Deliverable 5.7

Security software modules: Iteration 1

Start date of the project: 1st January 2023

Duration 48 months

Funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the granting authority can be held responsible for them.

Document Classification

Document Title	D5.7 Security software modules: Iteration 1
Author(s)	P12 – UNIVR – Niccolò Marastoni
Work Package	WP5 – System Architecture Modelling and Simulation
Dissemination Level	PU = Public
Nature	R = Report
Doc ID Code	2024_09_30_NEUROPULS_D5.7
Keywords	Security, attestation, authentication, software,

Document History

2024-06-12	Table of contents, structure defined, abstract and introduction	P12 UNIVR – Niccolò Marastoni
2024-03-19		

Document Validation

Project Coordinator	P1 CNRS – Fabio Pavanello Fabio.pavanello@cnrs.fr
Date	2024-09-30

Document Abstract

This document introduces the software security models that will be implemented in order to protect the neuromorphic accelerator of the NEUROPULS Horizon Europe project (Grant Agreement n° 101070238) from malicious agents. The two main protocols discussed are authentication and static remote software attestation, both of which are described in detail with a particular focus on the most recent advancements found in the literature.

The remote attestation literature review specifically presents papers that have been published in sufficiently reputable venues in the past 5 years, with particular attention given to protocols that are deployed on IoT systems while leveraging physically unclonable functions (PUFs).

The authentication part also focuses on protocols that are based on PUFs and provide enough security while keeping the power requirements low, which is a key aspect of the deployment on IoT devices.

An authentication protocol is proposed after careful consideration, while more advanced protocols for both authentication and remote software attestation will be presented in the second iteration of this deliverable.

Table of contents

1. Introduction 6
2. Background 7
2.1 Attestation 7
2.2 Authentication 8
2.3 Physical Unclonable Function (PUF) 8
2.3.1 Weak PUFs 9
2.3.2 Strong PUFs 10
3. Literature Review 10
3.1 Attestation 10
3.1.1 Brief Description of the Papers 11
3.1.2 PUFs 14
3.1.3 Attestation and PUF synergy 16
3.1.4 Root of Trust 17
3.1.5 Attacks 17
3.1.6 Adversary Assumptions 23
3.1.7 Security Evaluation 24
3.1.8 Performance Evaluation 26
3.2 Authentication 28
4. Proposed Authentication Approach 31
5. Neural network configuration and data encryption 33
6. Conclusions and next steps 34
7. References 35

1. Introduction

With the increasing deployment of neuromorphic accelerators in critical applications, from autonomous vehicles to medical devices, ensuring the security of these systems has become a paramount concern. These devices are often connected to various networks, and can be vulnerable to attacks that may compromise both data privacy and broader network security.
One of the goals in designing the NEUROPULS neuromorphic accelerator is guaranteeing the security of its operation.
This deliverable details the preliminary efforts in designing the software security modules for the neuromorphic accelerator.

An important aspect of the design of these modules is leveraging the intrinsic secure hardware primitives that are built into the neuromorphic accelerator, in particular Physically Unclonable Functions (PUFs). These are hardware modules that provide a unique fingerprint, or response, for each device by leveraging very small inherent manufacturing variations in physical systems. This makes them ideal for cryptographic applications like device authentication, key generation, remote attestation and secure identification.
A more thorough description of PUFs can be found in Section 2.

Organization of the deliverable

1. Section 2 provides a brief background on the primary protocols considered for the security of our devices, namely remote software attestation and authentication. The PUF and its variants are also discussed here.
2. Section 3 serves as a thorough literature review on remote software attestation and authentication. Particularly for attestation, this section provides a very detailed investigation of the most recent state of the art, focusing on works that focus on remote software attestation for IoT devices, leveraging PUFs.
3. Section 4 proposes a novel, lightweight authentication approach that leverages the PUFs installed on the neuromorphic accelerator.
4. Lastly, Section **Error! Reference source not found.**details the next steps of the research into the software security models, aiming to bridge the various gaps found in the existing literature that are detailed in the rest of the deliverable.

2. Background

2.1 Attestation

Remote attestation is an ensemble of techniques that allow for a trusted entity to detect changes in a remote, untrusted device. The goal of a remote attestation algorithm is to verify that the untrusted device has not been altered, thus providing the means to detect tampering attacks.

Remote attestation protocols are well equipped to deal with the problem of having a remote resource (e.g. an IoT device) that needs to be secure but cannot be reliably kept secure by conventional means. This often happens because of the limited computational power of the device, its limited space (i.e. this would prevent the installation of a trusted secure hardware module) or its age and the inability of updating its obsolete hardware components. The protocols generally require two entities: 1. a remote device that needs to prove that it is not compromised, called Prover, and 2. a secure machine that verifies the truthfulness of the Prover, called Verifier.

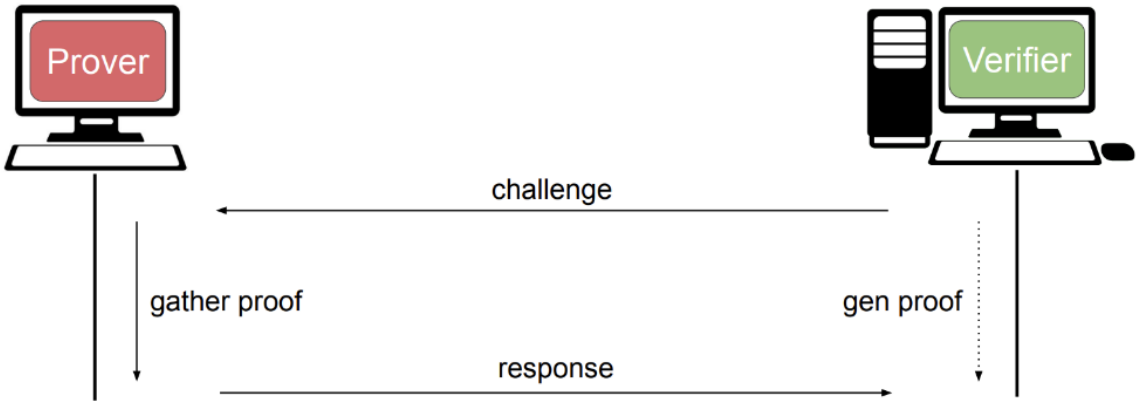

Figure 1: General workflow

A generic workflow can be seen in Figure 1, where the Verifier initiates the protocol by issuing a challenge to the remote device, then the Prover needs to gather enough information about its current state to send back as the response. While this is happening, the Verifier can generate the expected response (or it could fetch it from a database) in order to check its correctness. If the response sent by the Prover corresponds to what was expected by the Verifier, then the attestation protocol can be declared a success, otherwise it either fails or it restarts with a fresh challenge.

The actual protocols in literature can vary a lot in each step of the presented workflow. The challenge can be a simple nonce or it can contain different types of information. Sometimes it can contain the entire attestation program as well [37]. What usually varies the most from protocol to protocol is the information gathered by the Prover in order to

convince of its uncompromised nature, and the methods with which this information is gathered. As an example, the memory contents of the Prover is a common piece of data that can be sent to the Verifier to guarantee the absence of malicious adversaries. It is possible to send the checksum or hash of the entire memory dump (for very small devices) [42], which can attest the whole device, or just certain memory regions, which will in turn attest a specific software running on the device.

2.2 Authentication

Authentication refers to the concept of asserting the identity of a user or a device. Usually, a central entity, a server or a verifier, has the task to challenge clients wishing to connect, so that only authorized users or devices are allowed to access sensitive resources. Authentication can be *mutual*, i.e., the client may wish to verify the identity of the server as well.

Key distribution is another concept often associated with authentication. Once two parties authenticate each other, they might agree on a common key to use for encryption of subsequent communication.

Authentication protocols have been developed in order to establish a standard way of communication between two parties. Since the advent of the first computer networks, plenty of authentication protocols have been developed. Each protocol has its balance of security and performance.

Authentication is often based on a secret known by both parties, and so each protocol uses some sort of cryptography to manage secret information. Security and performance of high-level protocols depend on which underlying cryptographic primitives are used, and how they are combined.

2.3 Physical Unclonable Function (PUF)

Physical Unclonable Functions (PUFs) stand at the forefront of hardware-based security, providing a foundational layer characterized by ease of manufacture and inherent resistance to replication. This unique attribute positions PUFs as ideal candidates for generating and safeguarding cryptographic keys, particularly in device authentication. Operating through a Challenge-Response Pair (CRP) mechanism, PUFs introduce a robust process involving two crucial stages: enrollment and verification.

In the enrollment stage, the distinctive physical characteristics of a device undergo capture and transformation into a digital representation. This intricate process exposes the device to challenges manifested as specific input signals or stimuli. The device

responds to these challenges by leveraging its intrinsic physical properties, generating responses that collectively form the CRP.

The captured responses are a digital fingerprint, giving each device a unique identity. This identity arises from inherent variations and imperfections within the physical components, rendering accurate replication nearly impossible. The primary objective of the enrollment stage is to establish a reliable and distinct signature for each device, ensuring that no two devices share identical CRPs.

The verification stage employs the stored CRP to confirm the device's identity in subsequent interactions. The device responds when presented with a new challenge, and the generated output is compared to the stored CRP. A successful match between the response and the digital fingerprint affirms the device's identity, granting authentication. This verification process is a crucial security measure, permitting access only to legitimate devices with the correct physical characteristics. Any attempts to clone the device or employ fraudulent responses lead to mismatches during verification, thereby fortifying the system's overall security.

Beyond the enrollment and verification stages, PUFs exhibit diverse classifications based on material fabrication and security attributes. Material categorizations include distinctions such as silicon and non-silicon, electronic and non-electronic. Security classifications delineate PUFs into categories such as Weak and Strong PUFs. The NEUROPULS project specifically focuses on photonic PUFs, which rely on the random splitting of a laser beam interacting with multiple resonant devices.

The NEUROPULS project aims to leverage the same technology used for the accelerator and the photonic PUF. In particular, the photonic PUF will be coupled with the accelerator to provide a signature also of the accelerator. The utilization of PUFs created from silicon photonics has demonstrated efficacy, as evidenced by diverse studies and author contributions (1) (2).

The application of photonic PUFs, especially within silicon photonics, showcases promising results and aligns with ongoing research endeavors, such as the innovative NEUROPULS project. Two different PUFs are available in the literature, and the following subparagraph will describe them to better grasp their peculiarities.

2.3.1 Weak PUFs

A Weak PUF is characterized by a limited number of challenges, often as few as one fixed challenge. In contrast to Strong PUFs, which prioritize intricate challenge-response behavior, Weak PUFs utilize their limited set of challenges to derive a classical binary secret key. While Weak PUFs may be advantageous regarding invasiveness and individualization during production, their susceptibility to side-channel attacks, such as power consumption or emanation analysis, poses a challenge. Unlike Strong PUFs, they lack complexity and resistance to numerical prediction, relying on error correction for their limited challenges. Examples include the SRAM PUF, Butterfly PUF, and Coating PUF, each tailored for specific cryptographic applications focusing on simplicity in key

derivation. In essence, weak PUFs serve as a specialized form of secret key storage suitable for various cryptographic schemes but necessitates considering their susceptibility to certain attacks.

2.3.2 Strong PUFs

Strong PUFs are designed to provide heightened security in the face of potential adversarial threats. PUFs, which generate unique digital fingerprints through CRPs, serve as identifiers for electronic devices. Strong PUFs set themselves apart by exhibiting a complex and intricate relationship between challenges and responses, creating a formidable barrier for adversaries attempting to predict or imitate their behavior. The requirement for a vast and practically unmanageable number of possible challenges adds a layer of security, making it computationally infeasible for attackers to determine all challenge-response pairs within a limited timeframe. Strong PUFs resist cloning by ensuring that responses are extremely difficult to replicate, leveraging manufacturing variations and structural disorders beyond the control of the original manufacturer. Notably, the challenge-response behavior of Strong PUFs, exemplified by the Optical PUF using random optical reflection patterns, finds application in key establishment and identification protocols. The inherent complexity of Strong PUFs makes numerical prediction challenging for adversaries, enhancing the unpredictability and overall security of these specialized devices. In summary, Strong PUFs play a crucial role in fortifying the security of electronic devices by incorporating advanced features that resist adversarial attempts to compromise their identification mechanisms.

3. Literature Review

3.1 Attestation

In this section we explore 14 papers on remote attestation, highlighting their position in the research space by noting if and how they deal with some arguably important aspects of this research field. We have thus summarized how the selected papers can be categorized according to what type of PUF is used and how it is used, the definition of the root of trust, the attacks that are mentioned or the adversary model, and various types of evaluation, from security to computational complexity.

3.1.1 Brief Description of the Papers

Secure Boot and Remote Attestation in the Sanctum Processor (3)

This paper proposes an attested execution processor that derives its cryptographic identity from manufacturing variation measured by a PUF, eliminating the need for non-volatile memory or explicitly assigned private keys. A trapdoor computational fuzzy extractor ensures the reliability and security of PUF keys. Detailed evaluation results are provided for both secure boot and remote attestation. The main claim of this paper is providing the first implementation of a PUF that can generate a random seed inside a secure processor.

SACHa: Self-Attestation of Configurable Hardware (4)

This paper proposes a solution where an FPGA-based prover core attests the entire FPGA, including self-attestation, serving as a tamper-resistant module. This enables hardware-based attestation of a processor, safeguarding the hardware/software system from malicious code updates. From the title and the abstract this does not seem to fit in the field of remote attestation, but the description of the protocol in the paper makes it clear that it does. For example, the verifier (Vrf) and the prover (Prv) communicate with each other via a public channel and Vrf is an entity not limited in processing power, e.g. a laptop, while Prv is an FPGA.

PAtt: Physics-based Attestation of Control Systems (5)

PLCs often lack hardware support for techniques like remote attestation, which can verify logic code integrity. Additionally, existing remote attestation methods do not verify the integrity of the physical process controlled by the PLC. For this reason, the authors introduce PAtt, a system that combines remote software attestation with control process validation. PAtt utilizes operation permutations, subtle variations in operation sequences based on integrity measurements, to generate unique sensor readings during execution without affecting the physical process. This results in a novel PUF design that is based solely on physical processes. By incorporating integrity measurements into control operations, the system enables remote verification of the control logic's integrity using the resulting sensor data.

Defining Trust in IoT Environments via Distributed Remote Attestation using Blockchain (6)

In this work, a remote attestation protocol is introduced, utilizing blockchain technology to establish trust among IoT devices. The blockchain provides a secure framework for device registration, while in turn the attestation process relies on Physical Unclonable Functions (PUFs). Combining these technologies generates a tamper-resistant scheme, offering protection against physical and proxy attacks. This proposal aims to enhance security and integrity by leveraging blockchain and PUF-based attestation for IoT device trust establishment.

SGX-FPGA: Trusted Execution Environment for CPU-FPGA Heterogeneous Architecture (7)

This work introduces SGXFPGA, which establishes a trusted hardware isolation path and enables the first FPGA TEE. Trusted execution environments (TEEs), such as Intel SGX, are widely used for their minimal trusted computing base (TCB) and reduced attack surface, but current CPU-based TEEs do not support FPGAs. This is not ideal, given the rapid deployment of FPGA-based cloud computing services that still exhibit security vulnerabilities, hence the current proposal. SGXFPGA connects SGX enclaves and FPGAs in a heterogeneous CPU-FPGA architecture.

ATT-Auth: A Hybrid Protocol for Industrial IoT Attestation With Authentication (8)

This research paper focuses on developing attestation techniques specifically for Industrial Internet of Things (IIoT) systems. The proposed attestation protocols use PUFs to ensure hardware security and avoid physical attacks. The protocols also adopt a novel approach where the verifier does not calculate the reference checksum for each prover, instead, timing information is utilized. The proposed protocols achieve scalability by using timing information, instead of having the verifier calculate the reference checksums for every prover. This allows for efficient attestation of multiple devices in large-scale networks, with a high probability of detecting malware and reduced computation overhead.

AAoT: Lightweight attestation and authentication of low-resource things in IoT and CPS (9)

This paper addresses the increasing security and privacy risks associated with smart embedded devices in the context of IoT and CPS. It focuses on enabling remote attestation and authentication for low-resource embedded devices with AAoT, a mechanism that provides software integrity, mutual authentication, and tamper-proof features. AAoT utilizes PUFs, random memory filling, and software attestation without requiring changes to existing micro-controller units (MCUs). Efficient implementations and optimizations for each component of AAoT are demonstrated, including PUF-based memory filling, a checksum function, a pseudorandom function, a reverse fuzzy extractor, and a random number generator.

Design, Analysis and Application of Embedded Resistive RAM Based Strong Arbiter PUF (10)

This work focuses on Resistive Random Access Memory (RRAM)-based PUF designs, enabling the production of a strong arbiter PUF which uses a 1T-1R bit cell, designed with minimal changes to conventional RRAM memory arrays. The PUF utilizes repurposed components, such as the voltage sense amplifier, address, and data lines. The proposed PUF architecture is evaluated for uniqueness, uniformity, and reliability across various stages. It demonstrates

satisfactory performance in terms of intra-die Hamming Distance (HD) and inter-die HD, passing the NIST tests. The authors assess its vulnerability to machine learning attacks and showcase its application for data attestation in the Internet of Things. The proposed PUF-based data attestation offers low energy consumption and high-speed performance.

HAtt: Hybrid Remote Attestation for the Internet of Things With High Availability (11)

This work introduces a remote attestation protocol called hybrid remote attestation to address the cybersecurity concerns of IoT devices. The proposed protocol ensures high availability during the software attestation process. It utilizes a randomized approach to attest different parts of an IoT device's memory and employs PUFs to protect device secrets from physical attacks. A security analysis confirms the effectiveness of the proposed protocol in detecting roaming malware while an implementation on Raspberry Pi and AVR/ARM-based ATMEl microcontrollers, along with a comparison to existing techniques, demonstrates improved availability and reduced energy consumption.

A Lightweight Authentication and Attestation Scheme for In-Transit Vehicles in IoV Scenario (12)

This paper proposes a lightweight and secure authentication and attestation scheme for vehicles while they are on the road. The scheme includes both vehicle authentication with Road Side Units (RSUs) and attestation of ECU firmware from edge servers connected to RSUs. Security and performance analyses are conducted, comparing the proposed protocol with existing ones, demonstrating its feasibility for deployment.

Scalable Attestation Protocol Resilient to Physical Attacks for IoT Environments (13)

This article proposes a lightweight attestation protocol for IoT systems, leveraging an ideal environment with PUFs. The protocol ensures resilience against strong adversaries who physically access IoT devices, while experimental results demonstrate the scalability of the protocol and its suitability for dynamic networks. This is the first work that deals with swarm or collective attestation.

IoT-Proctor: A Secure and Lightweight Device Patching Framework for Mitigating Malware Spread in IoT Networks (14)

This work presents a secure patching framework for IoT networks to control and mitigate malware spread. Traditional schemes are ineffective due to device-to-device propagation and the scale of IoT devices. The framework uses remote attestation and virtual patching with PUFs to detect compromised devices that contain malware. It employs different network isolation levels based on the SEIR model for access control. Security and performance analyses confirm the framework's effectiveness, achieving faster malware reduction compared to existing techniques.

RPRIA: Reputation and PUF-Based Remote Identity Attestation Protocol for Massive IoT Devices (15)

This article presents a remote identity attestation protocol for IoT devices in smart cities. The protocol utilizes reputation mechanisms and PUFs to ensure efficient and secure mutual authentication and key agreement. The security of the approach is formally proven using the Burrows-Abadi-Needham (BAN) logic and Scyther tool, while empirical performance evaluation demonstrates the protocol's favorable security and efficiency compared to other protocols.

A lightweight remote attestation using PUFs and hash-based signatures for low-end IoT devices (16)

This work proposes a low-cost Root of Trust for Measuring and Reporting (RoTMR) in IoT devices to enable lightweight and secure remote attestation. The RoTMR combines a PUF and an Attestation Read-Only Memory (A-ROM), with hash-based digital signatures used in the attestation protocol. The PUF reconstructs secret keys, ensuring they are not stored, while the A-ROM contains unalterable attestation instructions executed sequentially. The solution is quantum-resistant and robust due to the unidirectionality of a hash function. The proposal utilizes One-Time Signature (OTS) and Many-Time Signature (MTS) schemes, well-suited for resource-constrained devices. Experimental validation with the ESP32 microcontroller demonstrates the effectiveness of the proposal, with OTS schemes requiring smaller code and faster execution times compared to ECDSA. OTS schemes also offer efficient communication bandwidth due to their small signatures.

3.1.2 PUFs

As mentioned in Section 2.3, PUFs can be easily divided into weak PUFs and strong PUFs, but their implementation details can vary a lot more. It is possible to obtain a PUF with different hardware components, such as an SRAM (17), a Ring Oscillator (18) physical processes (19) (5) and various more. The way these PUFs are engineered also leads to further classifications, where the type of PUF dictates how the PUF is constructed, e.g. Arbiter PUF (20) or XOR PUF (21). All of these offer specific advantages and drawbacks, hence why it is often important to specify which type of PUF has been used in a protocol.

Types

The PUFs encountered while compiling this survey have been classified with regard to their novelty and type. Four papers included in this survey provide a novel PUF design and thus have described their PUF architecture in detail.

Lebedev et al. (3) propose two PUF designs, P256 and P512, that leverage an array of Ring Oscillators and a trapdoor fuzzy extractor to safely generate the CRPs. They employ a

design by Herder et al. (22) to build a trapdoor LPN PUF with an array of identical ring oscillator pairs, which is then implemented on a commercial FPGA for evaluation.

Ghaeini et al. (5) introduce a novel PUF design that uses the physical processes typical of Control Systems in order to have a reliable source of entropy able to uniquely characterize each machine. This is described in the paper as a "PUF-like" use of physical processes, as it avoids the usual sources of entropy for PUFs, such as SRAM, RRAM or Ring Oscillators. The PUF design is less detailed than others in this section mainly due to the different mechanism that generates the characteristic PUF footprint. An interesting note is that this particular type of physical PUF is less prone to aging effects (none have been observed), which are very prominent in classic PUF designs due to physical effects in the resistors (23), (24).

Govindaraj et al. (10) provide perhaps the most detailed PUF design description in this survey (alongside (3)), proposing an Arbiter PUF based on Resistive Random Access Memory (RRAM), claiming that it is a promising candidate for future implementation of a non-volatile memory unit. This work also provides a good comparison with Ring Oscillator based PUFs, another very common design choice that incurs in additional area overhead compared to RRAM. The general design of the PUF follows closely the already mentioned Arbiter PUF.

Feng et al. (9) implement an SRAM PUF and provide details about its construction along with its robustness, uniqueness and randomness. The chosen type of PUF is weak, due to its inherent resilience against Machine Learning attacks, even if their protocol can accommodate both strong and weak PUFs, as only 2 CRPs are needed.

A weak PUF based on SRAM is used in (9) and (16), where the weakness of the construction is never mentioned but can be inferred by thoroughly reading some referenced papers, in particular (25).

The Arbiter PUF, a fairly common PUF design that accommodates the need for many CRPs, has been used in (7) and (10) (as described above), while (12) uses a tamperproof PUF based on RRAM that was first described in (26).

Two papers avoid choosing a specific PUF, based on the assumption that their approach requires an "ideal" PUF, which is described as a PUF without any weaknesses (4), (11).

In the rest of the papers there is a similar lack of proper PUF definition or description, although it is possible to infer that a strong PUF might be needed for two of these works (14), (15), while a weak PUF should be sufficient for (6), (8), since they require significantly less CRPs than their counterparts.

Advantages

Only three papers (7), (13), (8) explicitly state the advantages of using PUFs in their attestation protocol. According to (7), the hardware nature of the PUF reduces the overhead that is often encountered when generating and storing keys, since strong

PUFs can inherently generate large sets of CRPs. On a similar note, (13) asserts that the attractive features of PUFs are their lightweight requirements while achieving high throughputs. The addition of a PUF also does not increase the overhead for the attestation protocol. In (8) the focus is on the very low cost of production of PUFs, alongside maintaining a significantly low silicon area.

Less explicitly, the authors in (6) state that they introduce PUFs to an existing protocol, SWATT, in order to mitigate its inherent weakness against physical attacks, which can be considered an advantage intrinsic to PUFs.

3.1.3 Attestation and PUF synergy

Attestation protocols are harder to fit into a set taxonomy, as they are usually built incrementally on previous versions with specific improvements. An interesting thing to note in the works we reviewed is how the PUF is used in the system and how (or if) it synergizes with the attestation protocol. PUFs can be used at different stages of the attestation process, but they can also be used for separate phases of a security system that includes attestation (e.g. authentication).

The PUF is specifically identified as the root of trust of the attestation protocol in (3), (7), (6), (8). The same can be said of (4), since the PUF is used to generate the cryptographic key used for the MAC that will deliver the attestation payload. In (5), the PUF is based on physical signals from the actuators, and it is used directly for the attestation.

A more interesting use of the PUF can be found in (9), (8), (13), where the PUF is used in each iteration of the checksum function, inherently coupling it with the attestation protocol.

The PUF is used for attestation in (14) (since they mainly leverage HAtt (11)), but also for authorization and, more interestingly, for secure patching.

The PUF in (16) is combined with an Attestation Read-Only Memory to generate the actual root of trust for attestation, while (15) uses it embedded in the identity attestation protocol, making it an implicit root of trust. Similarly, the PUF in (10) is used to encrypt data in a data attestation protocol.

The only work where the PUF is not explicitly part of the attestation algorithm is (12), where it is used mainly to generate an authenticated communication channel. It can be argued that the secure channel is then used to exchange the messages in the attestation protocol, so it is parallel to the previous cited works.

3.1.4 Root of Trust

Only 7 of the examined papers mention a root of trust at all and only 3 of them explain in detail how the root of trust of their system is established (3), (9), (16).

RoT details

In (3) there are multiple Roots of Trust described, the first of which refers to the first-stage bootloader, which is a trusted program responsible for loading a payload binary segment from untrusted storage into secure system memory. Then the two implemented PUFs, P256 and P512, also act as part of the Root of Trust. Interestingly, this is also the only paper that evaluates the root of trust (3), showing how the implementation of the root of trust affects the size of the code and its latency.

The PUF combined with an Attestation PUF is employed in (16) in order to establish a Root of Trust for Measuring and Reporting. The authors argue that the usual methods of establishing a Root of Trust, such as a Trusted Platform Module, are too expensive and not available to most IoT devices, while PUFs can be easily embedded in most devices.

A root of trust for low-end embedded devices is proposed in (9), where the PUF is used to generate a PUF-based Root of Trust (PUFRoT). PUFRoT is a firmware that is able to safely measure the integrity and authenticity of the device, while also delivering reports to the verifier.

PUF as RoT

The PUF itself is also explicitly identified as part of the root of trust in (6), (8), (7), without further details.

In other works the root of trust is never mentioned, but the PUF is mainly used as a quasi-reliable source of entropy for key generation, sometimes becoming an implicit root of trust.

3.1.5 Attacks

Attestation protocols generally protect against tampering attacks, as the prover needs to be uncompromised in order for the verifier to accept the attestation attempt. That being said, there is a host of possible attacks that can be waged against systems that employ attestation measures. In this section we collect and present all the attacks explicitly mentioned in the paper.

Impersonation attack

In this scenario, the adversary wants to convince the verifier that it is in fact an uncompromised prover, thus impersonating an IoT device that has not been attacked.

It is possible also for the attacker to pretend to be the verifier, in order to initiate an attestation loop that can be used to replay the attestation payload for a future run of the protocol. In (4), (11) the PUF is part of the main mechanism that thwarts impersonation attacks, as PUFs offer the verifier an easy way to check if they are in fact communicating with the prover, thanks to their properties of unclonability and tamper evidence. In a similar way (12) proposes a method that embeds the PUF response with the private ID of the vehicle, meaning that only a legitimate vehicle is able to verify the generated message digest, effectively foiling any impersonation attempt. The PUF in (9) is used for authentication but also inside the checksum, in order to verify that the attestation result has been in fact computed on the right machine, effectively another type of impersonation attack. In (14) the mechanism is derived from HAtt (11) and it is proven theoretically that an impersonation attack cannot happen as long as the PUF is not compromised (e.g. due to a modeling attack). Two papers, (13) and (15) (here it is referred to as "fake identity") only mention the attack as a way in which attackers can affect negatively an attestation protocol.

The Man-In-The-Middle (MITM) attack is only mentioned in (12), (8). The attack is included here because it is not very commonly treated and, in order for a successful MITM attack to take place, the adversary has to impersonate both parties in a communication instance.

The reason most papers list the impersonation attack as one of the main ones is possibly because PUFs are often used as a source of entropy for keys used in the authentication part of the protocol, which is usually what blocks impersonation attacks.

Malware

Malware is not detected or contained in (4), rather the system is designed in a way that there will be no malicious code after an update. The system also ensures that no computing devices can connect to the FPGA and run malicious code.

The reduction of the spread of malware is one of the main goals of (14). In this paper, a PUF-based patching mechanism is presented as a solution to unbounded malware spread, as unpatched devices tend to be the most vulnerable to malicious code attacks. If the patching mechanism fails, a fail-safe might be activated, rendering the device unable to communicate with the outside, effectively blocking the malware spread.

In (6), (8), (9), (12), (13), malware is one of the attack vectors that the attackers could realistically implement in order to inflict damage, and that the attestation protocol aims to avoid. Similar to the mechanism engineered by (14), if a device is found to contain malware in (6), it is completely dropped from the network, rendering the malware impotent.

Roving malware is a type of malicious code that can relocate itself in memory or delete itself to come back at a future time, in order to avoid detection. This is the main concern of the work presented in (11), as it is a particularly difficult malware to detect with common techniques.

Replay attack

This attack involves the adversary capturing an existing attestation payload (i.e. from a previous successful attestation protocol run) and re-using it from a corrupted prover in order to fool the verifier. It is generally possible to foil this attack by including a nonce to the attestation challenge and subsequent response, in order to make sure that the received message is fresh. This is the approach used in (4), (11), (12), (15). In (4) the nonce is employed alongside a failsafe mechanism that also bounds the MAC computation to the order in which the verifier reads the configuration frames.

An interesting approach to replay attack detection is employed in (5), where the system can detect a replayed sensor reading by analyzing the sensor traces and comparing them to expected patterns. This can be done through either statistical analysis, signal processing techniques, or machine learning algorithms.

We believe that the proxy attack mentioned in (6) falls under the description of replay attack. It is described as an attack where the adversary has control of a proxy node with legitimate software, and thus can be used to carry out the attestation instead of the compromised device. This is an example of a replay attack where the replayed payload is not an old and reused message, but a fresh one. The

attack is foiled by the protocol described in the paper by requiring the PUF to be called in the checksum function, thus rendering the proxy attestation useless.

Similarly, the collusion attack is only mentioned in (9), but its description seems to fit the replay label, since it involves using a valid attestation payload from an uncompromised device in order to escape the protocol. The solution is also similar to the above, as the PUF is bound to the checksum function, effectively blocking any external device from computing the same payload.

The papers (13), (14) only mention the replay attack as one of the attacks that could be waged against the system, without specifying how it can be dealt with.

Physical attack

Physical attacks are very common in IoT and encompass any physical modification done to a system in order to achieve an adversary goal.

The addition of a malicious hardware module to multiple parts of the system is contemplated in (4), where a partial reconfiguration and configuration memory readback is used by the architecture to avoid such cases.

PUFs are touted as a way to avoid certain physical attacks in (11), (8) and (14), as they remove the need to store keys in the device's memory, thus eliminating it as a vector for attack.

The tamper evidence property of PUFs is used in (6) to enhance the security of SWATT (27), since the PUF will be rendered useless after physical tampering.

Cloning attack

Many devices can be cloned in order to create a perfect copy that can aid in performing malicious behaviors. PUFs are used in (8), (11), (16), (13) as an obvious foil to cloning attacks, as they are unclonable by design.

Side-channel attacks

Side-channel attacks can analyze the correlation between dynamic power consumption and the CRPs to extract information about the PUF's behavior. In (10) the proposed APUF design aims to foil side channel attacks by minimizing the correlation between the responses generated by the PUF and the power consumption of the circuit. This is done by introducing variations in the path delays, making it difficult to infer information about the responses based on power consumption.

The fact that the challenges and responses generated from the PUF are never exposed to the network is given as a reason for side-channel attack resistance in (12). The protocol described in (13) aims to address side-channel attacks but it is never explained how, or if, it does so.

Tampering

With this attack we do not consider physical tampering of the devices, as that is covered in the previous "physical attacks" paragraph. The tampering of the blockchain is theorized in (6), where the security against such

attack is guaranteed by the protocol, since it is tamper-proof if the number of malicious miners fails to exceed the number of honest miners.

In (11) tampering can happen in the packets exchanged during the attestation protocol, but it is foiled by the device which can verify that the packets are intact.

Data tampering is also considered in (14), where it is proven that the only way for an adversary to actually tamper with the messages exchanged in the protocol is by having direct access to the PUF (which is unfeasible in their scenario).

Denial Of Service attacks

Denial of Service (DoS) is a class of attacks that can affect most devices that connect to a public network, as it consists of overwhelming the system resources with enough requests that its functioning either slows to a crawl or stops altogether. Detecting and protecting against DoS attacks is not trivial as it is often hard to distinguish between legitimate heavy network traffic and an actual attack (28). Remote attestation protocols

for IoT can be affected by these attacks, as many of these protocols rely on timing to determine whether a prover is compromised.

Requiring the verifier to be authenticated with the prover prevents DoS attacks according to (9). A similar approach is adopted in (13), where the prover is uniquely identified through an ID that then allows the verifier to call an emergency function to block the attack.

DoS attacks are only mitigated to a certain extent in (15) thanks to its reputation mechanism, since messages coming from devices with low reputation are not processed.

Machine Learning (modeling) attacks

PUFs themselves can be prone to attacks, most importantly those utilizing Machine Learning models in order to create a reliable model of the PUF with a few CRPs. These attacks are particularly easy to wage against PUFs where the challenges are highly correlated to the responses (29), which is why many PUF constructions involve the addition of non-linearities to the circuit.

Weak PUFs are inherently resistant to modeling attacks since the space of CRPs is too small, which is one of the reasons they are chosen in (9).

In order to foil a modeling attack against strong PUFs in (12), the CRPs should not be exposed to the network. This prevents an adversary from building a training set that can be used to create a model of the PUF.

In (10), resistance to modeling attacks is one of the main driving factors in the PUF design. The way modeling attacks are dealt with is by incorporating XOR operations between multiple Arbiter PUF instances. XOR is a famously hard function to deal with in machine learning and this led to the design of many XOR-based PUFs, even if the security of these is all but guaranteed (30).

Memory attacks

A memory attack can be characterized as any attack that requires the manipulation of the device's memory in order to perform a malicious activity or avoid detection. This was already explored in (11), as memory relocation is one of the common mechanisms with which roving malware avoid detection.

In (9), memory attacks are dealt with by requiring the attestation protocol to use a PUF-based seed in order to traverse the memory. Therefore, an attacker would not be able to perform memory copy or memory compression attacks without first compromising the PUF itself. A similar consideration is cited in (8), since the presented protocol is an implementation of SWATT, which inherently randomizes memory access through a secret random seed, rendering memory attacks impossible.

Specific attacks

Some attacks are rarely discussed and appear only in one paper, and for this reason they are shown separately. These range from attacks that might already be discussed in previous works but lack proper naming to very specific types of attacks that are only considered due to a particular slant of the work itself. This is the case for the CPU-TO-CPU, FPGA-TO-CPU and CPU-TO-FPGA attacks mentioned in (7). These attacks only make sense in this paper, as the attestation protocol proposed identifies the CPU and the FPGA as the usual prover-verifier roles commonly found in works of this field. On a similar note, the hash approximation attack only appears in (5), since this is the only work that focuses on actuation sequences of PLC systems that are encoded into hashes.

Sanctum (3) provides protection against a specific type of software attacks that extract private information by analyzing a program's memory access patterns while additionally providing protection against subtle side channel attacks that exploit the cache state and shared page tables.

Every attestation algorithm can intuitively suffer from an attack where the adversary simply evades the protocol in order to carry out their malicious activity. However, the attestation evasion attack is only mentioned in (14), where their protocol is said to be safe against such attack due to the HAtt protocol's (11) inherent resistance to evasion.

Injecting packets in a network can be considered an attack in many scenarios, such as MITM or DoS, but it is considered an attack in and of itself only in (14).

The counterfeit attack is mentioned only in (15) and might be a mislabeled impersonation attack, but the paper lacks any description of it or any method to deal with it. On a similar note, the reflection attack is only mentioned in (13), but without a proper description of such attack it cannot be included in any of the previous labels.

Out of scope attacks

Few papers outright declare attacks to be out of scope. In (4), the PUFs adopted are assumed to be ideal, thus PUF attacks are explicitly declared out of scope. Runtime attacks, control-flow attacks and physical attacks are considered out of scope for (9). Only non-invasive attacks are considered in (13), meaning that all invasive attacks are implicitly out of scope.

No specific attacks have been mentioned for (16), but the possibility of the adversary performing sophisticated hardware attacks (i.e. as fault injection) is ruled out, since these could allow the modification of the Program Counter.

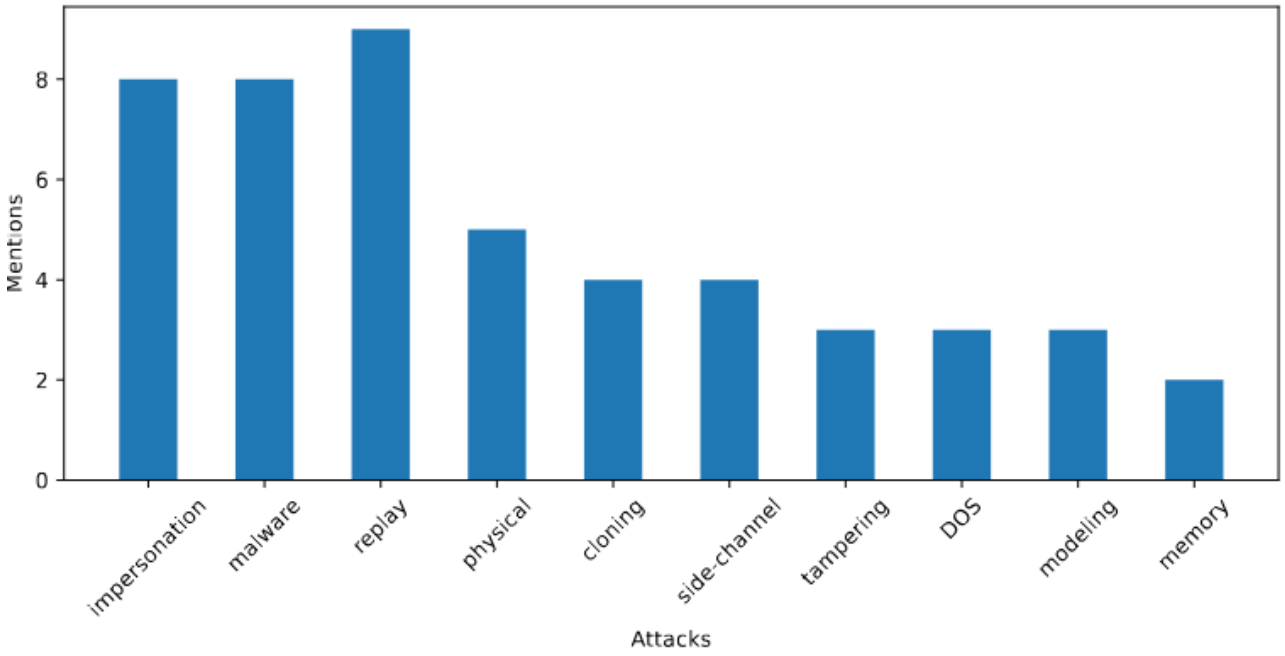

Figure 2: Attacks

3.1.6 Adversary Assumptions

Another important aspect to discuss in remote attestation works for IoT is the adversary model, as it can provide a useful framework to better identify the attacks mentioned above. The characteristic of an adversary can encompass their inherent abilities, their goals and their success criteria.

The adversary is well defined in (6), where it is said to be able to eavesdrop on communications, intercept any message and impersonate devices. The adversary can also physically compromise the IoT devices.

Another similar adversary is found in (9), with the same abilities but with an added limitation of not being able to properly clone or tamper with the PUF, since the attacker cannot access the CRPs directly.

A very similar adversary is considered in (8), with the same network message manipulation capabilities and access to the hardware to perform physical attacks. A successful adversary in this scenario is able to authenticate with the verifier and install malware in Industrial IoT devices, all with the goal of inflicting physical and economic damage.

In (5), the adversary has compromised the device (the PLC in this specific case) with the goal to affect the actuations of the physical process without being detected by the

attestation protocol. An attacker can only execute code on this compromised PLC, which means that they are bound to the inherent limitation of the PLC, such as limited processing power and memory.

A strong adversary which is able to physically access IoT devices is described in (13). The success criteria of the adversary is again to authenticate itself to the verifier and install malware, with the usual goal of inflicting economic and physical damage.

The adversary introduced in (14) has full control of the network, just like in above examples, and can compromise the devices in order to use them to send data to other devices in the network.

The Dolev-Yao adversary model is widely recognized as one of the most powerful in literature (30) and it is considered as the main model (15).

The only paper to not mention any attacks, (16), similarly defines the adversary only with its inability to perform fault attacks or other sophisticated hardware attacks.

3.1.7 Security Evaluation

Since remote attestation protocols are employed to enhance the security of IoT, it is important to assess the actual improvements that are achieved on this front. We consider an actual "security assessment" any explanation (or description) of how the proposed protocol achieves a specific security goal, then we divide these into "formal", "informal" and "empirical".

Informal security assessment

These assessments offer a descriptive evaluation on how their respective protocols deal with the proposed attacks. This is the case of (4), where every attack is followed by a brief explanation on how the protocol would react and neutralize it. The security evaluation offered in (9) contains slightly more extensive descriptions than the previous work but it is still lacking any empirical study or formal proof.

Empirical security assessment

Common empirical assessments simulate the attacks and practically evaluate the system's ability to cope with them.

In (5), the performance evaluation is said to be "theoretical" but it uses empirical experiments that are then assessed through common metrics such as False Positive Rate (FPR), False Negative Rate (FNR), Accuracy, F1-score, Sensitivity, Precision, Specificity and Matthews Correlation Coefficient (MCC).

The security of the PUF against modeling attacks is only verified in (10), where the generated CRPs are checked for uniqueness and uniformity, both characteristics that can foil machine learning attacks. A few modeling attacks are launched against the PUF, after which a Multi-Layer Perceptron (MLP) is selected in order to assess the PUF's resilience against such attacks. Side-channel attacks resilience is also tested in this work, this is done by calculating the correlation coefficient between response bits and the power drawn from the unit.

The simulation of malware attacks in an IoT network is used in (14) to assess the ability of the framework to control, contain, and mitigate malware spread. This is paired with a formal assessment which is presented in the following.

Formal security assessment

This type of assessment can include both formal proofs and tool-assisted verification. As an example, the security properties of (11) have been tested with an automated tool called ProVerif, which uses approximations to deliver a sound proof of such properties. This means that if the tool asserts a specific security property is met, then that is actually satisfied, while it cannot prove when certain properties fail.

Another formal verification tool, Scyther, is used in the security assessment of (13). Scyther leverages unbounded model checking methods and backward symbol state searching techniques to analyse security protocols. This work also provides a formal proof of the soundness of the attestation protocol by using BAN logic, alongside a more qualitative analysis. BAN logic is also used in a similar way in (15).

The probability of detecting malware is calculated formally both in (8) and (12). The former proposes a thorough probabilistic calculation to derive the threshold with which an attack can be detected, then it presents the probability of detection functionally linked to multiple different probabilities of compromised nodes. A different approach is shown in the latter, where the adversary's evasion probability is calculated by showing what is their best relocation strategy. The end result for a memory of N blocks is shown to be (1− 1N)N for every attestation protocol run, which is the best possible scenario from a security perspective.

The informal security assessment of (14) is accompanied by many lemmas with proofs, starting from the assumption that a PUF is in fact unique and unclonable and ending with a proof that the malware is successfully contained in the proposed framework.

No security assessment

Only one paper (3) specifies that the security evaluation of the proposed attestation protocol is out of scope. The remaining papers do not explain how the security goals are met and usually just mention the attacks that they thwart without any further analysis.

3.1.8 Performance Evaluation

There are three types of overhead that should be considered when applying an attestation protocol to an IoT device: 1. computational complexity, meaning the time that it takes to perform the actual attestation algorithm, 2. size overhead, or how much space is occupied by the attestation software and hardware, and 3. network overhead, measuring the size of the attestation payloads sent over the network and their rate of transmission. The computational complexity of an attestation protocol is of particular importance in the IoT field, as many processes are critical and thus cannot suffer any significant slow-down due to added security protocols. Another interesting aspect that is often glossed over is the network overhead caused by the attestation protocols, rendered particularly important in those protocols that rely on timing.

Energy consumption considerations

Parallel to the importance of considering the computational complexity of attestation protocols in IoT is the focus on energy consumption. IoT systems often cannot afford to allocate a significant amount of their energy to security protocols, which means that it is fundamental to study how these protocols affect the energy consumption of the devices. Only two of the reviewed papers explicitly consider energy consumption, (10) and (11). In the former, the ratio of energy per bit is measured when generating a key from the PUF, alongside the speed required to perform the operation. The resulting total energy for the attestation of a single block of data is measured at 9.88pJ for every 64-bits data block. In the latter, the energy consumption analysis is much more detailed, with a thorough comparison of the performances of the proposed protocol versus others in the literature, accounting for key size.

Experimental complexity assessment

The computational complexity of the protocols can be empirically assessed by measuring the duration of their runtime. This technique is employed in (4), where the duration of the execution of the entire protocol has been measured to around 28.5 seconds.

Very detailed timing data is provided in (7), where the protocol is shown to have very little added computational overhead when compared to previous work. Another very detailed analysis can be found in (9) where every component involved in the protocol is measured individually to check where the complexity resides.

The speed of the protocol in (10) is measured in kbps (120 kbps is the best result), as the duration of every attestation run depends solely on the data block size.

In (13), the computational overhead of the protocol is compared to 4 existing relevant protocols and it is shown that it is a clear improvement over 3 of them, while being very close but eventually losing in computational performance to (8). The communication overhead (discussed below) is what sets this paper apart from the compared literature.

The main focus of (14) is patching of multiple devices, thus their evaluation for computational overhead measures the time it takes for the protocol to patch a fixed number of devices.

Each operation involved in the protocol presented in (15) is individually timed and then favorably compared to two similar works in the literature. A similar approach is taken in (16), where the process has been divided into atomic components and the execution time of each component has been recorded and compared to other approaches.

Formal complexity assessment

Interestingly, only two papers provide a formal complexity study of their protocol using Big O notation (6), (8). The formal evaluation of (6) only states that in their approach, the attestation protocol has complexity O(N) when N iterations are needed, while other approaches in literature require O(Nmlnm), where m is is the field size for elliptic curve cryptography.

Size overhead

The whole SACHa architecture is shown to occupy less than 9% of the entire space of the FPGA (4).

A much more detailed breakdown of the space needed to implement the approach is offered in (7) where the footprint of both the PUF and the secure monitor implementations are evaluated in term of flip-flops, lookup tables, DSPs and BRAMS. The resulting architecture is estimated to not consume more than 1% of the allotted space for the PUF and 25% for the FPGA secure monitor. With the same spirit, all components are individually evaluated for space in (9).

The size of the public key and of the generated signatures is the main concern for (16), along with the code size. This is corroborated by an analysis on the communication overhead of the protocol which we mention in the following.

Communication overhead assessment

In (16), the size of the generated signature is the most important factor for reducing the communication overhead, as the size of the public key is less important over multiple shared messages.

The protocol introduced in (8) is argued to have less communication overhead than other related ones because it only requires the exchange of two messages.

The communication overhead of the system proposed in (13) is compared to 4 works in the literature and shows a notable improvement over all of them, especially while increasing the number of IoT devices to be attested. Similarly, the communication overhead of (11) is compared to other works in literature and is shown to be better than all but one, which is in turn worse in energy consumption.

The communication overhead of the central authentication server (IOC) presented in (15) is reduced when authenticating a large number of IoT devices simultaneously using the proposed protocol. This is a result of the protocol's approach to aggregate and preprocess a large number of IoT attestation request messages through the aggregator before sending them to the IOC for processing. Consequently, this approach significantly reduces the communication overhead between the IoT devices and the IOC.

The delay of the network communication is identified as the dominant characteristic of the protocol duration in (4).

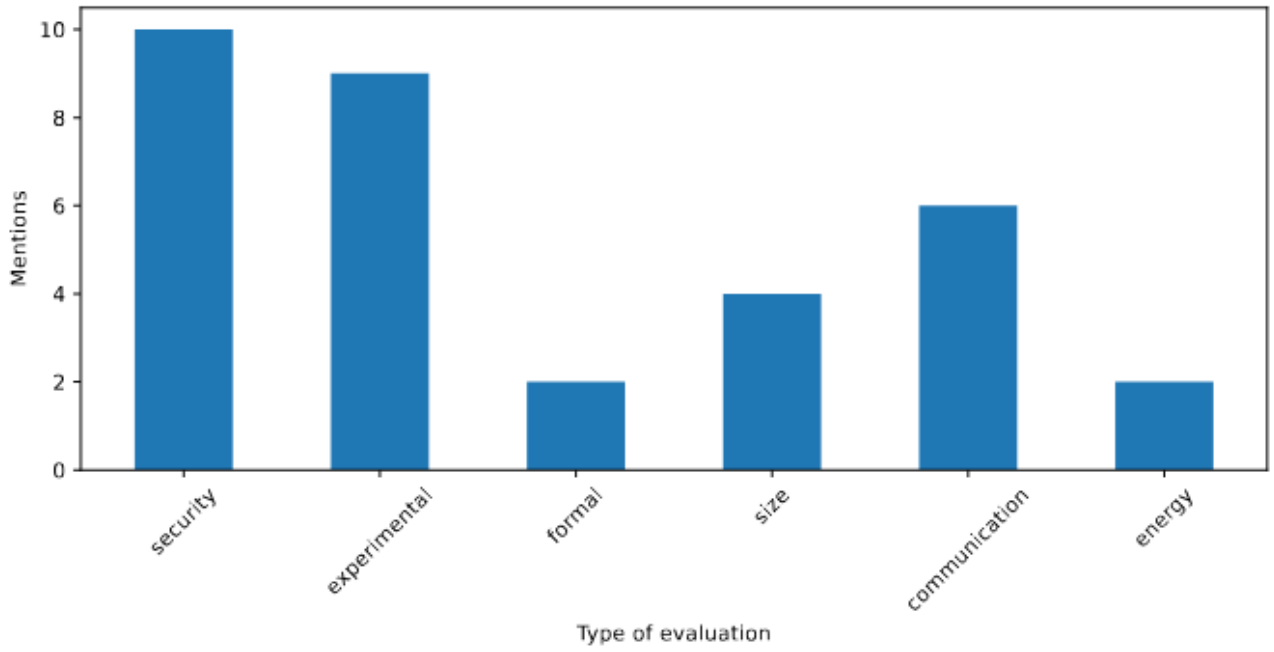

Figure 3: Type of evaluation

An intuitive representation of the evaluation types used in the papers can be found in Figure 3 .

3.2 Authentication

In this project we are interested in mutual authentication between the device and an external verifier. We also need to establish a shared session key for secure

communication over a possibly insecure channel. High level protocols use different low-level cryptosystems to achieve these goals

Symmetric and asymmetric cryptography

A symmetric cryptosystem uses the same key for both encryption and decryption, whereas an asymmetric cryptosystem uses a private key and a public one.

Asymmetric, or public-key, cryptography has the big advantage of not requiring a third party to know the secret key in order to communicate securely. On the other hand, symmetric cryptography is often simpler and more efficient.

DH

Diffie-Hellman (DH) is the earliest known method that allows two parties to securely establish a shared cryptographic key (31).

It's a form of asymmetric cryptography. Each party randomly generates a secret value that is used to generate a public value, using exponentiation and modulo operations. The parties exchange the public values, and both generate a new secret value using exponentiation and modulo operations. The generated value is the same for both parties and can thus be used as a shared session key.

The security of this Diffie-Hellman scheme is based on the difficulty of the Discrete Logarithm Problem. If an efficient solution to this problem is found, the secret values can be computed by an attacker.

The original protocol is not authenticated and is vulnerable to a man-in-the-middle attack.

RSA

RSA (from the names of the inventors, Rivest, Shamir and Adleman) is another asymmetric cryptosystem, whose security is based on the integer factorization problem (32).

The algorithm consists of key generation and distribution, encryption and decryption. Since it's a heavy algorithm, it's often used only for the initial exchange of a symmetric encryption key. Subsequent communication is carried out using lightweight symmetric encryption algorithms.

ECC

Elliptic Curve Cryptography (ECC) is a form of asymmetric cryptography using elliptic curves over finite fields. Unlike RSA and other cryptosystems that exploit the inherent difficulty of integer factorization, ECC's security is based on the infeasibility of the Elliptic Curve Discrete Logarithm Problem (33) (34).

ECC is used for key agreement and digital signatures in high level protocols such as TLS, and is suitable to be implemented in embedded systems, given the limited resources it needs.

The primary benefits of ECC over traditional cryptosystems is the smaller key size:

to provide a security that is equivalent to 128-bit AES, NIST recommends 3072-bit RSA/DH or 256-bit ECC (35).

Computing custom curves/domain parameters is risky and time-consuming, so the NIST recommended *standard curves*, chosen for their high security and efficiency (36).

Unfortunately, these curves are no longer trusted since a potential backdoor in a recommended cryptographically secure pseudorandom number generator (CSPRNG) was found.

Also, real world ECC systems are vulnerable to many attacks due to insecure implementation of standard curves.

The SafeCurves project (37) proposes curves that can easily be implemented in a secure way.

Another weakness of ECC is that it seems to be an easier target for quantum computing compared to RSA.

EKE

In Encrypted Key Exchange (EKE), a shared password is used to generate a session key (38).

In a naive key negotiation, two parties, Alice and Bob, are sharing a password. A random number generated by Alice could be sent to Bob, encrypted with the shared password. Then the random number becomes the shared session key. This simple protocol is vulnerable to several attacks.

In EKE, participants have public and private keys, used for encryption and decryption. Alice sends her public key to Bob, encrypted with the shared password. Only Bob knows Alice's public key and uses it to encrypt and send to Alice a random number, to be used as a session key.

To make the protocol resistant to replay attacks, Alice and Bob exchange random challenges (nonces), encrypting them with the session key. They then prove to each other that they know the nonces.

EKE can also be used to add authentication to Diffie-Hellman key exchange.

Strong PUF Authentication

With the advent of PUFs, cheap and secure authentication of hardware devices became possible (39). The first protocol that was proposed for the authentication of a device with an external verifier consists of two phases: a registration, or enrollment, phase, during

which a large set of challenge-response pairs are collected by the verifier and saved in a secure database. And an authentication phase, in which the device produces a response to a challenge sent by the verifier. The verifier accounts for a certain number of different bits in the response, as the strong PUF is not 100% reliable. The used CRP is discarded, to avoid replay attacks. This protocol is very simple and efficient in terms of time but requires a large secure CRP database in the verifier. Also, the device PUF is not protected, so unencrypted challenges and responses can be used to learn the behavior of the PUF.

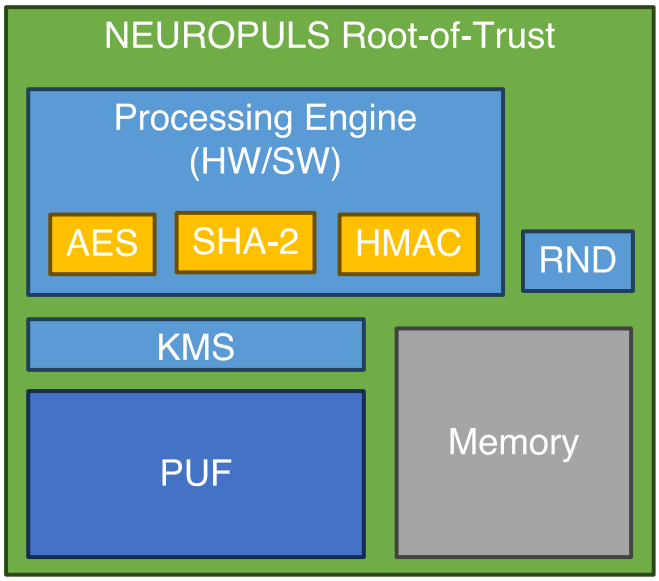

Figure 4: High-level preliminary NEUROPULS Root-of-Trust components.

4. Proposed Authentication Approach

Authentication is the first step in secure communication, which consists of verifying the identity of a participant (e.g., a device) before exchanging sensitive data with another participant. The context of NEUROPULS includes two main roles: the device to be verified and an external entity acting as the verifier. This context imposes two requirements: First, the authentication shall be mutual, i.e., considering that sensitive data travel in both directions, the device and the verifier should verify each other's identity. Second, the authentication process shall be lightweight because the resources on the device are constrained.

Existing authentication strategies based on PUFs require the verifier to store a large database of CRPs for each device, as first described in seminal works on PUFs (40), (41) and detailed by Suh et al. (42), or to exploit heavier protocols [6], [7]. However, to meet the lightweight requirement, we decided to adopt a different strategy, that is, HSC-IoT,

the authentication procedure proposed by Hossain et al. [8]. Their idea is to use only a single CRP as a shared secret between the device and the verifier to support mutual authentication and to update it after each use with a fresh CRP.

Practically, the first CRP is shared at manufacturing time and is meant to support the first actual authentication session. At each actual authentication session, the device uses a fresh CRP that is based on the response of the previously used CRP. The new response is sent to the verifier in encrypted form and, if mutual authentication succeeds, the current CRP is updated on both the device and the verifier.

Figure 5 shows a UML sequence diagram showing messages exchanged between the device and the verifier according to the mutual authentication protocol.

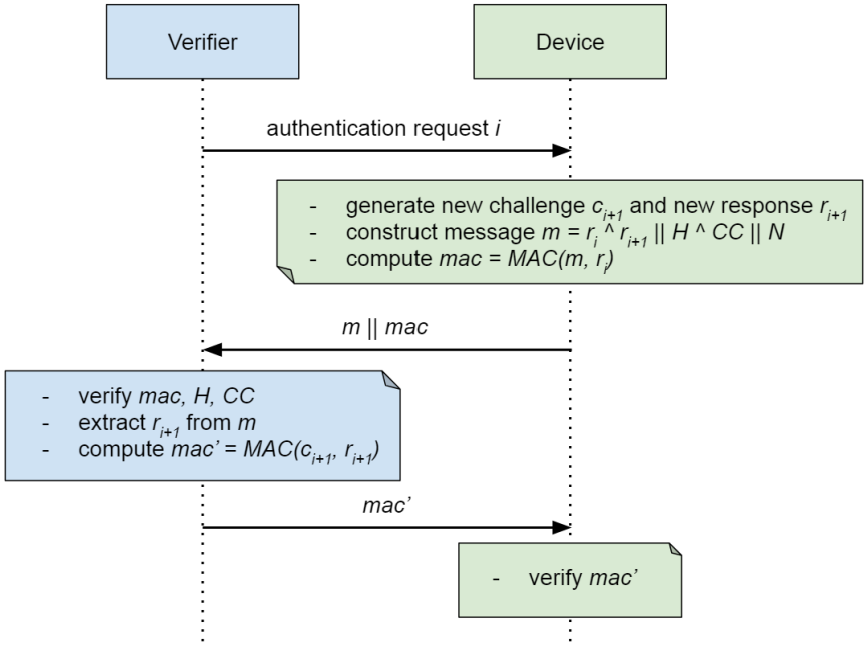

Figure 5: Mutual authentication protocol between the device and the verifier.

The protocol starts with the authentication request *i* from the verifier. The device derives the new challenge c_{i+1} from the current response r_i, using it as a seed for a pseudo-random number generation (RNG) function known to both participants, $c_{i+1} = RNG(r_i)$. The device computes a message m containing the new response r_{i+1}, XORed with the current response r_i. In the figure, the operator ^ denotes the XOR operation and || denotes concatenation. The message may also contain proof of the integrity of the software, such as the hash of the memory H and a clock count CC that represents the time needed to perform a given task. The message can contain a nonce N for freshness. This message m is then sent to the verifier along with its MAC signature, calculated using the MAC(data, key) function, whose first argument is the data to sign and the second is

the key, r_i in this case. Now, the verifier can authenticate the device by checking the message signature using the secret r_i. The new response r_{i+1} is derived from m and stored in the verifier to be used as a shared secret in the next authentication session. Finally, the verifier authenticates the device by demonstrating that it knows the new secret r_{i+1}, which is used to sign c_{i+1} (generated using r_{i+1}) through the MAC function again.

This protocol only needs one CRP to be known by the verifier at any point, which is more scalable than other solutions that require a large database of CRPs. In addition, CRPs are kept confidential because they are never exchanged in clear text. Although this protocol is very lightweight, it is designed to resist many attacks, as discussed by Hossain et al. [8].

5. Neural network configuration and data encryption

Another important security requirement is the confidentiality of the actual data processed by the accelerator and the confidentiality of the neural network configuration. To meet this confidentiality requirement, encryption encodes the data in all communications with external parties and all the software running on the device. The NN configuration, the input data to the device, and the computation output are encrypted using the secret keys described in Section II. This key is never exposed to the software layer, but encryption and decryption occur on the hardware in the implementation of the security primitives shown here:

Function name	*Parameters*	*Results*
load_network	ciphered_network	
execute_network	ciphered_input	ciphered_output

load_network receives the neural network configuration in encrypted form. The configuration is decrypted in hardware and loaded in the accelerator. execute_network takes the input as a decrypted parameter and fed to the accelerator. Then, the computation result is encrypted and returned by this function. As specified by the function signatures, data are never exposed in plaintext to the software. Data are decrypted internally by the device using primitives that never leave plaintext in the memory after execution, thus preserving confidentiality even against an internal attacker capable of reading the RAM.

6. Conclusions and next steps

This deliverable has presented a very detailed overview of the most recent advancements in remote software attestation and authentication, specifically targeting those works that target IoT systems and that leverage physically unclonable functions (PUFs).

Both of these aspects are fundamental: 1. The neuromorphic accelerator of the NEUROPULS project can be deployed in environments with limited access to power, sharing this aspect with IoT devices and 2. The project also proposes a novel optical PUF that can outperform existing PUF architectures, making its use in the security protocols paramount.

The literature overview of remote software attestation highlighted how most existing protocols tend to leverage an “ideal” PUF, which is a PUF that does not suffer from any of the problems that are common to these devices, namely their instability (i.e. the inherent bit error rate and the increase of it due to aging factors). This of course is not acceptable when designing a security protocol for a device that will be deployed with a real PUF, such as the neuromorphic accelerator proposed in this project.

The next iteration of this deliverable will propose:

- a novel remote software attestation protocol based on realistic PUFs, meaning that it will account for the intrinsic bit error rate of these devices and address the problems introduced by their aging.
- A novel authentication protocol that is lighter and more secure than the one identified in this document.

7. References

1. *NEUROPULS: NEUROmorphic energy-efficient secure accelerators based on Phase change materials aUgmented siLicon photonicS.* **Fabio Pavanello, Cedric Marchand, Ian O'Connor, Régis Orobtchouk, Fabien Mandorlo, Xavier Letartre, Sebastien Cueff.** Venice : IEEE European Test Symposium, 2023. doi: 10.1109/ETS56758.2023.10173974.

2. *Photonic Physical Unclonable Function Based on Symmetric Microring Resonator Arrays.* **Paul Jimenez, Raphael Cardoso, Maurício Gomes de Queiroz, Mohab Abdalla, Clément Zrounba, Ulrich Rührmair, Cédric Marchand, Xavier Letartre, and Fabio Pavanello.** Tacoma, Washington United States : Optica, 2023. ISBN: 978-1-957171-29-6.

3. *Secure boot and remote attestation in the sanctum processor.* **Ilia Lebedev, Kyle Hogan, and Srinivas Devadas.** 2018. IEEE 31st Computer Security Foundations Symp. (CSF).

4. *SACHa: Self-attestation of configurable hardware.* **Jo Vliegen, Md Masoom Rabbani, Mauro Conti, and Nele Mentens.** 2019. Design, Automation & Test in Europe Conf. & Exhibition (DATE).

5. *PAtt: Physics-based Attestation of Control Systems.* **Hamid Reza Ghaeini, Matthew Chan, Raad Bahmani, Ferdinand Brasser, Luis Garcia, Jianying Zhou, Ahmad-Reza Sadeghi, Nils Ole Tippenhauer, and Saman Zonouz.** 2019. 22nd Int'l Symp. on Research in Attacks, Intrusions and Defenses (RAID 2019).

6. *Defining trust in IoT environments via distributed remote attestation using blockchain.* **Uzair Javaid, Muhammad Naveed Aman, and Biplab Sikdar.** 2020. Twenty-First Int'l Symp. on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing.

7. *Sgx-fpga: Trusted execution environment for cpu-fpga heterogeneous architecture.* **Ke Xia, Yukui Luo, Xiaolin Xu, and Sheng Wei.** 2021. 58th ACM/IEEE Design Automation Conf. (DAC).

8. *ATT-Auth: A hybrid protocol for industrial IoT attestation with authentication.* **Sikdar, Muhammad Naveed Aman and Biplab.** 6, 2018, IEEE Internet of Things Journal, Vol. 5, pp. 5119–5131.

9. *AAoT: Lightweight attestation and authentication of low-resource things in IoT and CPS.* **Wei Feng, Yu Qin, Shijun Zhao, and Dengguo Feng.** 134, 2018, Computer Networks, pp. 167–182.

10. *Design, analysis and application of embedded resistive RAM based strong arbiter PUF.* **Rekha Govindaraj, Swaroop Ghosh, and Srinivas Katkoori.** 17, 2018, IEEE Transactions on Dependable and Secure Computing, Vol. 6, pp. 1232–1242.

11. *HAtt: Hybrid remote attestation for the Internet of Things with high availability.* **Muhammad Naveed Aman, Mohamed Haroon Basheer, Siddhant Dash, Jun Wen Wong, Jia Xu, Hoon Wei Lim, and Biplab Sikdar.** 8, 2020, IEEE Internet of Things Journal, Vol. 7, pp. 7220–7233.

12. *A lightweight authentication and attestation scheme for in-transit vehicles in IoV scenario.* **Tejasvi Alladi, Sombuddha Chakravarty, Vinay Chamola, and Mohsen Guizani.** 12, 2020, ation and attestation scheme for in-transit vehicles in IoV scenario., Vol. 69, pp. 14188–14197.

13. *Scalable attestation protocol resilient to physical attacks for IoT environments.* **inyin Xiang, Jin Cao, and Weiguo Fan.** 15, 2020, IEEE systems journal, Vol. 3, pp. 4566–4577.

14. *IoT-proctor: a secure and lightweight device patching framework for mitigating malware spread in IoT networks.* **Muhammad Naveed Aman, Uzair Javaid, and Biplab Sikdar.** 16, 2021, IEEE Systems Journal , Vol. 3, pp. 3468–3479.

15. *RPRIA: Reputation and PUF-Based Remote Identity Attestation Protocol for Massive IoT Devices.* **Jin Cao, Sheng Li, Ruhui Ma, Yuxi Han, Yueyu Zhang, and Hui Li.** 19, 2022, IEEE Internet of Things Journal, Vol. 9, pp. 19174–19187.

16. *A lightweight remote attestation using PUFs and hash-based signatures for low-end IoT devices.* **Roberto Román, Rosario Arjona, and Iluminada Baturone.** 2023, Future Generation Computer Systems.

17. *Design of SRAM PUF with improved uniformity and reliability utilizing device aging effect.* **Kim, Achiranshu Garg and Tony T.** 2014. IEEE Int'l symp. on circuits and systems (ISCAS).

18. *Improved ring oscillator PUF: An FPGA-friendly secure primitive.* **Schaumont, Abhranil Maiti and Patrick.** 24, 2011, Journal of Cryptology, pp. 375–397.

19. *MEMS gyroscopes as physical unclonable functions.* **Oliver Willers, Christopher Huth, Jorge Guajardo, and Helmut Seidel.** 2016. ACM SIGSAC Conf. on Computer and Communications Security.

20. *A new arbiter PUF for enhancing unpredictability on FPGA.* **Takanori Machida, Dai Yamamoto, Mitsugu Iwamoto, Kazuo Sakiyama.** 2015, The Sicentific World Journal.

21. *XOR gate based low-cost configurable RO PUF.* **Lei Zhang, Chenghua Wang, Weiqiang Liu, Maire O'Neill, and Fabrizio Lombardi.** 2017. IEEE Int'l Symp. on circuits and systems (ISCAS).

22. *Trapdoor computational fuzzy extractors and stateless cryptographically-secure physical unclonable functions.* **Charles Herder, Ling Ren, Marten Van Dijk, Meng-Day Yu, and Srinivas Devadas.** 2016, IEEE Transactions on Dependable and Secure Computing.

23. *ACRO-PUF: A low-power, reliable and aging-resilient current starved inverter-based ring oscillator physical unclonable function.* **Chang, Chao Qun Liu and Yuan Cao and Chip Hong.** 64, 2017, IEEE Transactions on Circuits and Systems I: Regular Papers, Vol. 12, pp. 3138–3149.

24. *An aging-resistant RO-PUF for reliable key generation.* **Md Tauhidur Rahman, Fahim Rahman, Domenic Forte, and Mark Tehranipoor.** 3, 2015, IEEE Transactions on Emerging Topics in Computing, Vol. 4, pp. 335–348.

25. *Hardware security for extended merkle signature scheme using SRAM-based PUFs and TRNGs.* **Roberto Román, Rosario Arjona, Javier Arcenegui, and Iluminada Baturone.** 2020. 32nd Int'l Conf. on Microelectronics (ICM).

26. *An ultracompact switching-voltage-based fully reconfigurable RRAM PUF with low native instability.* **Xiaojin Zhao, Qiang Zhao, Yongpan Liu, and Feng Zhang.** 7, 2020, IEE E Transactions on Electron Devices, Vol. 67, pp. 3010–3013.

27. *SWATT: Software-based attestation for embedded devices.* **Arvind Seshadri, Adrian Perrig, Leendert Van Doorn, and Pradeep Khosla.** s.l. : IEEE Symp. on Security and Privacy, 2004.

28. *A framework for classifying denial of service attacks.* **Alefiya Hussain, John Heidemann, and Christos Papadopoulos.** s.l. : Conf. on Applications, technologies, architectures, and protocols for computer communications, 2003.

29. *PUF modeling attacks on simulated and silicon data.* **Ulrich Rührmair, Jan Sölter, Frank Sehnke, Xiaolin Xu, Ahmed Mahmoud, Vera Stoyanova, Gideon Dror, Jürgen Schmidhuber, Wayne Burleson, and Srinivas Devadas.** IEEE transactions on information forensics and security.

30. *The gap between promise and reality: On the insecurity of XOR arbiter PUFs.* **Becker, Georg T.** 2015. Cryptographic Hardware and Embedded Systems–CHES 2015: 17th Int'l Workshop.

31. *New directions in cryptography.* **Diffie, W. and Hellman, M.** 1976, IEEE Transactions on Information Theory, Vol. 22, pp. 644-654.

32. *A method for obtaining digital signatures and public-key cryptosystems.* **Rivest, R. L., Shamir, A. and Adleman, L.** New York, NY, USA : Association for Computing Machinery, February 1978, Commun. ACM, Vol. 21, pp. 120–126. ISSN: 0001-0782.

33. *Elliptic Curve Cryptosystems.* **Koblitz, Neal.** s.l. : American Mathematical Society, 1987, Mathematics of Computation, Vol. 48, pp. 203–209. ISSN: 00255718, 10886842.

34. *Use of Elliptic Curves in Cryptography.* **Miller, Victor S.** [ed.] Hugh C. Williams. Berlin : Springer Berlin Heidelberg, 1986. Advances in Cryptology — CRYPTO '85 Proceedings. pp. 417–426. ISBN: 978-3-540-39799-1.

35. **NIST.** Cryptographic Algorithms and Key Sizes for Personal Identity Verification. *Cryptographic Algorithms and Key Sizes for Personal Identity Verification.* [Online] 2024. https://csrc.nist.gov/pubs/sp/800/78/5/final.

36. —. Digital Signature Standard (DSS). *Digital Signature Standard (DSS).* [Online] 2023. https://csrc.nist.gov/pubs/fips/186-5/final.

37. **Bernstein, Daniel J. and Lange, Tanja.** SafeCurves: choosing safe curves for elliptic-curve cryptography. *SafeCurves: choosing safe curves for elliptic-curve cryptography.* [Online] 2024. https://safecurves.cr.yp.to.

38. *Encrypted key exchange: password-based protocols secure against dictionary attacks.* **Bellovin, S. M. and Merritt, M.** 1992. Proceedings 1992 IEEE Computer Society Symposium on Research in Security and Privacy. pp. 72-84.

39. *A Retrospective and a Look Forward: Fifteen Years of Physical Unclonable Function Advancement.* **Chang, Chip-Hong, Zheng, Yue and Zhang, Le.** August 2017, IEEE Circuits and Systems Magazine, Vol. 17, p. 32.

40. *Physical One-Way Functions.* **Pappu, Ravikanth, Ben Recht, Jason Taylor, and Neil Gershenfeld.** 2002, Science.

41. *Silicon physical random function.* **Blaise Gassend, Dwaine Clarke, Marten Van Dijk, and Srinivas Devadas.** 2002. 9th ACM Conference on Computer and Communications Security.

42. *Physical unclonable functions for device authentication and secret key generation.* **Devadas, G. Edward Suh and Srinivas.** 2007. Proceedings of the 44th annual design automation conference.