdf/10.1007/3-540-48405-1_25.pdf

[83] E. Ronen, A. Shamir, A.-O. Weingarten, and C. O'Flynn, "Iot goes nuclear: Creating a zigbee chain reaction," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 195–212.

[84] J. Krämer, D. Nedospasov, A. Schlösser, and J.-P. Seifert, "Differential photonic emission analysis," in *Constructive Side-Channel Analysis and Secure Design*, E. Prouff, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 1–16.

[85] A. Schlösser, D. Nedospasov, J. Krämer, S. Orlic, and J.-P. Seifert, "Simple photonic emission analysis of aes," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2012, pp. 41–57.

[86] J. Krämer, "Why cryptography should not rely on physical attack complexity," *it-Information Technology*, vol. 59, no. 1, pp. 53–56, 2017.

[87] O. Kömmerling and M. G. Kuhn, "Design principles for tamper-resistant smartcard processors." *Smartcard*, vol. 99, pp. 9–20, 1999.

[88] V. Rozic, B. Yang, W. Dehaene, and I. Verbauwhede, "Highly efficient entropy extraction for true random number generators on fpgas," in *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 2015, pp. 1–6.

[89] C. S. Petrie and J. A. Connelly, "A noise-based ic random number generator for applications in cryptography," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 47, no. 5, pp. 615–621, 2000.

[90] J. Senden, "Biasing a ring-oscillator based true random number generator with an electro-magnetic fault injuction using harmonic waves," Master's thesis, University of Twente, 2015.

[91] P. Bayon, L. Bossuet, A. Aubert, and V. Fischer, "Electromagnetic analysis on ring oscillator-based true random number generators," in *2013 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2013, pp. 1954–1957.

[92] Y. Su, J. Wu, C. Long, and L. Wei, "Secure decentralized machine identifiers for internet of things," in *Proceedings of the 2020 The 2nd International Conference on Blockchain Technology*, 2020, pp. 57–62.

[93] M. Barr, "Memory types," *Embedded Systems Programming*, vol. 14, no. 5, pp. 103–104, 2001.

[94] U. Gatti, "One-time programmable memories for harsh environments," *Rad-hard Semiconductor Memories*, p. 151, 2019.

[95] C. M. Maxfield, "Chapter 15 - memory ics," in *Bebop to the Boolean Boogie (Third Edition)*, third edition ed., C. M. Maxfield, Ed. Boston: Newnes, 2009, pp. 193–212. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9781856175074000152

[96] D. Kahng and S. M. Sze, "A floating gate and its application to memory devices," *The Bell System Technical Journal*, vol. 46, no. 6, pp. 1288–1295, jul 1967.

[97] C. M. Maxfield, "Chapter 16 - programmable ics," in *Bebop to the Boolean Boogie (Third Edition)*, third edition ed., C. M. Maxfield, Ed. Boston: Newnes, 2009, pp. 213–234. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9781856175074000164

[98] R. F. Rizzolo, T. G. Foote, J. M. Crafts, D. A. Grosch, T. O. Leung, D. J. Lund, B. L. Mechtly, B. J. Robbins, T. J. Slegel, M. J. Tremblay *et al.*, "Ibm system z9 efuse applications and methodology," *IBM Journal of Research and Development*, vol. 51, no. 1.2, pp. 65–75, 2007.

[99] H. Divva, A. P. Chavan, and S. Krishnamurthy, "Design and verification of ecc scheme to optimize area and tester time in otp rom controller," in *2019 4th International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT)*, 2019, pp. 151–155.

[100] J.-M. Schmidt, M. Hutter, and T. Plos, "Optical fault attacks on aes: A threat in violet," in *2009 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*. IEEE, 2009, pp. 13–22.

[101] ——, "Optical fault attacks on aes: A threat in violet," in *2009 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*. IEEE, 2009, pp. 13–22.

[102] M. Hutle and M. Kammerstetter, "Chapter 4 - Resilience Against Physical Attacks," F. Skopik and P. Smith, Eds. Boston: Syngress, Jan. 2015, pp. 79–112. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128021224000043

[103] S. Skorobogatov, "Physical attacks and tamper resistance," in *Introduction to Hardware Security and Trust*. Springer, 2012, pp. 143–173.

[104] M. Tunstall, *Smart Card Security*. Cham: Springer International Publishing, 2017, pp. 217–251. [Online]. Available: https://doi.org/10.1007/978-3-319-50500-8_9

[105] J. Jung, J. Cho, and B. Lee, "A secure platform for iot devices based on arm platform security architecture," in *2020 14th International Conference on Ubiquitous Information Management and Communication (IMCOM)*, 2020, pp. 1–4.

[106] L. Bossuet, M. Grand, L. Gaspar, V. Fischer, and G. Gogniat, "Architectures of flexible symmetric key crypto engines—a survey: From hardware coprocessor to multi-crypto-processor system on chip," *ACM Computing Surveys (CSUR)*, vol. 45, no. 4, pp. 1–32, 2013.

[107] S. Gueron, "Intel advanced encryption standard (aes) instructions set," *Intel White Paper, Rev*, vol. 3, pp. 1–94, 2010.

[108] I. ARM, "Armv8-a architecture reference manual," *URL: https://documentation-service.arm.com/static/60e6f8573d73a34b640e0cee*, 2015.

[109] L. Gaspar, V. Fischer, F. Bernard, L. Bossuet, and P. Cotret, "Hcrypt: a novel concept of crypto-processor with secured key management," in *2010 International Conference on Reconfigurable Computing and FPGAs*. IEEE, 2010, pp. 280–285.

[110] S. A. Rotondo, *Trusted Computing Group*. Boston, MA: Springer US, 2011, pp. 1331–1331. [Online]. Available: https://doi.org/10.1007/978-1-4419-5906-5_498

[111] S. L. Kinney, *Trusted Platform Module Basics: Using TPM in Embedded Systems*. USA: Newnes, 2006.

[112] T. C. Group, "Trusted platform module library part 1: Architecture," Trusted Computing Group, Tech. Rep. 01.59, Nov. 2019. [Online]. Available: https://trustedcomputinggroup.org/wp-content/uploads/TCG_TPM2_r1p59_Part1_Architecture_pub.pdf

[113] K. Murdock, D. Oswald, F. D. Garcia, J. Van Bulck, D. Gruss, and F. Piessens, "Plundervolt: Software-based fault injection attacks against intel sgx," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 1466–1482.

[114] S. Saab, P. Rohatgi, and C. Hampel, "Side-channel protections for cryptographic instruction set extensions," *Cryptology ePrint Archive*, 2016.

[115] Y. Lu, "Attacking hardware aes with dfa," *arXiv preprint arXiv:1902.08693*, 2019.

[116] T. C. Group, "Profile pc client specific trusted platform module tpm family 2.0," Trusted Computing Group, Tech. Rep. 1.3, Sep. 2021.

[117] "Fips 140-3 - security requirements for cryptographic modules," Tech. Rep., apr 2019.

[118] B. Pearson, C. Zou, Y. Zhang, Z. Ling, and X. Fu, "Sic 2: Securing microcontroller based iot devices with low-cost crypto coprocessors," in *2020 IEEE 26th International Conference on Parallel and Distributed Systems (ICPADS)*. IEEE, 2020, pp. 372–381.

[119] Z. Zieliski, J. Chudzikiewicz, and J. Furtak, *An Approach to Integrating Security and Fault Tolerance Mechanisms into the Military IoT*. Cham: Springer International Publishing, 2019, pp. 111–128. [Online]. Available: https://doi.org/10.1007/978-3-030-02807-7_6

[120] R. Toegl, "Tagging the turtle: Local attestation for kiosk computing," in *Advances in Information Security and Assurance*, J. H. Park, H.-H. Chen, M. Atiquzzaman, C. Lee, T.-h. Kim, and S.-S. Yeo, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 60–69.

[121] N. Kuntze, A. Fuchs, and C. Rudolph, "Reliable identities using off-the-shelf hardware security in manets," in *2009 International Conference on Computational Science and Engineering*, vol. 2. IEEE, 2009, pp. 781–786.

[122] G. Inc, "Introduction to secure elements," May 2018. [Online]. Available: https://globalplatform.org/wp-content/uploads/2018/05/Introduction-to-Secure-Element-15May2018.pdf

[123] A. Umar and K. Mayes, *Trusted Execution Environment and Host Card Emulation*. Cham: Springer International Publishing, 2017. [Online]. Available: https://doi.org/10.1007/978-3-319-50500-8_18

[124] B. Lepojević, D. Simić, and A. Radulović, "Architecture of tsm solutions in systems based on nfc technology," 2012.

[125] NXP, "P5cx012/02x/40/73/80/144 family," Jan. 2008.

[126] B. Lepojevic, B. Pavlovic, and A. Radulovic, "Implementing nfc service security–se vs tee vs hce," in *SYMORG Conference*, 2014.

[127] K. Mayes and T. Evans, *Smart Cards and Security for Mobile Communications*. Cham: Springer International Publishing, 2017, pp. 93–128. [Online]. Available: https://doi.org/10.1007/978-3-319-50500-8_4

[128] S. Mangard, E. Oswald, and T. Popp, *Power analysis attacks: Revealing the secrets of smart cards*. Springer Science & Business Media, 2008, vol. 31.

[129] K. E. Mayes and K. Markantonakis, *Smart cards, tokens, security and applications*. Springer, 2008, vol. 1.

[130] V. Lomne, "Common criteria certification of a smartcard: a technical overview," in *CHES*, 2016.

[131] E. B. Sanjuan, I. A. Cardiel, J. A. Cerrada, and C. Cerrada, "Message queuing telemetry transport (mqtt) security: a cryptographic smart card approach," *IEEE Access*, vol. 8, pp. 115 051–115 062, 2020.

[132] Y. Jeon and Y. Kang, "Implementation of a lorawan protocol processing module on an embedded device using secure element," in *2019 34th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC)*, 2019, pp. 1–3.

[133] B. S. S. B.V., "Bosch ip video and data security guidebook," Bosch, techreport 2.0, Apr. 2021. [Online]. Available: https://resources-boschsecurity-cdn.azureedge.net/public/documents/Data_Security_Guideb_Special_enUS_9007221590612491.pdf

[134] C. Lesjak, T. Ruprechter, J. Haid, H. Bock, and E. Brenner, "A secure hardware module and system concept for local and remote industrial embedded system identification," in *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, 2014, pp. 1–7.

[135] C. Lesjak, T. Ruprechter, H. Bock, J. Haid, and E. Brenner, "Estado — enabling smart services for industrial equipment through a secured, transparent and ad-hoc data transmission online," in *The 9th International Conference for Internet Technology and Secured Transactions (ICITST-2014)*, 2014, pp. 171–177.

[136] R. N. Akram, P.-F. Bonnefoi, S. Chaumette, K. Markantonakis, and D. Sauveron, "Improving security of autonomous uavs fleets by using new specific embedded secure elements-a position paper," in *2nd AETOS international conference on "Research challenges for future RPAS/UAV systems", Bordeaux, France*, 2014.

[137] I. GlobalPlatform, "Tee system architecture," GlobalPlatform Technology, techreport GPD_SPE_009, 2018. [Online]. Available: https://globalplatform.org/wp-content/uploads/2018/09/GPD_TEE_SystemArch_v1.1.0.10-for-v1.2_PublicReview.pdf

[138] A. Vasudevan, E. Owusu, Z. Zhou, J. Newsome, and J. M. McCune, "Trustworthy execution on mobile devices: What security properties can my mobile platform give me?" in *International conference on trust and trustworthy computing*. Springer, 2012, pp. 159–178. [Online]. Available: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.220.220&rep=rep1&type=pdf

[139] I. GlobalPlatform, "Trusted user interface api," GlobalPlatform, techreport GPD_SPE_020, Jun. 2013. [Online]. Available: {https://globalplatform.org/wp-content/uploads/2013/06/GlobalPlatform_Trusted_User_Interface_API_v1.0.pdf}

[140] T. Alves, "Trustzone: Integrated hardware and software security," *White paper*, 2004.

[141] V. Costan and S. Devadas, "Intel sgx explained." *IACR Cryptol. ePrint Arch.*, vol. 2016, no. 86, pp. 1–118, 2016. [Online]. Available: http://css.csail.mit.edu/6.858/2020/readings/costan-sgx.pdf

[142] A. Rao, "Rising to the challenge - data security with intel confidential computing," Intel, Feb. 2022. [Online]. Available: https://community.intel.com/t5/Blogs/Products-and-Solutions/Security/Rising-to-the-Challenge-Data-Security-with-Intel-Confidential/post/1353141

[143] M. McReynolds, "Azure announces next generation intel sgx confidential computing vms," Nov. 2021. [Online]. Available: https://techcommunity.microsoft.com/t5/azure-confidential-computing/azure-announces-next-generation-intel-sgx-confidential-computing/ba-p/2839934

[144] S. Pinto and N. Santos, "Demystifying arm trustzone: A comprehensive survey," *ACM Computing Surveys (CSUR)*, vol. 51, no. 6, pp. 1–36, 2019.

[145] H. Yang and M. Lee, "Demystifying arm trustzone tee client api using op-tee," in *The 9th International Conference on Smart Media and Applications*, 2020, pp. 325–328.

[146] T. Firmware, "Open portable trusted execution environment," 2013. [Online]. Available: https://www.op-tee.org/

[147] B. McGillion, T. Dettenborn, T. Nyman, and N. Asokan, "Open-TEE – an open virtual trusted execution environment," in *2015 IEEE Trustcom/BigDataSE/ISPA*. IEEE, aug 2015.

[148] N. Zhang, H. Sun, K. Sun, W. Lou, and Y. T. Hou, "Cachekit: Evading memory introspection using cache incoherence," in *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2016, pp. 337–352.

[149] M. Lipp, D. Gruss, R. Spreitzer, C. Maurice, and S. Mangard, "{ARMageddon}: Cache attacks on mobile devices," in *25th USENIX Security Symposium (USENIX Security 16)*, 2016, pp. 549–564.

[150] R. Guanciale, H. Nemati, C. Baumann, and M. Dam, "Cache storage channels: Alias-driven attacks and verified countermeasures," in *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2016, pp. 38–55.

[151] N. Zhang, K. Sun, D. Shands, W. Lou, and Y. T. Hou, "Truspy: Cache side-channel information leakage from the secure world on arm devices," *Cryptology ePrint Archive*, 2016.

[152] A. Machiry, E. Gustafson, C. Spensky, C. Salls, N. Stephens, R. Wang, A. Bianchi, Y. R. Choe, C. Kruegel, and G. Vigna, "Boomerang: Exploiting the semantic gap in trusted execution environments." in *NDSS*, 2017.

[153] Z. István, P. Rosero, and P. Bonnet, "Always-trusted iot—making iot devices trusted with minimal overhead."

[154] Intel, "linux-sgx," Github, 2015. [Online]. Available: https://github.com/intel/linux-sgx

[155] A. Nilsson, P. N. Bideh, and J. Brorsson, "A survey of published attacks on intel sgx," *arXiv preprint arXiv:2006.13598*, 2020.

[156] A. Brandão, J. S. Resende, and R. Martins, "Hardening cryptographic operations through the use of secure enclaves," *Computers & Security*, vol. 108, p. 102327, 2021.

[157] V. Shanbhogue, J. W. Brandt, and J. Wiedemeier, "Protecting information processing system secrets from debug attacks," Feb. 10 2015, uS Patent 8,955,144.

[158] G. Chen, W. Wang, T. Chen, S. Chen, Y. Zhang, X. Wang, T.-H. Lai, and D. Lin, "Racing in hyperspace: Closing hyper-threading side channels on sgx with contrived data races," in *2018 IEEE Symposium on Security and Privacy (SP)*, 2018, pp. 178–194.

[159] S. Lee, M.-W. Shih, P. Gera, T. Kim, H. Kim, and M. Peinado, "Inferring fine-grained control flow inside {SGX} enclaves with branch shadowing," in *26th USENIX Security Symposium (USENIX Security 17)*, 2017, pp. 557–574.

[160] P. Kocher, J. Horn, A. Fogh, , D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher, M. Schwarz, and Y. Yarom, "Spectre attacks: Exploiting speculative execution," in *40th IEEE Symposium on Security and Privacy (S&P'19)*, 2019.

[161] M. Lipp, M. Schwarz, D. Gruss, T. Prescher, W. Haas, A. Fogh, J. Horn, S. Mangard, P. Kocher, D. Genkin, Y. Yarom, and M. Hamburg, "Meltdown: Reading kernel memory from user space," in *27th USENIX Security Symposium (USENIX Security 18)*, 2018.

[162] G. Chen, S. Chen, Y. Xiao, Y. Zhang, Z. Lin, and T. H. Lai, "Sgxpectre: Stealing intel secrets from sgx enclaves via speculative execution," in *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2019, pp. 142–157.

[163] J. Van Bulck, M. Minkin, O. Weisse, D. Genkin, B. Kasikci, F. Piessens, M. Silberstein, T. F. Wenisch, Y. Yarom, and R. Strackx, "Foreshadow: Extracting the keys to the intel {SGX} kingdom with transient {Out-of-Order} execution," in *27th USENIX Security Symposium (USENIX Security 18)*, 2018, pp. 991–1008.

[164] C. Canella, D. Genkin, L. Giner, D. Gruss, M. Lipp, M. Minkin, D. Moghimi, F. Piessens, M. Schwarz, B. Sunar, J. Van Bulck, and Y. Yarom, "Fallout: Leaking data on meltdown-resistant cpus," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 769–784. [Online]. Available: https://doi.org/10.1145/3319535.3363219

[165] S. Van Schaik, A. Milburn, S. Österlund, P. Frigo, G. Maisuradze, K. Razavi, H. Bos, and C. Giuffrida, "Ridl: Rogue in-flight data load," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 88–105.

[166] M. Schwarz, M. Lipp, D. Moghimi, J. Van Bulck, J. Stecklina, T. Prescher, and D. Gruss, "Zombieload: Cross-privilege-boundary data sampling," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 753–768.

[167] H. Vill, "Sgx attestation process," 2017.

[168] J. Van Bulck, D. Oswald, E. Marin, A. Aldoseri, F. D. Garcia, and F. Piessens, "A tale of two worlds: Assessing the vulnerability of enclave shielding runtimes," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 1741–1758.

[169] NIST, "National vulnerability database," 2022. [Online]. Available: https://nvd.nist.gov/vuln/search/results?form_type=Basic&results_type=overview&query=TrustZone&search_type=all&isCpeNameSearch=false

[170] D. Cerdeira, N. Santos, P. Fonseca, and S. Pinto, "SoK: Understanding the prevailing security vulnerabilities in TrustZone-assisted TEE systems," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, may 2020.

[171] F. Brasser, U. Müller, A. Dmitrienko, K. Kostiainen, S. Capkun, and A.-R. Sadeghi, "Software grand exposure:{SGX} cache attacks are practical," in *11th USENIX Workshop on Offensive Technologies (WOOT 17)*, 2017.

[172] A. Moghimi, G. Irazoqui, and T. Eisenbarth, "Cachezoom: How sgx amplifies the power of cache attacks," in *International Conference on Cryptographic Hardware and Embedded Systems*. Springer, 2017, pp. 69–90.

[173] S. K. Bukasa, R. Lashermes, H. L. Bouder, J.-L. Lanet, and A. Legay, "How trustzone could be bypassed: Side-channel attacks on a modern system-on-chip," in *IFIP International Conference on Information Security Theory and Practice*. Springer, 2017, pp. 93–109.

[174] Z. Chen, G. Vasilakis, K. Murdock, E. Dean, D. Oswald, and F. D. Garcia, "{VoltPillager}: Hardware-based fault injection attacks against intel {SGX} enclaves using the {SVID} voltage scaling interface," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 699–716.

[175] S. Gueron, "A memory encryption engine suitable for general purpose processors," *Cryptology ePrint Archive*, 2016.

[176] C. Lesjak, D. Hein, and J. Winter, "Hardware-security technologies for industrial iot: Trustzone and security controller," in *IECON 2015-41st Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 2015, pp. 002 589–002 595.

[177] Z. Ling, H. Yan, X. Shao, J. Luo, Y. Xu, B. Pearson, and X. Fu, "Secure boot, trusted boot and remote attestation for arm trustzone-based iot nodes," *Journal of Systems Architecture*, vol. 119, p. 102240, 2021.

[178] J. Wang, Z. Hong, Y. Zhang, and Y. Jin, "Enabling security-enhanced attestation with intel sgx for remote terminal and iot," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 1, pp. 88–96, 2018.

[179] A. Durand, P. Gremaud, J. Pasquier, and U. Gerber, "Trusted lightweight communication for iot systems using hardware security," in *Proceedings of the 9th International Conference on the Internet of Things*, 2019, pp. 1–4.

[180] M. Jianhua, Z. Qiaoyan, and H. Guotian, "Authenticity verification scheme based on tee and blockchain," in *2021 18th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*. IEEE, 2021, pp. 141–144.

[181] T. Weingaertner and O. Camenzind, "Identity of things: Applying concepts from self sovereign identity to iot devices," *The Journal of The British Blockchain Association*, p. 21244, 2021.

[182] R. Pappu, B. Recht, J. Taylor, and N. Gershenfeld, "Physical one-way functions," *Science*, vol. 297, no. 5589, pp. 2026–2030, 2002.

[183] P. Tuyls, B. Škorić, S. Stallinga, A. H. Akkermans, and W. Ophey, "Information-theoretic security analysis of physical uncloneable functions," in *International Conference on Financial Cryptography and Data Security*. Springer, 2005, pp. 141–155.

[184] B. Škorić, P. Tuyls, and W. Ophey, "Robust key extraction from physical uncloneable functions," in *International Conference on Applied Cryptography and Network Security*. Springer, 2005, pp. 407–422.

[185] G. A. Fink, D. V. Zarzhitsky, T. E. Carroll, and E. D. Farquhar, "Security and privacy grand challenges for the internet of things," in *2015 International Conference on Collaboration Technologies and Systems (CTS)*. IEEE, 2015, pp. 27–34.

[186] Y. Atwady and M. Hammoudeh, "A survey on authentication techniques for the internet of things," in *Proceedings of the International Conference on Future Networks and Distributed Systems*, ser. ICFNDS '17. New York, NY, USA: Association for Computing Machinery, 2017. [Online]. Available: https://doi.org/10.1145/3102304.3102312

[187] M. Mamdouh, A. I. Awad, A. A. Khalaf, and H. F. Hamed, "Authentication and identity management of ioht devices: Achievements, challenges, and future directions," *Computers & Security*, vol. 111, p. 102491, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404821003151

[188] C. Herder, M.-D. Yu, F. Koushanfar, and S. Devadas, "Physical unclonable functions and applications: A tutorial," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1126–1141, 2014.

[189] H. Kang, Y. Hori, T. Katashita, M. Hagiwara, and K. Iwamura, "Cryptographie key generation from puf data using efficient fuzzy extractors," in *16th International conference on advanced communication technology*. IEEE, 2014, pp. 23–26.

[190] J. Guajardo, S. S. Kumar, G.-J. Schrijen, and P. Tuyls, "Fpga intrinsic pufs and their use for ip protection," in *International workshop on cryptographic hardware and embedded systems*. Springer, 2007, pp. 63–80.

[191] U. Rührmair, H. Busch, and S. Katzenbeisser, "Strong pufs: models, constructions, and security proofs," in *Towards hardware-intrinsic security*. Springer, 2010, pp. 79–96.

[192] U. Rührmair and D. E. Holcomb, "Pufs at a glance," in *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2014, pp. 1–6.

[193] D. Nedospasov, J.-P. Seifert, C. Helfmeier, and C. Boit, "Invasive puf analysis," in *2013 Workshop on Fault Diagnosis and Tolerance in Cryptography*. IEEE, 2013, pp. 30–38.

[194] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber, "Modeling attacks on physical unclonable functions," in *Proceedings of the 17th ACM conference on Computer and communications security*, 2010, pp. 237–249.

[195] G. T. Becker, "On the pitfalls of using arbiter-pufs as building blocks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 8, pp. 1295–1307, 2015.

[196] N. Wisiol, C. Mühl, N. Pirnay, P. H. Nguyen, M. Margraf, J.-P. Seifert, M. van Dijk, and U. Rührmair, "Splitting the interpose puf: A novel modeling attack strategy," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 97–120, 2020.

[197] A. Vijayakumar and S. Kundu, "A novel modeling attack resistant puf design based on non-linear voltage transfer characteristics," in *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2015, pp. 653–658.

[198] A. Mahmoud, U. Rührmair, M. Majzoobi, and F. Koushanfar, "Combined modeling and side channel attacks on strong pufs." *IACR Cryptol. ePrint Arch.*, vol. 2013, p. 632, 2013.

[199] A. Vijayakumar, V. C. Patil, C. B. Prado, and S. Kundu, "Machine learning resistant strong puf: Possible or a pipe dream?" in *2016 IEEE international symposium on hardware oriented security and trust (HOST)*. IEEE, 2016, pp. 19–24.

[200] J. Delvaux, R. Peeters, D. Gu, and I. Verbauwhede, "A survey on lightweight entity authentication with strong PUFs," *ACM Computing Surveys*, vol. 48, no. 2, pp. 1–42, nov 2015.

[201] A. Mahmoud, U. Rührmair, M. Majzoobi, and F. Koushanfar, "Combined modeling and side channel attacks on strong pufs," *Cryptology ePrint Archive*, 2013.

[202] S. Tajik, E. Dietz, S. Frohmann, H. Dittrich, D. Nedospasov, C. Helfmeier, J.-P. Seifert, C. Boit, and H.-W. Hübers, "Photonic side-channel analysis of arbiter pufs," *Journal of Cryptology*, vol. 30, no. 2, pp. 550–571, Apr 2017. [Online]. Available: https://doi.org/10.1007/s00145-016-9228-6

[203] D. Merli, D. Schuster, F. Stumpf, and G. Sigl, "Semi-invasive em attack on fpga ro pufs and countermeasures," in *Proceedings of the Workshop on Embedded Systems Security*, ser. WESS '11. New York, NY, USA: Association for Computing Machinery, 2011. [Online]. Available: https://doi.org/10.1145/2072274.2072276

[204] D. Nedospasov, J.-P. Seifert, C. Helfmeier, and C. Boit, "Invasive puf analysis," in *2013 Workshop on Fault Diagnosis and Tolerance in Cryptography*. IEEE, 2013, pp. 30–38.

[205] A. R. Korenda, F. Afghah, B. Cambou, and C. Philabaum, "A proof of concept SRAM-based physically unclonable function (PUF) key generation mechanism for IoT devices," in *2019 16th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. IEEE, jun 2019.

[206] C. Böhm, M. Hofer, and W. Pribyl, "A microcontroller sram-puf," in *2011 5th International Conference on Network and System Security*. IEEE, 2011, pp. 269–273.

[207] D. Mukhopadhyay, "Pufs as promising tools for security in internet of things," *IEEE Design & Test*, vol. 33, no. 3, pp. 103–115, 2016.

[208] M. A. Qureshi and A. Munir, "PUF-IPA: A PUF-based identity preserving protocol for internet of things authentication," in *2020 IEEE 17th Annual Consumer Communications & Networking Conference (CCNC)*, IEEE. IEEE, jan 2020, pp. 1–7.

[209] K. B. Frikken, M. Blanton, and M. J. Atallah, "Robust authentication using physically unclonable functions," in *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2009, pp. 262–277.

[210] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Silicon physical random functions," in *Proceedings of the 9th ACM conference on Computer and communications security - CCS '02*. ACM Press, 2002.

[211] D. Lim, J. Lee, B. Gassend, G. Suh, M. van Dijk, and S. Devadas, "Extracting secret keys from integrated circuits," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 13, no. 10, pp. 1200–1205, oct 2005.

[212] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber, "Modeling attacks on physical unclonable functions," in *Proceedings of the 17th ACM conference on Computer and communications security*. ACM Press, 2010, pp. 237–249.

[213] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Controlled physical random functions," in *18th Annual Computer Security Applications Conference, 2002. Proceedings.* IEEE Comput. Soc, 2002.

[214] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Lightweight secure PUFs," in *2008 IEEE/ACM International Conference on Computer-Aided Design*. IEEE, nov 2008.

[215] M. Majzoobi, M. Rostami, F. Koushanfar, D. S. Wallach, and S. Devadas, "Slender PUF protocol: A lightweight, robust, and secure authentication by substring matching," in *2012 IEEE Symposium on Security and Privacy Workshops*. IEEE, may 2012.

[216] J. Delvaux and I. Verbauwhede, "Fault injection modeling attacks on 65 nm arbiter and RO sum PUFs via environmental changes," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 6, pp. 1701–1713, jun 2014.

[217] Ü. Kocabaş, A. Peter, S. Katzenbeisser, and A.-R. Sadeghi, "Converse puf-based authentication," in *Trust and Trustworthy Computing*, S. Katzenbeisser, E. Weippl, L. J. Camp, M. Volkamer, M. Reiter, and X. Zhang, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 142–158.

[218] M.-D. Yu, M. Hiller, J. Delvaux, R. Sowell, S. Devadas, and I. Verbauwhede, "A lockdown technique to prevent machine learning on PUFs for lightweight authentication," *IEEE Transactions on Multi-Scale Computing Systems*, vol. 2, no. 3, pp. 146–159, jul 2016.

[219] Y. Gao, H. Ma, S. F. Al-Sarawi, D. Abbott, and D. C. Ranasinghe, "PUF-FSM: A controlled strong PUF," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–1, 2017.

[220] A. Braeken, "Puf based authentication protocol for iot," *Symmetry*, vol. 10, no. 8, p. 352, 2018.

[221] U. Chatterjee, V. Govindan, R. Sadhukhan, D. Mukhopadhyay, R. S. Chakraborty, D. Mahata, and M. M. Prabhu, "Building PUF based authentication and key exchange protocol for IoT without explicit CRPs in verifier database," *IEEE Transactions on Dependable and Secure Computing*, vol. 16, no. 3, pp. 424–437, may 2019.

[222] M. Ebrahimabadi, M. Younis, and N. Karimi, "A PUF-based modeling-attack resilient authentication protocol for IoT devices," *IEEE Internet of Things Journal*, pp. 1–1, 2021.

[223] Z. Huang and Q. Wang, "A puf-based unified identity verification framework for secure iot hardware via device authentication," *World Wide Web*, vol. 23, no. 2, pp. 1057–1088, 2020.

[224] J. Delvaux, D. Gu, D. Schellekens, and I. Verbauwhede, "Helper data algorithms for PUF-based key generation: Overview and analysis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 6, pp. 889–902, jun 2015.

[225] A. Shamsoshoara, A. Korenda, F. Afghah, and S. Zeadally, "A survey on hardware-based security mechanisms for internet of things," *ArXiv.org*, 2019.

[226] B. Škorić, P. Tuyls, and W. Ophey, "Robust key extraction from physical uncloneable functions," pp. 407–422, 2005.

[227] K. Kursawe, A.-R. Sadeghi, D. Schellekens, B. Skoric, and P. Tuyls, "Reconfigurable physical unclonable functions - enabling technology for tamper-resistant storage," in *2009 IEEE International Workshop on Hardware-Oriented Security and Trust*. IEEE, 2009.

[228] G. Suh, C. O'Donnell, I. Sachdev, and S. Devadas, "Design and implementation of the AEGIS single-chip secure processor using physical random functions," in *32nd International Symposium on Computer Architecture (ISCA'05)*. IEEE, 2005.

[229] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," pp. 9–14, 2007. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/4261134

[230] I. Eichhorn, P. Koeberl, and V. van der Leest, "Logically reconfigurable PUFs," in *Proceedings of the sixth ACM workshop on Scalable trusted computing - STC '11*. ACM Press, 2011.

[231] L. Zhang, Z. H. Kong, and C.-H. Chang, "PCKGen: A phase change memory based cryptographic key generator," in *2013 IEEE International Symposium on Circuits and Systems (ISCAS2013)*. IEEE, may 2013.

[232] J. Zhang and G. Qu, "Physical unclonable function-based key sharing via machine learning for IoT security," *IEEE Transactions on Industrial Electronics*, vol. 67, no. 8, pp. 7025–7033, aug 2020.

[233] K. Murdock, D. Oswald, F. D. Garcia, J. Van Bulck, F. Piessens, and D. Gruss, "Plundervolt: How a Little Bit of Undervolting Can Create a Lot of Trouble," *IEEE Security & Privacy*, vol. 18, no. 5, pp. 28–37, Sep. 2020.

[234] C. Tarnovsky, "Attacking tpm part 2 a look at the st19wp18 tpm device," 2013.