PUFsecurity

PUF-based Solutions and Applications

Lawrence Liu

Book Series on Hardware Security

Founding Editor: Charles Ching-Hsiang Hsu

PUF-based Security Solutions and Applications

Lawrence Liu

Book Series on Hardware Security

Founding Editor: Charles Ching-Hsiang Hsu

Quantum Tunneling PUF: Basics, Circuits, and Security Applications

By Kent Kai-Hsin Chuang

PUF-based Security Solutions and Applications

By Lawrence Liu

Anti-Tampering Designs in Hardware Security

By Meng-Yi Wu, Kent Kai-Hsin Chuang

Random Number: Generation, Emulation, and Practice

By Balance Chun-Heng You, Kent Kai-Hsin Chuang

Contemporary Cryptography: Fundamentals and Algorithms

By Chun-Yuan Yu, Danny Yung Chih Chen, Wayne Wen-Ching Lin

TPM and HSM: Implementation and Application

By Shih-Li Hsu

PUF-based Security Solutions and Applications

Written by Lawrence Liu
Foreword by Charles Ching-Hsiang Hsu
Edited by Ada Ying-Yun Huang, Andrew Irvin, Ann Yi-An Lin, Evans Ching-Song Yang, Meng-Yi Wu

For further requests, please contact Info@pufsecurity.com.

Printed by
PUFsecurity Corporation

First Edition, 2024

PUFsecurity

About the Author

Lawrence Liu

After twenty-some years in the discrete DRAM and Flash (NAND and NOR) memory business, Lawrence Liu decided to try his hand at IP memory and joined eMemory in 2018. He then moved to PUFsecurity in 2019 to learn more about hardware security as a deputy director of the R&D department. Soon after joining, it quickly became obvious that integrating a PUF with traditional designs could yield great benefits, not only enhancing security, but also improving efficiency and performance for certain applications reliant on traditional designs. It was at this point that Lawrence switched from a role centered around technical design to one focused on applying his technical know-how to help others understand the mysteries of PUFs and PUF-based security.

Lawrence received his B.S. and M.S. degrees in electrical engineering from Stanford University, focusing on computer architecture. After presenting at the Design Automation Conference and publishing his work in Embedded Computing Design, Design & Reuse, and PSA Certified as a featured whitepaper, Lawrence seeks more opportunities to promote the advantages of PUF-based security.

Foreword

“I was seldom able to see an opportunity until it had ceased to be one.”

For many years, this quote from Mark Twain has been imprinted in my mind, reminding me that good ideas, whenever they arrive, must be seized to become genuine opportunities. So when I met Tang Ma (馬騰桂) in 2015, shortly after his retirement from Maxim, his explanation of the possibilities of PUF (Physically Unclonable Function) for semiconductors, and the critical role it could play in security, felt like an opportunity worth pursuing.

As a semiconductor device physicist, I immediately thought that the characteristics associated with a device’s physical dimensions could potentially produce unclonable features, due to the natural variances during the fabrication process. The thickness and quality of the film, for example, or the length and breadth of a transistor gate, parameters such as those produce minute differences in a transistor’s microstructure. When extracting and comparing the adjacent transistors’ electrical characteristics, we can register the difference as 0 or 1 if it is distinct enough within the measurable region.

So, the critical factor is identifying the parameter with a linear difference in its geometry, yet the electrical behavior measured from the terminals is exponential. Drawing on my decades of work on electron transport in thin gate dielectrics, I immediately thought that the tunneling current between gate dielectrics would be

significantly different, even if the thickness or quality of the dielectric retains only minimal variations.

To verify this idea, I brought Wei-Jer (翁偉哲), Meng-Yi (吳孟益), Hsin-Ming (陳信銘), and Evans (楊青松) to collaborate together, by setting up an experiment that used our existing NeoFuse (anti-fuse using oxide tunneling) with two oxide capacitors in parallel. The results produced two critical findings:

1. *Only one of the oxide capacitors will have a significant tunneling current when the applied voltage is large enough.*
2. *The occurrence of the tunneling current between the pair of oxide capacitors is random, arriving equally on either the left or right capacitor.*

I knew this technology would be foundational for securing the future of interconnected computing. Since then, these two critical findings have established the silicon fingerprint, NeoPUF, which led to the founding of our company, PUFsecurity, and the development of our integrated suite of security subsystems that play a vital role in the Hardware Security ecosystem. After establishing PUFsecurity, we were encouraged to try and enlighten the broader semiconductor community on the risk of unsecured chips through the education and promotion of Quantum Tunneling PUF, which ultimately led to publishing this book series to introduce the fundamentals of PUF-based Hardware Security.

These books will cover a wide range of topics, including quantum tunneling PUF, an overview of PUF-based solutions, tamperproof

design, random number generation, and the importance of cryptography to Hardware Security. We will also cover applications like AI and IoT, as well as provide an overview of the latest security standards and regulations. Our hope is that they can play a role in furthering our collective understanding of Hardware Security and its impact on the future of computing.

As the internet enables more and more connected devices, deploying a traditional physically secure boundary for a system is no longer sufficient. We must embrace the principles of “Zero Trust,” explicitly verifying every linked device and decentralizing the secure boundaries of the network to a solution embedded on each device. It is similar to how we biometrically identify each of us through our fingerprints.

A PUF can play the role of a chip fingerprint, uniquely identifying both the device and the chip in which it is embedded. It can then be integrated with an NVM OTP to establish a Hardware Root of Trust that can generate, store, and safely manage Keys. Then, we can implement a Security Coprocessor by combining a PUF-based Hardware Root of Trust with Crypto Engines to provide a device with a fully integrated Security Subsystem.

I want to offer my sincerest gratitude to everyone that helped realize this project, in particular, its authors; Dr. Kent Chuang (莊愷莘), Lawrence Liu (劉持志), Dr. Meng-Yi Wu (吳孟益), Balance You (游鈞恆), Danny Chen (陳勇志), Dr. Wayne Lin (林文景), Chun-Yuan Yu (游鈞元), Dr. Li Hsu (徐世理) and Matthew Yu (于立宏). I would also like to thank Ada Huang (黃楹芸), Andrew Irvin

(張安筑), and Ann Lin (林奕安) for their diligent work editing, formatting, and publishing these books. Lastly, I would like to thank Dr. Evans Yang (楊青松) for his tireless dedication in guiding this project.

Thank you all.

Charles Ching-Hsiang Hsu
April 2023

Preface

The idea for writing this book was formed after a series of discussions at work revolved around how it was a shame that the concept of a physically unclonable function (PUF) was mostly known only to people in the specialized fields of hardware security and device physics. My colleagues and I thought about how beneficial it would be if PUFs and their practical applications were better known to a wider audience. In fact, I must admit that before joining eMemory, I was also guilty of not being familiar with PUFs. However, after joining the eMemory family and becoming acquainted with the main concepts, I soon realized the unique benefits that a PUF brings to the security of a system. So, when the opportunity to introduce PUF-based security to a general audience arose, I naturally jumped at the chance to put these thoughts to paper.

A portion of this book is based on the customer document package that is included in every set of deliverables from PUFsecurity. To make this book more accessible, all technical concepts and terms were examined to determine if they could be even further expanded upon. The goal being to create an easy-to-follow guide for readers to understand the benefits of PUFs and how PUF-based solutions operate in a secure system.

After this important background information was established, the dual solutions of PUFrt and PUFcc from PUFsecurity were introduced before selected applications pulled from the growing

fields of AI and IoT were described in detail to give readers practical examples what PUF-based solutions can do in real life scenarios. Originally created as part of a separate book on AI and IoT, the chapters related to applications were later incorporated into this book.

From the original book concept to the writing of this preface, a period of almost two years has passed. Beginning with an outline of topics to cover and weathering three major rewrites and reorganizations, this book has finally been shaped into the form that you have in your hands today. I hope reading this book is as rewarding as I have found writing it.

Lastly, as creativity does not flourish in a vacuum, I must thank the numerous contributors I consulted for advice and technical information, as well as the editors who spent untold hours poring over my rough manuscripts, helping to polish the text and professionally laying it out.

Lawrence Liu
September 2023

Hardware Security Overview

We are all eagerly embracing the technology revolution underway with the Internet of Things (IoT), artificial intelligence (AI), e-finance, and electric vehicles. However, we need to safeguard against the security risks they create. While these technologies are rapidly evolving, the number of connected devices will soar in the near future. Securing increasingly sophisticated device networks has become a critical security topic. To accomplish the security goals, it is important to make connected devices secure by design, which requires intensive effort to make hardware security a fundamental part of the conceptual stage for electronic devices and systems.

The primary consideration for hardware security is the use of dedicated components as gatekeepers in a system. This leads to three important research topics in this field: hardware acceleration, countermeasures against attacks, and Hardware Roots of Trust. Implementing cryptographic algorithms in hardware is typically faster than in software. Especially for public-key encryption algorithms and signature schemes, performing operations in software may be unacceptably slow within some platforms. Hardware implementations for cryptographic algorithms are inherently more efficient and elegant.

For edge devices deployed anywhere in the world, the risks of hijacking by malicious parties and vulnerability to attack are high. Under such circumstances, a device, and the system on top of it, could be vulnerable even if standardized cryptographic algorithms

and security protocols are applied. This weakness can be exploited by advanced attack techniques, including side-channel attacks [1], fault attacks [2], and invasive physical attacks [3]. Consequently, we need to understand possible vulnerabilities caused by new attack techniques and design relevant hardware countermeasures.

A Hardware Root of Trust (HRoT) should be the first element in the Chain of Trust of a security system to adequately protect the physical layer. It should provide an unpredictable and tamperproof secret that enables the required hardware security features, which, historically raises security issues around generation and storage.

Hardware Security Book Series Overview

PUFsecurity Corporation published a Hardware Security Book Series, covering a wide range of topics on hardware security. The book series starts with introducing the essential knowledge about Physically Unclonable Functions (PUFs) and PUF-based Root of Trust (RoT) solutions. The next milestone is understanding how security applications can benefit from high-quality PUF and RoT. The next book in the series introduces several integrated PUF-based IPs as a fully comprehensive security solution. The discussions of these solutions will cover hardware architecture, functionality, specifications, and important use cases in Artificial Intelligence (AI) and the Internet of Things (IoT).

More details about generic RoT solutions using NeoPUF, a technology developed by eMemory Technology, are covered in two following books. One focuses on anti-tampering features of

NeoPUF and RoT solutions, including design techniques applied to mitigate common security threats. The other book describes true random number generators, their essential concepts, and demonstrations of their design integration within RoT solutions.

Three types of cryptography are introduced given their theory and design methods in circuit implementation, which include Cryptographic Hash Functions, Symmetric-Key Ciphers, and Public-Key Cryptography. It is crucial to understand the significant differences between algorithms executed in software compared to those in hardware. Finally, two modules, Trusted Platform Module (TPM) and Hardware Security Module (HSM), are illustrated in the last book. Through the specification explanation, we will learn the types of TPMs and HSMs that are secure enough to protect keys during the lifecycle and learn methods to improve the relevant designs of the modules.

This book series addresses the methodologies, applications, and market insights related to the core technology in hardware security comprehensively. It serves as a practical and handy tool for readers at all levels, from beginners to experts. Readers can acquire better and deeper understandings about PUF, RoT, PUF-based solutions, and hardware security, which will further assist them in pursuing excellence in academia and industry.

Contents

List of Figures

List of Tables

1

Why is security important? This chapter gives a brief overview of the current challenges and threats in security. Subsequently, we explored the foundational underpinnings of security establishment – trust and elaborated on the protective mechanisms prevalent in today's security domain.

1. Introduction

Security has become such an important issue for modern systems that it is now a “must-have” rather than a “nice-to-have” feature. And since security plays such a fundamental part of the design cycle, it would be helpful to first think of security as based on the concept of trust. This is so when we talk about security, we may stay focused on its basic tenet of preserving trust, instead of being distracted by the numerous subcategories and tangential issues that are encompassed under the broad subject of “security.”

A secure system is one that users trust to carry out the tasks assigned to it, while ensuring the confidentiality and integrity of any sensitive data it has been entrusted to process. Furthermore, a secure system trusts that authenticated users will behave in a safe manner and not endanger the system through malicious tampering. To operate in a secure environment, a system depends on starting from a trusted initial state, often implemented using a secure boot flow upon power up.

A trusted computing platform relies on a secure supply chain, using authentic components and blocking the insertion of unauthorized (hardware) Trojan horses while the system is being built. Secure communications are based on mutual trust amongst the parties involved. In short, without trust, there can be no security. This trust must be protected from various challenges and threats, which will only increase over time, to maintain system security.

1.1. Challenges and Threats

The past 30 years of the Information Age can be divided into three stages: the Internet of Computers, the Internet of People, and the Internet of Things. In *Figure 1-1*, the number of connected items has rapidly grown when moving from one stage to the next [4].

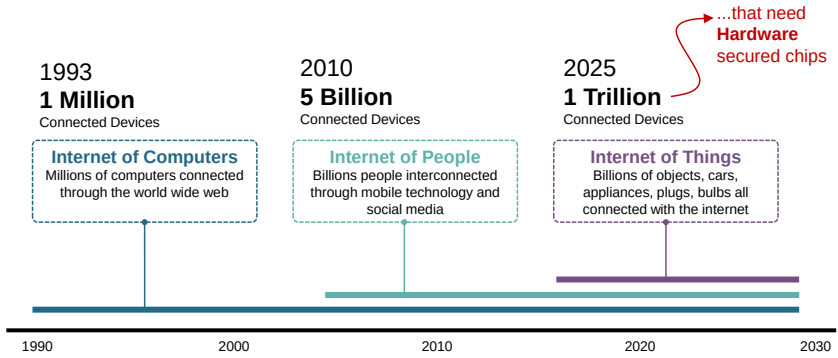


Figure 1-1 Growing connections at threat to security.

What has brought about this drastic change? The first transition was mainly due to the switch from manually-connecting networks to automatically-connecting ones, which increased the number of connected devices from 1 million to 5 billion [5]. With the rising demand for automation, the predicted number of connected devices (and people) will increase to 1 trillion by 2025 [6]. It is easily seen that along with the growing number of connected items, there will be a corresponding growth in the volume of data flowing on the Internet, emphasizing the importance of information security. We see that establishing and protecting trust as we move further into this third “Internet of Things” stage will be a challenging task.

In the Internet of Things (IoT) stage, we can examine an illustration of the threats to security and trust that an AIoT or AI plus IoT device will face. Smart AIoT devices, such as the one pictured in *Figure 1-2*, are subject to many points of attack, targeting the training data, input data, inference model, inference results, and the device itself:

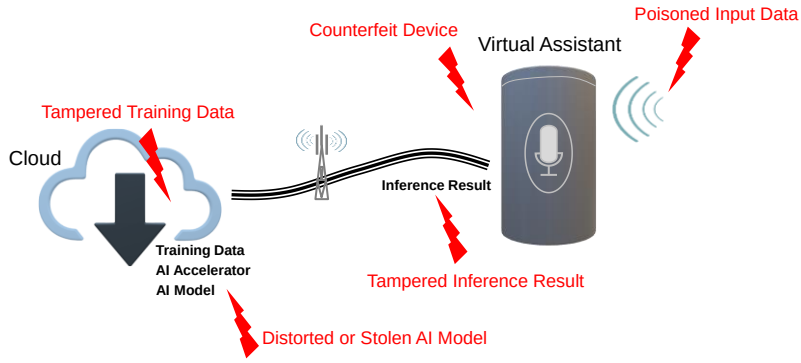


Figure 1-2 Threats to AIoT terminal devices.

Starting in the cloud, where the training data, AI accelerator, and AI model are stored, the training dataset is an inviting target for attackers who wish to tamper with the inference results produced by the AI model by influencing its training. If the attack is unable to modify the AI model through malicious manipulation of its training data, they may choose to directly alter the inference results, either at the source or during transmission of the results from the cloud to the edge nodes, such as a voice-enabled assistant. Even if the inference results arrive unmolested at the destination node, the individual device may be a counterfeit, or is authentic but was built with counterfeit components that were inserted into the supply chain undetected. If that was not enough, even input data at the

edge needs to be protected from poisoning, yielding inaccurate inference results. Finally, since a fully trained AI model represents many man hours of training, it becomes a most tempting target to steal for competing device/AI creators to take an unethical, but time-saving shortcut through the development process.

After seeing the rising challenges of securing an almost exponentially growing number of connected devices and the numerous attack surfaces facing modern AIoT devices, it comes as no surprise that security architects for this current Information Age have their work cut out for them.

1.2. Establishing and Building Trust

As we understand how important trust is to the security of a system, as well as the growing challenges and threats to that trust as we move forward, the question remains as how we establish and build the trust needed to ensure security. When we talk about security, it is natural that the first thing we think of is cryptography, particularly ciphers and how they are trusted to keep messages safe from prying eyes, as seen in *Figure 1-3*.

For symmetric ciphers, a plaintext message that needs to be sent to a recipient over a non-secure channel needs to be encrypted into ciphertext so that only those members with the encryption key may decipher the coded message into a readable one. So long as the crypto engine, or cipher, is of sufficient strength and the encryption key does not fall into an unauthorized party’s hands,

those members with the right key may trust that their secret communications remain private.

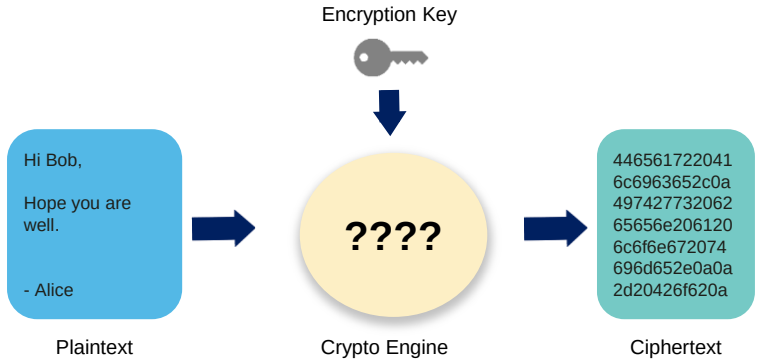


Figure 1-3 Cipher for message security.

Ciphers, including both symmetric and asymmetric algorithms, as well as other cryptographic tools such as hashes, key derivation functions, and digital signatures, are utilized to establish the trust required for a secure platform. However, this trust must still be built upon a foundation that is inherently trusted.

For example, if there is a person named Ted in a circle of friends who has been deemed trustworthy by everyone, then Ted may act as the foundation of trust for that circle. One day Bob introduces a new friend to the circle named Alice. After her introduction to the other members, there may be a conversation between the others that goes like: “Why should we trust Alice? Because Bob trusts Alice…and Bob is trusted by Ted…who is trusted by everyone.” A system’s “Ted”, or Root of Trust (RoT), has the highest level of trust

in a system and is the basis of trust. Because a RoT can be used to authenticate or verify other parts of the system, but cannot authenticate itself, a RoT must be implicitly trusted. Once a RoT is established, it is used to build the chain of trust that allows for trusted operations to take place on the platform, as seen in *Figure 1-4*.

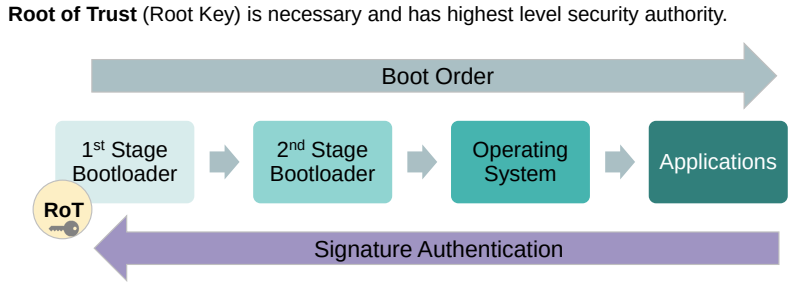


Figure 1-4 Chain of trust concept.

The Root of Trust safeguards a special service key, which protects a district key, which keeps the data encryption key safe, which is the key used by the cipher to protect confidential data. Such a chain of protection is known as a Chain of Trust, with the Root of Trust as its always and implicitly trusted basis.

Thus, with the RoT establishing trust in a system and cryptography acting to build out that trust to cover the entire secured system and its operations, confidential computing is realized. The next section takes a deeper dive into more details of cryptography and how it builds trust in a secure system – trust that is first established with the Root of Trust.

1.3. Five Primary Cryptographic Functions

Cryptography supports secure communication and provides several functions that are crucial for protecting sensitive information. Its purpose is to enable the transmission of information in a way that is protected from unauthorized access, modification, or interception. The five primary functions of cryptography are confidentiality, integrity, authentication, non-repudiation, and key exchange [7]. This section will briefly introduce each of these functions.

Confidentiality is used to preserve data from leakage, so only those authorized to access it can do so. Regarding messages, privacy ensures that only the sender and intended recipient can read a private message. Confidentiality establishes the trust that system data is secure and has not been stolen.

Authentication is used to prove an assertion, most often when verifying a user's identity in a security context. The user asserts they are whom they say they are, then the process of authentication establishes the trust that the user has claimed is true.

Integrity checking confirms that data has not been altered. With regards to messages, integrity reassures the receiver that the received message is the same as the one originally sent. Integrity establishes the trust that system data and message have remained intact and untampered with.

Non-repudiation ensures that the identity of the data creator or message sender is irrefutably linked to the data or message. It also provides evidence of the integrity of the data or message. By establishing trust that the data or message is authentic and originated from the person or entity whose identity is associated with it, non-repudiation helps prevent disputes over the origin or validity of information.

Key exchange is a method to securely pass keys between different parties. Key exchange establishes the trust that a shared key remains safe from leakage and theft during transit.

The above-mentioned authentication and non-repudiation are two functions that are often confused, but fundamentally they are two related yet distinct concepts in cryptography. Authentication is about ensuring that the user or system is who they claim to be, and to prevent unauthorized access. While non-repudiation is about providing evidence that a message or transaction was sent by a particular user or system and has not been altered since it was signed.

In essence, authentication is necessary to establish trust between parties and prevent unauthorized access, while non-repudiation is necessary to establish trust and prevent disputes over the validity of actions or information. The five primitive functions of cryptography are summarized in *Figure 1-5*, along with brief descriptions for each one.

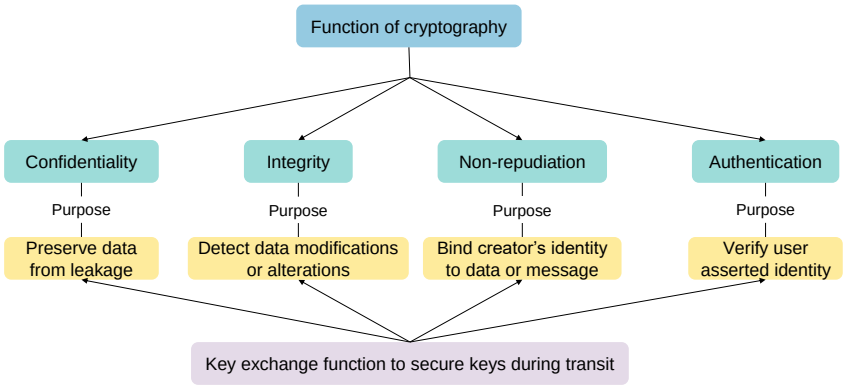


Figure 1-5 Five primitive functions of cryptography.

1.4. Organization of this Book

As we understand that trust is at the root of security, this book intends to guide readers into the concept that using a hardware solution to establish and enhance trust in a system is necessary nowadays. The focus of this book is on PUF-based (Physical Unclonable Function based) hardware security solutions.

Chapter Two examines the broad subject of “security”, including a discussion of current security trends, transitioning to the roots of trust in Chapter Three, which covers PUF-based roots of trust and how they enhance traditional security systems. Starting from Chapter Four, the focus shifts to hardware security solutions that have been commercialized, describing PUFsecurity’s basic hardware root of trust called PUFrt. Moving onto Chapter Five, it will cover the PUFcc secure crypto co-processor. Each of these two chapters on PUFrt and PUFcc includes a discussion of design

philosophy, architecture, functional blocks, and the operations of these two foundational products. The final Chapter Six goes into example applications of PUF-based solutions in the fields of artificial intelligence (AI) and the Internet of Things (IoT).

2

Hardware security has now emerged as a trend. Serving as a robust backbone for software, it not only safeguards the whole system but also enhances the security throughout the entire product life cycle. To swiftly grasp the requirements of hardware security in various application domains, we introduce some third-party testing, verification, and international standards on the current market.

2. Trends in Hardware Security

Initially, hardware security took a backseat to software security as hardware was assumed to be more difficult to tamper with, such as when inserting malicious elements, like a Trojan Horse. As more attention was given to hardware security, the focus was on individual problems, such as side-channel information leakage or the addition of hardware cryptographic accelerators.

Now, hardware security is starting to take a more systematic and holistic view, accounting for how hardware interacts with software, as well as how security is handled over the entire lifetime of a system.

2.1. Hardware Security as a System Requirement

As a sign of the new normal for hardware/system security, Microsoft has added the requirement that systems must have an installed trusted platform module (TPM) to be able to install its latest Windows 11 operating system [8]. Recognizing that software alone cannot completely secure a system and its operations, Microsoft is adopting the Trusted Computing Group's TPM 2.0 concept and making it a "must-have" for all users who wish to use the latest version of Windows. TPM is a security chip that stores sensitive information and keys and provides hardware-based security functionality.

The features required by Microsoft include:

Encryption authentication: TPM can be used to authenticate encryption keys to ensure they are issued by trusted entities and remain intact when used.

Encryption protection: TPM can be used to protect encryption keys and sensitive information to prevent unauthorized access or external attacks.

Secure boot: TPM can be used to verify the integrity and correctness of the boot process to ensure that the operating system and applications run in a secure environment.

Tamper resistance: TPM can be used to detect any changes or tampering of system components and notify system administrators for necessary actions.

Key management: TPM can be used to store and manage user keys to ensure key security and prevent unauthorized access.

In addition to the requirements mentioned above, Microsoft also requires that the TPM chip have its own unique identifier, known as the Endorsement Key (EK). The EK is a unique cryptographic key that is burned into the TPM chip during manufacturing and cannot be changed. It serves as a unique identifier for the TPM chip and is used to verify the authenticity and integrity of the TPM chip.

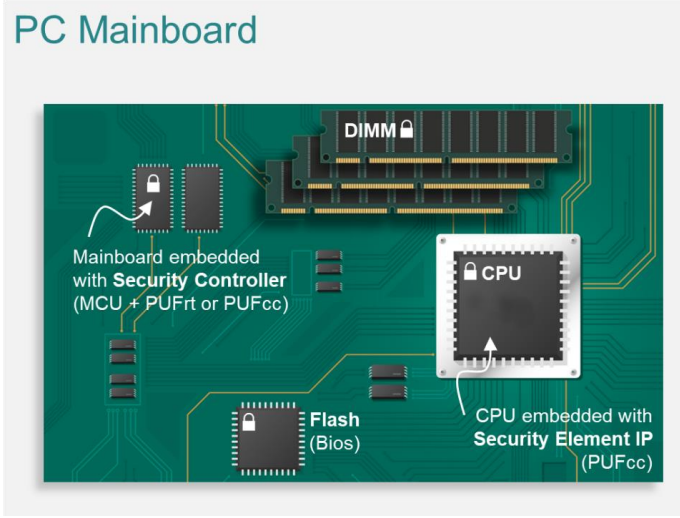


Figure 2-1 PC Mainboard to meet Windows 11 requirements.

An example implementation of a security element IP such as PUFcc to support the TPM for a PC system is shown in *Figure 2-1*. With an installed TPM, the system has a trusted element that it can rely on to ensure secure system operations by supporting such functions as device encryption, measured boot through Windows Defender System Guard, system health monitoring, secure key storage, virtual smart card, and secure biometrics log in access.

Furthermore, if the architect wishes to keep the secure boot flow separate from the central processing unit (CPU), an embedded security controller, either PUFrt or PUFcc, can be optionally mounted as its own unit, completely siloed away from the security element (TPM). In this way, the secure boot flow is kept separate from normal processing, so that the system is guaranteed to

always start from a secure state, preventing any unauthorized root/boot kits from being loaded during the start-up process.

Now that Microsoft has set the precedent of specifying a TPM as part of its minimum system requirements for running its latest operating system, it would not be unexpected to see other software/operating systems follow suit. Soon buyers will be just as concerned about a system's security elements as the amount of pre-installed RAM.

2.2. Data Security at Every Stage

Just as it is not enough to think of protecting the hardware and software of a system separately, but rather as a whole combination, all three states of data: "at rest", "in transit", and "in use", need to be considered when thinking about system security. Traditionally, ciphers have been put into use to protect data at rest (in storage) and in transit (as a message), with their ability to encrypt plaintext into ciphertext and vice versa. But as systems and hackers grow in sophistication, the potential total surface attack area rises along with the creativity of attackers, leading to security architects needing to find methods to protect the third state of data in use, as only covering two out of three states of data will leave a gap in total system protection.

Protecting data at rest, in transit, and in use requires a combination of technical controls, policies, and procedures to ensure the confidentiality, integrity, and availability of the data. The measures that are usually implemented to do so include "Encryption",

“Access Control”, and “Isolated Environment”. Companies such as Arm have come up with solutions like Confidential Compute, which includes their concept of Realms, to only allow users with the proper privileges to access confidential data, keeping data safe from theft and modification by unauthorized users [9].

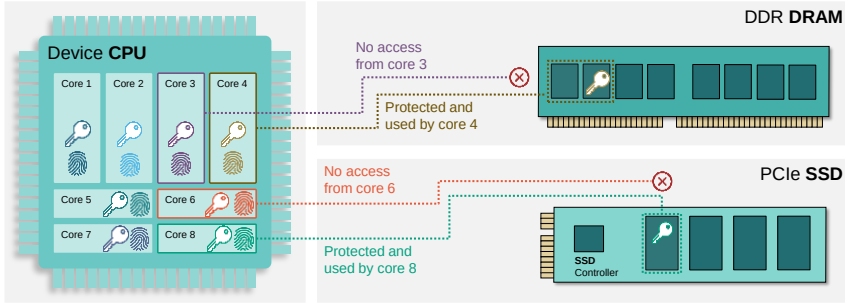


Figure 2-2 Data access restricted by core.

A similar concept is illustrated in *Figure 2-2*, each processor core is assigned a unique ID (silicon fingerprint) and granted access to individual blocks of data based on each core’s ID. The data is encrypted in any state and can only be accessed and decrypted with the corresponding ID and key. For example, since the data in the first block of the SSD is protected by a key derived from Core 8’s ID, only Core 8 may access and use that block of data. Core 6 cannot decrypt the data to use for its own. And since Core 8 is physically separated from Core 6, while the Core 8 data-in-use is being processed, it cannot be seen by Core 6 by any means. The above concepts of data security can be realized through the usage of security containers. Different applications are effectively isolated and operated in different environments, avoiding improper

access to working memory and stored data. Each core has its own unique secret and security features, which effectively achieve secure storage, transmission, and computation of private data.

2.3. Product Lifecycle Protection

The scope of security is expanding. Software is starting to require the backing of secure hardware elements. Data protection needs to account for when it is in use, and not just at rest or in storage. Similarly, the security requirements of a product's entire lifecycle now need to be considered, starting at the design stage, and going all the way to a product's end-of-life.

The various stages included are shown in *Figure 2-3*. Each of these different stages will require the support of varied functions to ensure security over a product's operational lifetime. In the design stage, the security architect will need to determine which cryptographic primitives need to be designed, bought, or handed over to an external unit for support. When it comes to manufacture the product, the director in charge of provisioning will need to examine the supply chain for any weaknesses to prevent any hardware Trojans, counterfeit components, or theft of the proprietary design.

Furthermore, the processes of deployment, provisioning, commissioning, user authorization, and secure updates all face their own sets of non-overlapping issues, making the task of protecting a product throughout its lifecycle a non-trivial task. Finally, when the product has reached its end-of-life, any private

keys and data will still need to be erased or zeroized before they can be safely disposed of or recycled.

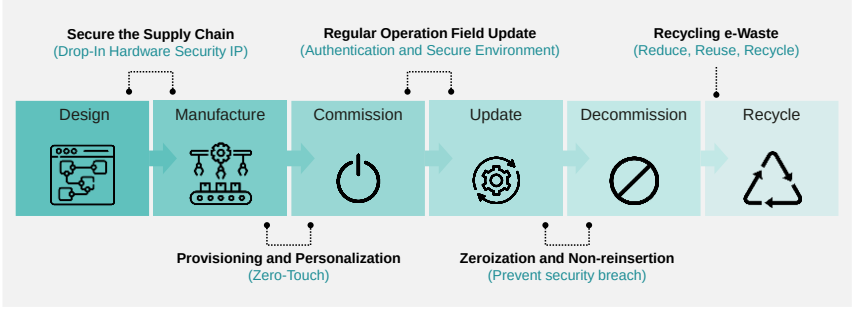


Figure 2-3 Product lifecycle: Design to end-of-life.

Keys play a critical role in cryptographic algorithms, especially for the ones in widespread use, such as those standardized by NIST. This is because the key is the only “secret” of a cryptographic algorithm, such as AES. The entire process and each step of how plaintext is turned into ciphertext and deciphered in the reverse direction through AES is public knowledge. Therefore, it is the private keys that are the source of security for those cryptographic algorithms that employ them. Once a key is known, anyone may convert from plaintext to ciphertext and back to plaintext using the AES cipher.

Thus, only the fact that an attacker does not have a private key is what prevents them from being able to decrypt an encrypted message. Therefore, keys are very important to any secure system, so that key security, key management, and the key

lifecycle warrant separate consideration from a product's lifecycle, as seen in *Figure 2-4*.

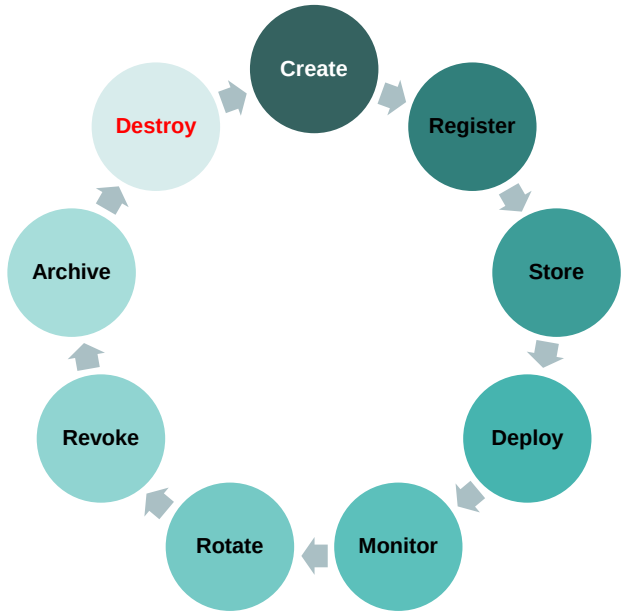


Figure 2-4 Key lifecycle: creation to destruction.

From the time a key is created until it is deemed unusable and destroyed, keeping it safe is paramount to a system's security. Thus, the most secure systems will have a key management setup in place to protect keys from tampering and exfiltration. As with the trend to protect a product over its entire lifetime, so will the protection of keys necessitate the same lifecycle focus, which should include a formalized key management system using best practices or protocols.

2.4. Increased Adoption of Standards and Certifications

As security becomes more and more important for current and future products, following best practices from the start of the design stage only makes sense. To facilitate the adoption of best-in-use practices, it is highly recommended to choose third-party certified solutions and follow industry standards. Since solutions certified by the same organization will use the same common set of criteria, it allows for direct comparison between the solutions and/or a grading scale, such as FIPS 140-2 levels 1/2/3/4.

In addition, certified solutions can help effectively save on development time and costs since designers do not need to start from a blank slate, as well as providing peace of mind that the solutions employed are correct and of robust quality. Finally, using certified components in a design will help speed up the final product's overall security certification, further shortening a product's time to market. Two such examples of certified security solutions, PUFrt by Riscure and PUFcc by PSA Certified Level 2 Ready, being integrated into an Arm-based system are shown in *Figure 2-5*.

Based on experiences from evaluating attacks on smartcards with the Common Criteria methodology, Riscure adopted the Joint Interpretation Library (JIL) attack rating methodology to examine and certify elements using the recommendations from the JIL Hardware Attacks Subgroup (JHAS). We see another example of standards, PSA certification, playing an important role in the case.

PSA Certified used the ANSSI CSPN framework to confer a PSA Level 2 Ready rating to PUFcc, which will speed the eventual certification of the larger, integrating Arm-based system that will use PUFcc at PSA Level 2 or 3 [10].

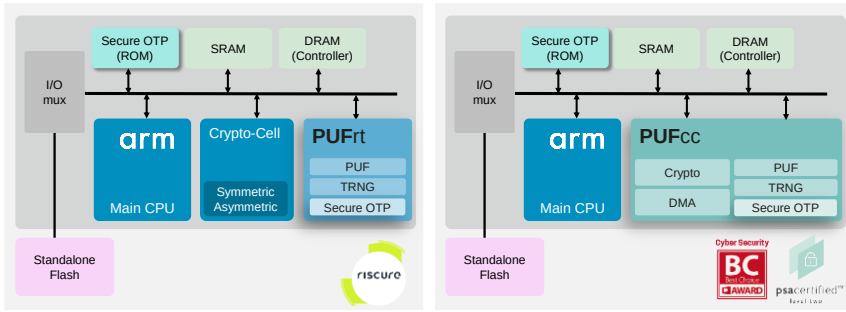


Figure 2-5 Certified solutions integrated into systems.

With the rapid increase in the number of connected devices that will need to be supported in the fields of artificial intelligence (AI) or the Internet of Things (IoT), it is as critical as ever to lean heavily on industry standards and certified security solutions to ensure a base level of security and common reference framework.

This fact has led to more and more governments requiring their official provisioners, through acts and regulations, to purchase equipment that meets a minimum level of security grading. We can at least prepare for new security challenges by using our current best practices and industry consortium standards to face them. Hence, an increased adoption rate of standards and certified security solutions is an inevitable future trend.

This book focuses on introducing how PUF-based solutions enhance hardware security. Here are some other recommended relevant standards and specifications to consider when designing products for readers' reference.

1. **FIPS 140-2** specifies the security requirements for cryptographic modules used by the US federal government and other regulated industries such as finance and healthcare. It requires that cryptographic modules meet rigorous standards for physical security. Products that pass level 2 or higher must include secure hardware [11].

2. **California SB-327** is a California state law that was enacted on September 28, 2018. It addresses the growing concerns about the security of internet-connected devices, also known as the Internet of Things (IoT), and the potential risks they pose to consumers' privacy and security. Especially important for the IoT field, each device should be able to be uniquely identified. This is related to the importance that ID authentication plays in security [12].

3. **Code of Practice for Consumer IoT Security** published by the UK government, provides guidance for manufacturers, service providers, and developers on how to ensure the security of internet of things (IoT) devices. It emphasizes that security cannot only rely on software. Security must be considered starting from the chip design stage. In addition, every IoT device should have its own key [13].

4. **SESIP** is a set of security guidelines and recommendations developed by the Internet Engineering Task Force (IETF)**.** It recommends that the root of trust be protected using physical security measures, such as tamper-evident seals, and that the private keys associated with the root of trust be stored in secure hardware modules, such as hardware security modules (HSMs), to prevent unauthorized access or tampering. The guidelines and recommendations in SESIP are widely adopted by vendors and service providers [14].

5. **NIST SP800-131A** is a publication that recommends which cryptographic algorithms and key lengths should be used for various types of security applications, such as data encryption, digital signatures, and key establishment protocols. It also includes recommendations for the security and management of cryptographic keys, including the use of key escrow and recovery mechanisms [15].

6. **NIST SP800-22** is a set of statistical checks designed to detect both intentional and unintentional biases in RNG output and to ensure that the RNG output is suitable for use in cryptographic applications. This publication also provides guidance on the selection and use of RNGs in various cryptographic applications, including key generation, encryption, and digital signature schemes [16].

3

The hardware root of trust is foundational to device security. Activated upon first power-up, it is crucial throughout the device's lifecycle. We begin with secure boot to explore why traditional Roots of Trust are no longer sufficient to safeguard modern systems. An effective root of trust should be grounded on hardware based on PUF.

3. Hardware Root of Trust

After an introduction to trust and security, we are ready to dive into the specifics of how security is established, starting at the source, the root of trust. Since trust plays such an important role in the five primary functions of cryptography, it is vital that trust in a system is based on an unshakeable foundation.

How does such trust become established? Looking at the example of online shopping, buyers might look up the reviews of a seller to get a feel for how trustable the seller is. But how does the buyer know that the reviews are honest and are not fake ones written by the seller themselves? Perhaps the buyer will search specifically for those reviewers who have written many different ones, or the shopping platform has certain reviewers that have been labeled as “trusted reviewers.”

Trust in a system needs to be established in a similar way, originating from a source that is always assumed to be trustable, from which the other components/software/firmware of the system may be checked for authenticity and then deemed trustable. This source of trust in a system is often called a “root-of-trust” or “RoT” – because it can be used to authenticate or verify other parts of the system but cannot authenticate itself. Therefore, a RoT must be inherently trustworthy. This brings us to the classic question "Can software act as a completed function root of trust?" As we all know, software is very flexible because of its re-programmability.

This enables smartphones to provide the many services and varied applications available today. But along with such convenience comes a variety of security concerns, as we often see reports of the latest malware on the news. Thus, due to the malleable nature of software, it is difficult to use software as an ever-unchanging "root of trust", especially since hardware is always the foundation that makes software work. Having a hardware root of trust is a necessary measure to enhance device security. Since hardware is much less easily tampered with, it is much better suited as the always-trusted component of a system used to verify the other components.

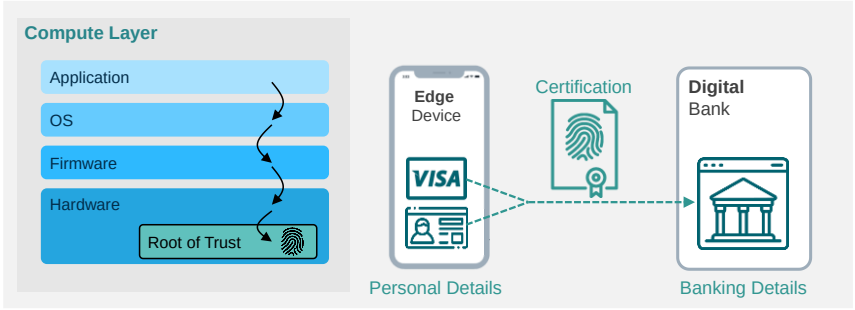


Figure 3-1 Hardware root of trust for compute layer protection.

As seen in *Figure 3-1*, the system's hardware is at the bottom level, upon which runs the hypervisor at the firmware level, upon which runs the operating system, and finally upon which the user interacts with the system through the installed applications. In this system stack, each layer relies on the layer below. For any two layers, the upper layer always assumes that the lower layer is

authentic and operating correctly and is only allowed to begin execution after being verified by the lower layer. Thus, it must be implemented from the bottom-most hardware level for a root-of-trust to be used most effectively in a system. Such roots-of-trust implemented at the hardware layer will hereafter be referred to as a hardware root-of-trust (HRoT).

3.1. Root of Trust Anchors Secure Boot

At this point, we understand that the Root of Trust should ideally be created in hardware, and it is used to authenticate the other parts of the system, building trust from the ground up. But how does this work in practice? How to use hardware to ensure the security of software and the operating environment? This process of starting with the HRoT to authenticate the boot code and to make sure the system has not been tampered with is called “Secure Boot.”

The goal of secure boot is to safely bring up the system into a known and secure state, using authentic and untampered versions of the boot code and operating system [17]. *Figure 3-2* illustrates the concept of the secure boot process. The user may run their applications on the system with the confidence that they will perform as expected. To do so, a chain of trust is established with the hardware root of trust anchoring this chain which is demonstrated at the very left of the diagram.

Starting from the HRoT, it authenticates and validates the next stage of the secure boot process, which is loading the boot code in this example, through cryptographic methods such as integrity checking and digital signature verification. Once the boot code has been established to be authentic and unmodified, it can be loaded by the system. Afterwards, it may start the same authentication and validation process with the next block in the secure boot process, which is the third block stating OS Loader. In this way, so long as each stage is completed successfully, the system will move down this set of chained stages until it reaches the point when user applications can begin to execute.

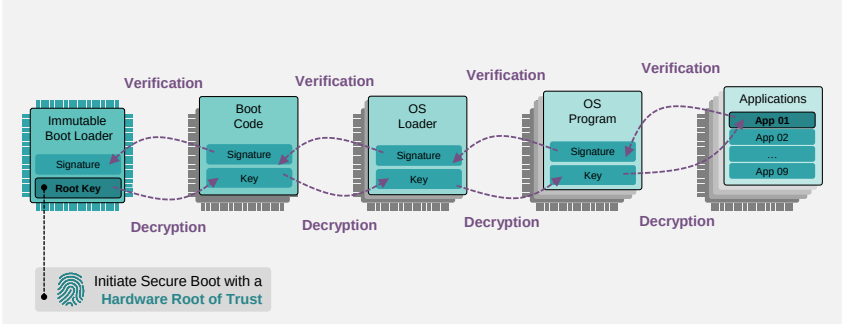


Figure 3-2 Secure boot with chain of trust.

Similar to how we can use our individual fingerprints to protect our privacy, the hardware root of trust can protect and verify software, data, and the system. But what is the equivalent of a fingerprint for hardware? How can we ensure that a silicon fingerprint is unique like a human fingerprint? This means that every chip or die will need its own inborn and unique identity.

There is a silicon structure called a Physically Unclonable Function (PUF) that is the most suitable technology available to us for representing a silicon fingerprint [18]. A PUF takes advantage of small, random variations in the wafer fabrication process which are unpredictable and unrepeatable from die to die. From these random variations, a PUF will generate an inborn, reliable, unique, and physically unclonable "fingerprint." *Figure 3-3* illustrates some of the functions that are supported by the inclusion of an onboard PUF.

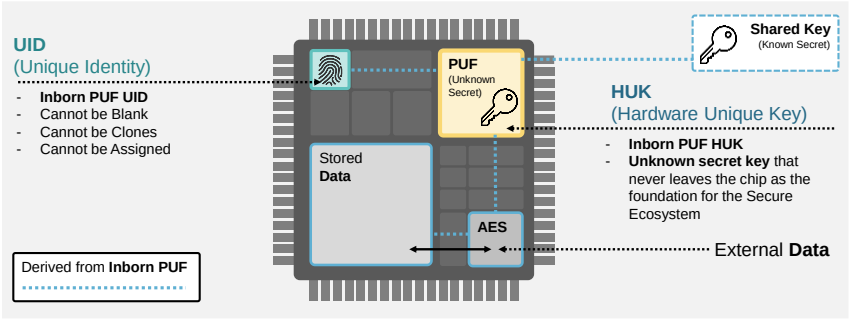


Figure 3-3 PUF-supported functions.

The unique identity (UID) of each die or chip is derived from the PUF without any outside intervention, such as through key injection Thus, there is no need for additional steps after final testing of the dies to assign an ID to each device.

In addition, the hardware unique key (HUK) used for deriving the other important system keys may also be extracted from the PUF – minimizing the risk of key leakage by being able to internally

generate the key ladder through an appropriate key derivation protocol. As such, there are some real advantages to selecting a root of trust with an included PUF, as opposed to one without a PUF.

3.2. Differences Between PUF-based and Traditional Solutions

The main differences and benefits of a PUF-based solution come from the inclusion of a PUF. First, when a PUF is included as a standard feature for a security solution, the root key and/or hardware unique key (HUK) generation can be handled by the PUF rather than using a conventional key injection process that a traditional solution must rely on. Just switching from an external key provisioning service to an internal flow will save time and money for every die that requires the assignment of a unique and unguessable root key.

Second, because the root key is sourced from the PUF in a PUF-based solution, the anti-tampering protections of the PUF automatically extend to cover the root key, starting with the fact that a PUF is non-writeable. Contrasted with the standard solution of storing a root key in a generic non-volatile memory (NVM), such as when using a traditional eFuse array, a PUF-derived root key will be less vulnerable to tampering, modification, or unauthorized read access.

However, when a PUF is paired with a NVM, such as a one-time programmable (OTP) array, the inborn randomness of the PUF

may be used to create a PUF-protected OTP (Secure OTP) that allows for the scrambling of the physical OTP addresses. As seen in *Figure 3-4*, such a PUF protection scheme allows for the same data to be stored in the OTP, but at different physical locations for each die.

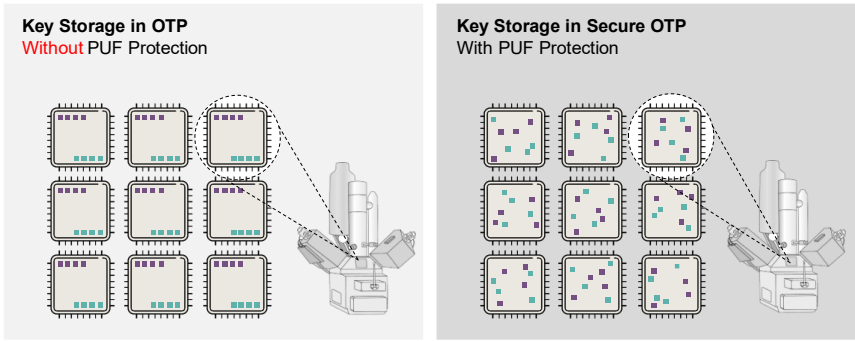


Figure 3-4 PUF protection scrambles physical addresses for Secure OTP.

Third, when comparing the secure storage of a root key on a PUF versus on a traditional eFuse array, eFuses are especially vulnerable to reverse engineering, even without gaining control of the array. Because physical access alone will allow anyone with a powerful enough microscope to read out the entire eFuse array through visual inspection only. As a matter of fact, the eFuse array does not even need to be powered on for this visual inspection to happen.

Fourth, the true random characteristics of a PUF will improve the quality and performance of a True Random Number Generator (TRNG) when included as part of its design. To start with, there is

no need to wait for a random seed collector to gather enough entropy before the TRNG is allowed to begin, as random PUF values will be ready for use immediately after the chip is powered on. After initial random values are created by the main random tumbler engine, they may be further refined using a dedicated entropy pool derived from the PUF that enhances the final TRNG output. This is what is meant by “entropy refined engine” (as seen in *Figure 3-5*), which is done without the need to implement a specialized secure hash engine which has the additional benefit of speeding up the random value generation wait time when compared to employing a full hash engine for refinement.

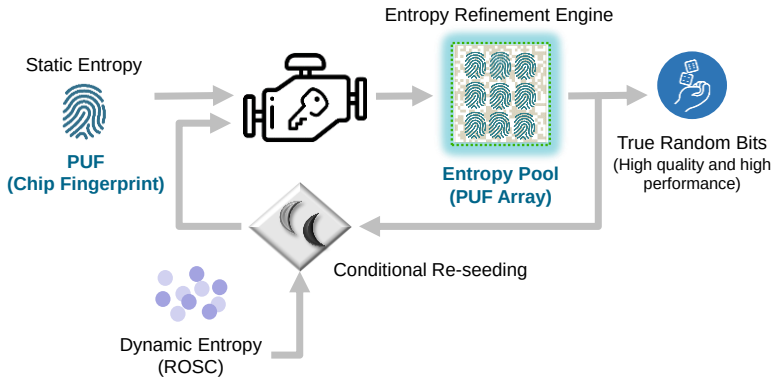


Figure 3-5 PUF enhances TRNG quality and performance.

Fifth, and finally, a PUF-based security solution combines both analog and digital design elements into one integrated unit that is protected by built-in digital and analog anti-tampering designs It is opposed to the digital-only anti-tampering protections that traditional security solutions must rely on. A traditional, digital-only

security solution will still require the addition of extra memory, which in turn requires separate protection. Since separate protection is not normally provided by a standard memory IP, it is necessary for the system integrator to do so. Note that such protections for the add-on memory of a security solution are not optional. Castle walls may protect the valuables held within, but any food supplies stored outside of those walls are still vulnerable to attack.

3.3. Root of Trust Wishlist

Finally, to guide the acceleration of the adoption of secure chips, let’s focus on what features a secure RoT should have. This section will provide the features and a summary of what we have learned so far about roots of trust. The RoT “wish list” of features is as follows:

1. Based in hardware
2. Included PUF
3. Secure Storage
4. High quality TRNG entropy
5. Anti-tampering designs certified by third party

A root of trust is the implicitly trusted element of a secure system; therefore, it should reside at the base level of a system - the hardware level. If a root of trust is located anywhere else, for example in firmware, then if the beneath system stack layer in hardware is corrupted, then the firmware root of trust would have no way of confirming that the hardware layer is trustworthy.

Adding a PUF to a design brings in a host of nice benefits, especially when integrated into a root of trust. A PUF with near ideal qualities is an invaluable resource for roots of trust that require a quality and static entropy source. A root of trust is entrusted by the system to protect its stored secrets. Thus, having a built-in secured storage is a must-have feature. Robustness over a wide range of operating conditions and resistance to attempts to reverse engineer its data are assumed to be standard on any storage element that is labeled as “secure.”

A PUF is a good place to begin building an entropy source. However, a PUF is limited since it is a static entropy source based on a fixed amount of entropy, such as process variations between die to die. Therefore, it naturally follows that a root of trust should also include a dynamic entropy source, such as a true random number generator (TRNG) or deterministic random bit generator (DRBG).

In summary, roots of trust are foundational components of a secure system that establish trust in the authenticity and integrity of a system's operation. They are typically implemented in hardware and provide a secure execution environment that can be trusted to perform critical functions, such as authentication, encryption, and decryption. A system’s trusted root should not be able to be modified or tampered with by bad agents because it is used to verify other parts of the system, while not being able to verify itself. Thus, roots of trust with anti-tampering designs are highly desirable, and compared to traditional methods, PUF-based RoTs with third-party certification offer more advantages.

4

PUFrt is a prime example of a hardware Root of Trust. It offers a chip fingerprint (PUF), PUF-based TRNG, and secure storage with highly anti-tampering features. Its straightforward design and user-friendly functions speed up market entry while enhancing security. PUFrt plays a central role in the generation and protection of key operations.

4. PUFrt

PUFrt is the root of trust from PUFsecurity Corporation. It packages the basic but necessary RoT functions into an integrated IP solution [19]. This allows customers to tailor a security system to their exact needs by using PUFrt as a base and adding only the security functions they require. A conceptual diagram of PUFrt in *Figure 4-1* illustrates the essential functions of a root of trust included in this IP.

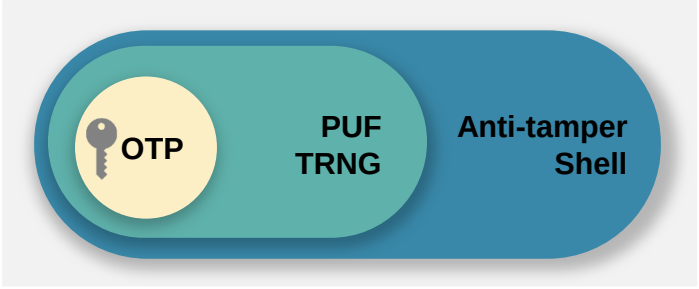


Figure 4-1 PUFrt: PUF-based root of trust.

4.1. Design Philosophy

PUFrt is designed to be integrated into a larger system as its root of trust. When the concept was first formed during initial product development discussions, it quickly became clear that the main objective was to distill the essential components of a root of trust into PUFrt. With that goal in mind, PUFrt is created using these

four guiding principles. The following content will introduce these four principles one by one.

1. Protect The Core
2. Use Multiple Levels of Protection
3. Make It Easy
4. Focus On Entropy

4.1.1. Protect the Core

To be a root of trust, the trusted foundation of a system, PUFrt's top priority is the protection of its secure core, the OTP memory. A system's security relies heavily on its root of trust, yet a system cannot verify its own root of trust – no earlier or more basic security element comes before the root of trust that can perform such a task. Hence, a RoT is always assumed to be secure and unaltered from its last authorized write until the end of a system's lifetime.

Consequently, it is important that "root" data stored in PUFrt's core memory array must be protected from unauthorized access, including reads and writes. This data could be in the form of the hardware unique key (HUK) of the device, or the hashed digest of the system's boot code, stored alongside the digital signature guaranteeing the authenticity and integrity of this vital piece of the system's secure boot process. Regardless of the form of this trusted data, it needs to be secured within the core of a RoT, inside of PUFrt.

4.1.2. Use Multiple Layers of Protection

The tools available nowadays to bad agents are many and varied and can be non-invasive, semi-invasive, or invasive in scope. Therefore, it is expected that the protections deployed by a root of trust to prevent intrusions and exfiltration need to be just as varied. In addition, these protections would best be arranged in multiple layers, much like a strong castle defensed with multiple walls of encirclement. The approach ensures that even if one wall is breached, the overall security is not completely compromised.

PUFrt protects the core one-time programmable (OTP) memory array with protection derived from both static and dynamic entropy sources. It wraps the entire intellectual property (IP) in layers of anti-tampering countermeasures. By protecting the core with multiple layers, sensitive keys and data are safely stored inside PUFrt.

4.1.3. Make It Easy

Not only should a root of trust be secure and well-protected, it should also be easy to implement and use for authorized accesses. With the era of the Internet of Things (IoT) upon us, security is an important and continually growing design issue to consider, especially for connected devices. To encourage greater adoption of added security measures, such as a root of trust, PUFsecurity has designed PUFrt with these twin goals in mind, ease of use and integration. PUFrt supports the most common ARM peripheral bus protocols for easy use and integration into ARM-compatible system architectures. A comprehensive design kit, including notes

for application usage and system integration, comes as standard with PUFrt.

In addition, a reference firmware/application programming interface (API) library is available for users who would like to shorten the setup time to get PUFrt up and running after integration. Finally, moving between fabrication plants (fabs) and process nodes is a snap, due in part to the long history of PUFsecurity's parent company eMemory. During its more than 20 years in operation, eMemory Technology has formed partnerships with all major fabs worldwide. PUFsecurity's IP designs can leverage eMemory's pre-qualification on over 450 process platforms, enabling PUFsecurity to bring the designs to market more efficiently.

4.1.4. Focus on Entropy

Most of the commonly used cryptographic algorithms today are public knowledge, that is, anyone can see the inner workings of even the most secure encryption schemes, such as the modes of the Advanced Encryption Standard (AES) [20]. Therefore, the one thing that preserves the security of such a publicly known cryptographic algorithm is keeping the secret key known only to the authorized users. Moreover, the secret key needs to be random and very hard to guess – that is, it needs a sufficient amount of entropy to ensure the safety of any data that will be encrypted using that secret key. Modern cryptography depends heavily on entropy, which is why developing high-quality sources

of entropy is critical during the design process of PUFrt. Even the strongest algorithm cannot protect data when using a guessable key, or one used for multiple applications.

PUFrt provides two types of high-quality entropy, both static and dynamic. Static entropy is derived from the PUF at the heart of PUFrt, a novel and reliable fixed entropy source suitable enough to derive a device's all-important hardware unique key (HUK). The HUK is used to generate the other important system keys by using a secure key derivation function (KDF).

The other included entropy source of PUFrt is the true random number generator (TRNG) from which dynamic entropy can be extracted for the use of temporary keys, initialization vectors, nonces, and seeds for deterministic random bit generators (DRBGs). It also protects the operation of the system, including randomizing power signatures, read/program operations, and masking intermediate values that are generated during essential cryptographic operations such as encryption.

Finally, since entropy is important to the security of any system, the entropy sources must be of the highest quality. One way to ensure this is to perform checks using standardized tests, such as applying the National Institute of Standards and Technology's (NIST) SP800-22 statistical analysis suite [21] for determining the quality of randomness, as well as the SP800-90B as the qualification for a dynamic entropy source for DRBGs [22], [23]. Naturally, both the static PUF (SP800-22) and TRNG (SP800-90B)

of PUFrt have passed such qualification tests with ease, measured from different silicon samples over a wide range of process nodes.

4.2. Architecture of PUFrt

The PUFrt is composed of two main parts, the hard macro in a full-custom layout (GDS) format and the soft macro in RTL design. The hard macro is sourced from eMemory, bundling their anti-fuse OTP and quantum tunneling PUF into one unit. PUFsecurity then adds digital logic blocks and a standard bus protocol interface which is implemented in RTL code to create the PUFrt hardware root of trust.

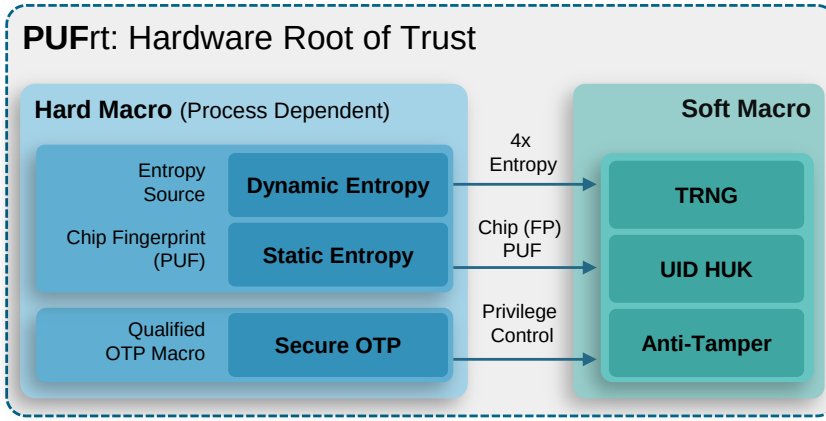


Figure 4-2 PUFrt architecture.

In *Figure 4-2* showing the architecture of PUFrt, the hard macro contains the OTP and PUF arrays along with their associated decode, sense amp, charge pump, and control logic circuits. The

hard macro is mostly an analog block and is delivered as a hardened (layout completed) module to the fabrication plant. At the fab, the hard macro is merged with the synthesized digital logic to create the top-level layout block of PUFrt, after which it may be integrated into the design of a larger system, such as an SoC. The hard macro module of PUFrt is composed of eMemory's NeoFuse (OTP) and NeoPUF (PUF) IPs.

NeoFuse is the ultimate solution for embedded Non-Volatile Memory (NVM) in complementary metal-oxide semiconductor (CMOS) logic or logic-derived processes. Targeting those markets and applications with the most efficient approaches, NeoFuse delivers outstanding performance with a highly reliable, embedded non-volatile memory available in a wide variety of densities and flexible configurations, over various CMOS technologies.

NeoPUF is a weak PUF which challenge-response-pairs is limited, capable of generating true random bits, suitable enough to act as a silicon fingerprint. NeoPUF has the near ideal PUF characteristics of 50% HW (Hamming-Weight), 50% Inter-HD (Hamming-Distance), 0% Intra-ID HD and 0ppm BER (Bit-Error-Rate). NeoPUF passes both the NIST SP800-22 and NIST SP800-90B independent and identically distributed (IID) statistical analysis test suites for randomness [22], [23].

The digital portion consists of interfacing logics connecting the system, hard macro, and PUFrt logic blocks, a true random number generator (TRNG), and the control logic that implements

the functions of PUFrt. This digital portion is written entirely in RTL code, whose corresponding graphic design system (GDSII) data is created by following an industry standard digital design synthesis/place and route flow.

4.3. PUFrt Block Descriptions

At the heart of every PUFrt is eMemory's hard macro featuring the NeoFuse OTP memory and NeoPUF Quantum Tunneling PUF, which provide both secure storage and a near-ideal source of static entropy. Building on top of this base, PUFsecurity adds functional blocks to provide system/hard macro interfacing, true random number generation, unique identifier generation, and encrypted storage capabilities. The functional blocks of PUFrt are illustrated in *Figure 4-3*. These blocks have been created to implement the "wishlist" as listed in the previous section 3.3. The following sections cover these five blocks in more detail.

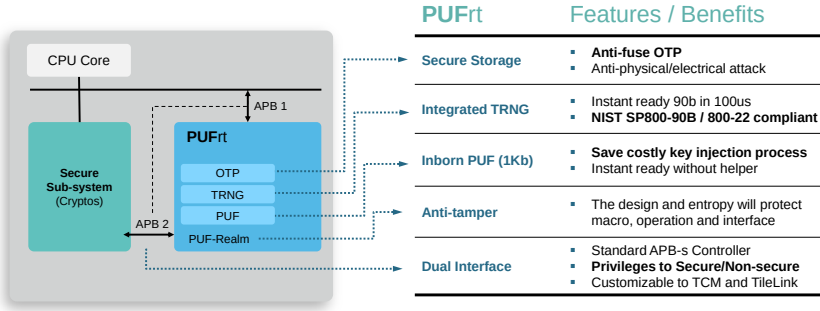


Figure 4-3 PUFrt block diagram.

4.3.1. Dual Interface

The dual interface block allows PUFrt to be placed on any AMBA-compatible peripheral bus, such as the Advanced Peripheral Bus (APB) for easy system integration [24], allowing memory-mapped register access of PUFrt. Unifying the communications between the system and PUFrt IP, the dual interface allows users to control over all IP functions, including access to the hard macro OTP/PUF memory arrays.

4.3.2. TRNG

The true random number generator (TRNG) complies with the NIST SP800-90B standard. Based on dynamic and static entropy sources driving a data whitener and randomness extractor utilizing a PUF-based refinement engine, this PUF-based TRNG's bitstreams have passed the NIST's statistical test suite for randomness, as described in the document SP800-22.

4.3.3. Inborn PUF

The quantum tunneling PUF acts as the entropy source for the unique identification (UID) module. Because keys are created within the quantum tunneling PUF itself and therefore within the boundaries of PUFrt, there is no need for key provisioning or injection using a specialized hardware secure module (HSM) and/or access to a secure clean room environment. Thus, the absence of key injection mitigates the risk of key leakage. Furthermore, as this QT PUF has been silicon-proven on many

different process nodes, as well as passing NIST's SP800-22 statistical analysis suite for randomness, this inborn PUF can provide not only a UID for the system but also the HUK. When the inborn PUF is paired with a cryptographically strong key derivation function (KDF), all important device keys are derived from the HUK, which includes data encryption keys (DEK), key encryption keys (KEK), secure storage keys (SSK), and other important private keys. Thus, the HUK is also known as the "root key."

4.3.4. Secure Storage

PUFrt's secure storage module utilizes a PUF-based data encryption/decryption module enabling secure storage of private keys or sensitive data on the OTP array. When enabled, it transforms programmed data into ciphertext through data masking or shuffling, depending on the OTP configuration. The process is reversed during read operations with minimal access penalties. In addition to data encryption, PUFrt will scramble the physical addresses using random values derived from the PUF.

4.3.5. Anti-Tampering Designs

Comprehensive tamper-proof features have been implemented in PUFrt. The hard macro portion of the PUFrt has already passed third-party security assessments and is certified for set-top box (STB) applications, which is a market known for its high-security requirements. Along with the fundamental and reliable analog-based tamper-proof designs of the hard macro, digital-based anti-

tamper designs have been included with PUFrt, offering protection against invasive, semi-invasive, and non-invasive attacks.

The analog-based designs protect the memory cells, memory arrays, analog circuitry, and signal routing from such threats as optical (microscopy) reverse engineering, photo emission detection, power analysis, fixed ion beam (FIB) metal cuts/adds, and malicious chip probing attacks. Digital-based anti-tamper designs keep data in transit safe along shared system buses, shield exposed pins from fault injection, add access management rights that can be defined at the granularity of a word (32 bits), as well as perform address and IO scrambling. The static and dynamic entropy sources of PUFrt play an important role in these digital tamper prevention designs, highlighting once again the need to include both types of entropy sources in a root of trust.

4.4. PUFrt Operations

The operations supported by PUFrt can be broken down into five categories:

1. Data Access
2. PUF Setup
3. Settings/Options
4. Access Permissions
5. Hard Macro Specific

These categories cover the total range of operations of PUFrt. Data access operations move data in and out of PUFrt, to and from the OTP/PUF/TRNG. PUF setup operations are run before the built-in PUF can be used. Setting/option operations enable and disable the different PUFrt options, as well as turn on either testmode or PUF lock, which locks users out from accessing non-user mode operations.

Access permission operations allow users to turn on write protection for selected words or collections of words. The zeroization function to destroy all stored data in PUFrt is also included in this “access permissions” category as a permanent means to deny access to PUFrt data. Finally, hard macro specific operations allow users to directly access the hard macro, which includes putting the hard macro into power savings mode, as well as on/off cell margin read for screening out initial bad bits.

4.4.1. Data Access

Data access operations move data into and out of PUFrt. The OTP memory can be written to and read from when the permission settings allow it. A UID may be read from the on-board PUF when the permission settings allow, but it is forbidden to write any external data into the PUF. The TRNG can be accessed to provide random numbers after the TRNG setup time has elapsed. Data into and out of PUFrt is moved in words of 32-bits in length. The three different types of data available from PUFrt are shown in *Table 4-1*.

Table 4-1 PUFrt data access operations

Location/module	Action
OTP	Read/ Write
UID/ PUF	Read
TRNG	Read

4.4.2. PUF Setup

A newly manufactured PUFrt needs to go through a one-time setup process to populate the built-in PUF with random values before it is ready to be used. Because the PUF is an entropy source for not only the UID but also for the TRNG and secure storage module, a PUF that has not been enrolled will affect the entire operation of PUFrt. These setup and enrollment operations are summarized in *Table 4-2*.

Table 4-2 PUF setup operations

Operation	Description
Quality Check	Check the quality of PUF cells (pre-enrollment)
Enrollment	Populate PUF with random values
Health Check	Check the quality of PUF randomness (post-enrollment)

PUFrt supports the following operations to ensure the correct setup of the PUF has taken place: PUF enrollment, quality check,

and health check. Enrollment is the actual process of taking the random, microscopic differences present after silicon manufacturing and turning them into random data that is stored in the PUF. PUF quality check is the operation that ensures the physical storage cells of the PUF have reliable data retention by screening out those cells initially showing questionable read margins. Finally, PUF health check is for verifying that the entropy of the enrolled PUF is at an acceptable level since it is the static entropy source for many PUFrt functions, including the TRNG, UID, and secure storage protection schemes.

4.4.3. Settings/Options

PUFrt has allocated several OTP locations to act as dedicated, permanent registers for option settings. These registers are considered permanent since once they are written, they cannot be reset to the factory default options. Five different options that may be set using these registers, also known as the “set flag” operations, are shown in *Table 4-3*.

Table 4-3 PUFrt option settings

Option	Description When Option Is Set
Program Protect	Enable Read-Before-Write Overprogram Protection
Testmode Lock	Disable Testmodes and Option Settings
PUF Lock	Disable PUF Enrollment and Zeroization
Shuffle Read	Enable Shuffle Read Operation
Shuffle Write	Enable Shuffle Write Operation

Program Protect is to enable “read before write” – meaning, when this option is enabled, any program operation will be preceded by a read to the target location. This is done to ensure that any programmed location will not see another high-voltage programming signal, to protect against overprogramming.

Testmode Lock is to disable those operations deemed to be non-essential for normal PUFrt usage, as well as to deny further user access to the PUF array. Testmode Lock also disables any further setting of options through the “set flag” operations, including Testmode Lock itself. Testmode Lock is recommended to be performed after the final testing of PUFrt has been completed, and before it leaves the factory, to guarantee the secure operation of PUFrt.

PUF lock is similar to Testmode Lock, but works on a narrower scope, affecting only PUF array operations. PUF lock will permanently disable the operations of PUF enrollment and UID zeroization.

Finally, the Shuffle Read/Write enable option turns on the shuffling operations for read and write individually. These final two options are special because they can be permanently set, as with the above described Program Protect/Testmode Lock/PUF Lock options, or they can be temporarily enabled by using non-OTP registers. It depends on whether the user wants to test out the shuffle function, or if the product is ready to go into production and requires the shuffle option to be permanently enabled for all units.

4.4.4. Access Permissions

PUFrt has several methods to limit access to locations within the OTP/PUF/UID. Using a combination of these methods, PUFrt ensures that privileged data can only be accessed by authorized users, or entirely cut off all access. The various types of access settings available for PUFrt are shown in *Table 4-4*.

The secure range allows locations of the OTP and PUF to be designated as either “secure” or “non-secure”. That is, non-secure processes may only access “non-secure” locations, while still allowing secure processes access to both “secure” and “non-secure” areas.

Table 4-4 PUFrt access permission settings

Name	Affected Location	Access Setting (Read/Write/No Access)	Permanent?
Secure Range	OTP/PUF	NA (for non-secure processes)	Y
Lock OTP	OTP	RW/RO/NA	Y
Post Masking	OTP/PUF	NA	N
Zeroization	OTP/PUF	NA (destroy original data)	Y

Lock OTP allows each location of the OTP to be set with its own access permission, such as read/write (RW) access, read-only (RO) access, or no access (NA). These permissions are permanent and similar to the set flag operations described in section 4.4.3. All locations are set as RW by default and can be

changed to either RO or NA. However, once a location has been set to RO, it cannot be reset back to RW, and once a location is set to NA, it remains forever inaccessible, unable to be read from nor written to.

To accommodate cases where the access permission may only need to be set temporarily, the post masking operation allows locations to be set to NA, as long as the power or PUFrt is reset. After post masking, a location's access permission is set to NA, but after the power is cycled or after PUFrt is reset, all locations that used post masking will revert to their original settings, as determined by each location's lock OTP status.

Finally, as the ultimate safeguard against unauthorized access, the zeroization operation can be used to permanently destroy the data stored in the OTP or UID. Any data that is zeroized will be overwritten as all zeros, unable to be recovered using any physical or logical means, even with full control over PUFrt.

4.4.5. Hard Macro Specific

The last category of operations encompasses all hard macro specific commands. They include putting the hard macro module into a deep sleep for power savings and test modes to detect marginally good or initially failing bits. There is also an auto repair operation for finding and fixing initial bad bits by using redundant cells for repair. The operations that fall under this hard macro category are listed in *Table 4-5*.

Table 4-5 Hard macro (OTP) specific operations

Operation	Description
Deep Sleep	Put hard macro into power-saving mode
Off Cell Margin Read	Off cell margin read (single location)
Initial Off Checking	Off cell margin read (entire array)
On Cell Margin Read	On cell margin read
Auto Repair	Automatic scan and repair of initially bad OTP locations

5

PUFcc is a comprehensive solution tailored for all security applications. Built on a hardware Root of Trust, it integrates hardware accelerators for five core cryptographic functions and offers a robust physical attacks resistant environment. Coupled with highly compatible interfaces, PUFcc can be easy to adopt. PUFcc ensures stringent hardware-based security isolation, confining key generation, operation, and storage within its secure boundary.

5. PUFcc

PUFcc is the secure crypto co-processor from PUFsecurity [25]. It adds a collection of standard cryptographic engines to the fundamental PUFrt. PUFcc creates a general-purpose integrated security IP module capable of supporting the most common cryptographic algorithms, allowing users to build a variety of security protocols.

5.1. Design Philosophy

PUFcc is designed to be the trusted secure co-processor for a larger system, such as when integrated into a system on chip (SoC). With that in mind, PUFcc is created with these four guiding principles.

1. Contain PUF-based root of trust (locally implemented)
2. Be a trusted subsystem that can be isolated from the main system
3. Provide cryptographic services/accelerators for data protection
4. Pair securely to the main system

5.1.1. PUF-based Root of Trust

A system’s security is based on a root of trust, which is why a trusted subsystem such as PUFcc must implement its RoT, instead of relying on an external one for verification and authentication.

One of the important functions of a root of trust is to create a unique ID or hardware unique key (HUK) for a system. This root key may be used as is or used as a basis to derive the other important keys for a system, such as data encryption keys (DEK), key encryption keys (KEK), attestation keys, or keys for digital signatures.

The HUK may be created either externally through a secure key provisioning process flow, or internally using a device known as a physically unclonable function (PUF). Either method will create a suitable root key. The major benefit of using an internal method, such as with a PUF, is that the process is entirely self-contained. That is, there is no need to buy additional hardware such as a hardware secure module (HSM) to create the root keys. Furthermore, this approach eliminates the expenses needed for maintaining a secure clean room environment for the injection of said HSM-created root keys. For these reasons, PUFcc uses a PUF-based root of trust to remove the need for external key provisioning.

Notably, just as not all roots of trust are built the same, there are many technologies that a PUF can be based on. Since a PUF-based RoT relies on its PUF to create a secure HUK which can be considered a system's silicon fingerprint, it is crucial that a PUF demonstrating near-ideal properties is utilized. For this important reason, PUFsecurity has chosen a PUF based on the mechanism of Quantum Tunneling (QT) to build the root of trust that is at the foundation of PUFcc.

5.1.2. Isolated and Trusted Subsystem

PUFcc is designed to work as a trusted subsystem that can be isolated from the larger system that it is integrated into. Sensitive and secure operations can be safely performed by PUFcc because they are operationally separated from the normal functions that the main system is responsible for. As a security co-processor, the operations that are executed within the boundaries of PUFcc need to be implicitly trusted by the system. These are the operations that ensure the overall security of the entire system, especially when the components outside the security boundary of PUFcc cannot be guaranteed to be 100% secure. The system memory, I/Os, and other devices sharing the same system bus are prime targets for hackers to gain access or control of the system.

Relying on the principle of hardware isolation, PUFcc integrates the secure storage, a TRNG, cryptographic hardware acceleration, and the previously mentioned PUFrt root-of-trust inside its security boundary which is bolstered by anti-tampering designs. This is the most restrictive form of separation (hardware isolation) between PUFcc and the main system, guaranteeing the safety of security operations executed inside PUFcc. This is because all required components for the secure functioning of PUFcc are already inside a trusted boundary of protection.

5.1.3. Cryptographic Services and Accelerators

PUFcc includes a full suite of cryptographic services and hardware accelerators for the protection of data at rest and while in transit.

The cryptographic services and hardware accelerators have undergone NIST's Cryptographic Algorithm Validation Program (CAVP) certification process [26]. Data is protected through the properties of confidentiality and integrity. Confidentiality is provided in the forms of both symmetric and asymmetric ciphers. If a bad actor gains the ability to snoop upon the system bus, the encrypted bus traffic would be of no use to them. Integrity is provided in the forms of both a hash engine and message authentication codes. If a hacker manages to alter any important system data, such as a piece of the boot code, integrity checking will alert the system to such tampering.

In addition, to reemphasize the previous section's point on isolation, it is important to note that these crypto accelerators are built into PUFcc. There is no need for sensitive data to leave the secure boundaries of PUFcc while it is in plaintext. Finally, PUFcc's security boundary has been created from the same anti-tamper and anti-hacking protections that are employed in PUFrt, whose strength has been independently verified by third-party laboratories.

5.1.4. Secure Pairing

PUFcc is designed to be integrated into a larger system, acting as a trusted partition, such as when it is a part of a system on chip (SoC). To meet that goal, PUFcc comes equipped with a standard APB slave interface to connect to a secure channel that is a dedicated private bus between the SoC and PUFcc. In case a dedicated secure channel has not been made available by the

SoC, the APB protocol itself can support the separation of secure/non-secure transactions on a non-private APB bus to protect sensitive communications between the main system and its secure subsystem, PUFcc.

5.2. PUFcc Architecture

PUFcc is a cryptographic co-processor for offloading the security tasks that would otherwise be left for the main processor to handle. It is designed to be easily integrated into the system while also having the ability to complete all security-related tasks independently. PUFcc is based on the PUFrt root of trust which allows it to be implicitly trusted by the system. This allows PUFcc to anchor for all required chains of trust, such as secure boot and update.

PUFrt also provides static and dynamic entropy sources for the creation and derivation of system keys, as well as a block of secure OTP memory for users to store their private data. *Figure 5-1* illustrates the architecture of PUFcc and demonstrates how it integrates into a larger system using the APB/AHB/AXI buses. Collectively a part of an industry bus standard AMBA (Advanced Microcontroller Bus Architecture), the APB (Advanced Peripheral Bus), AHB (Advanced High-performance Bus), and AXI (Advanced eXtensible Interface) bus interfaces act as a backbone for carrying communications between the different components of a system along a shared bus [24].

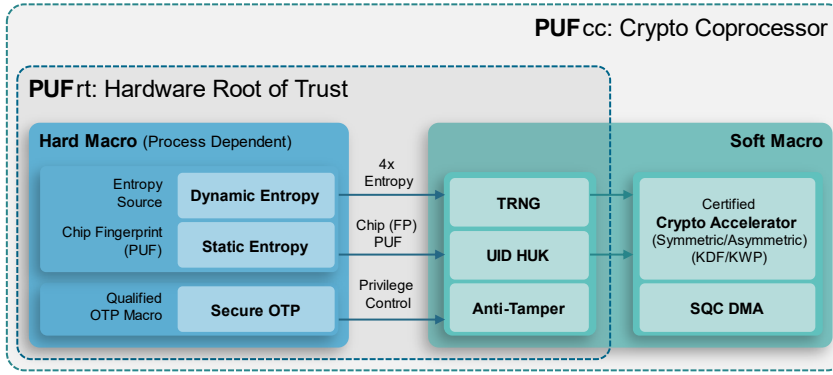


Figure 5-1 PUFcc architecture.

A fixed sequencer (SQC) logic control module gives PUFcc the ability to run its built-in cryptographic algorithms with minimal to no help required from the main processor, freeing up valuable cycles to be allocated for other CPU jobs.

PUFcc supports a complete set of NIST CAVP-certified and China's Office of State Commercial Cryptography Administration (OSCCA) compliant [27] cryptographic algorithms implemented in hardware which have been certified by third parties. Customization of these PUFcc crypto algorithms remains flexible due to their modular design. This means that any user requests, such as choosing either SM4 or AES for a cipher, can be accommodated. PUFcc can therefore meet the current and future security requirements of flexible platforms based on the RISC-V architecture. Besides the included crypto engines, numerous digital and analog anti-tampering designs further strengthen PUFcc into a reliable crypto coprocessor.

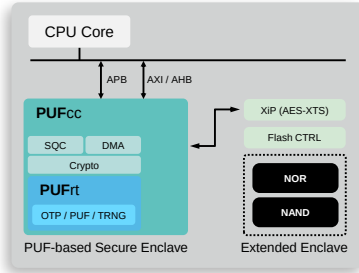
To lower the barriers of SoC integration, PUFcc supports a standard APB control interface used for register access control and a DMA with a standard AXI4 control interface to access larger amounts of data stored in system memory quickly. The accompanying Software Development Kit (SDK), including Linux bare-metal firmware and high-level APIs, helps further accelerate the software development and deployment of this IP.

Along with NIST-standard Key Wrapping (KWP) and Key Derivation Function (KDF) to protect key usage and export, PUFcc can generate multiple keys on-demand for the RISC-V physical memory protection (PMP) scheme for the separate protection of each application. Furthermore, the unique HUK of each PUF serves well for Secure Boot and Secure OTA updating, as the same software on different IoT devices will have its own secret key. PUFcc can thus establish a firm foundation of security as the future potential of the Internet of Things is realized, which will annually add billions of connected devices, all of which require protection.

5.3. PUFcc Block Descriptions

A detailed block diagram of the PUFcc crypto coprocessor is shown in *Figure 5-2*. The heart of PUFcc is its root of trust, PUFrt. Combining both hard and soft macros, PUFrt provides both internal UID/HUK generation and secure code/key storage, along with a NIST-compliant TRNG. Key management is supported by PUFcc with included key array (KA) and key wrapping (KWP) modules. A suite of cryptographic engines, including ciphers, hashes, message authentication codes, and key derivation

support the latest secure protocols. Access to PUFcc is performed through dual APB and AHB/AXI (user selectable) bus controllers which interfaces already familiar to most system integrators. Finally, a direct memory access (DMA) unit facilitates the connection of external memories to the same shared AHB/AXI bus with PUFcc. The components and their sub-blocks in *Figure 5-2* will be covered in the following sections.



PUFcc	Features / Benefits
PUFrt	- Inborn HUK/UID, High-Q TRNG and Secure OTP. - Support zero-touch secure supply chain.
Crypto Algorithms	- AES, SHA, ECC **NIST-CAVP** passed Cryptos. - Support secure boot and secure operation.
DMA	- Unload CPU loading for digest. - Support AXI or AHB master.
SQC (State Machine)	- No CPU/MCU and boot rom required. - Suitable for closed system and lightweight IoT.
Extendable Enclave	- Expandable secure boundary on ext. Flash. - Realize zero-trust secure storage using PUF.

Figure 5-2 PUFcc block diagram.

5.3.1. PUFrt Root of Trust

The sub-blocks (PRTC/PUFtrng/PUFuid/PUFkeyst/Hard Macro) have already been covered in Chapter 4 and its subsections. Please refer to those sections for details regarding the PUFrt root of trust that PUFcc derives its security from.

5.3.2. Crypto Algorithms

The standard package of built-in algorithms supported by PUFcc includes asymmetric and symmetric ciphers, hashes, Key

Derivation Functions (KDF), and Message Authentication Codes (MAC). As each customer's needs are different, the configuration of these crypto engines can be customized accordingly. In addition, the relevant certifications for these crypto engines have already been obtained, such as from the NIST (US) and OSCCA (China).

5.3.3. Standard Bus Interfaces including DMA Module

PUFcc is equipped with two AMBA-compliant bus controllers for convenient system integration. The APB client module is for command requests or responses and their associated configuration data. The AXI master module along with DMA, a direct memory access control, is used to move larger blocks of data directly from external memory storage into and out of the crypto engines at a high throughput. The following content will introduce more details about these interfaces.

5.3.3.1. APB Client

The APB client module allows system commands to interface natively with PUFcc. There is no need for system integrators to build additional translation modules to use PUFcc in their systems that already support the APB interface. In addition to handling system requests, this APB client also includes flexible access control logic with both secure/non-secure world support to set full access/read only/no access permissions for the hard macro OTP/PUF locations of PUFrt, at 32-bit word granularity.

5.3.3.2. AXI or AHB Master

The AXI/AHB master module, in conjunction with the direct memory access (DMA) controller, allows an external memory on the same AXI (Advanced eXtensible Interface) or AHB (Advanced High-performance Bus) system bus to interface directly with PUFcc. For larger blocks of data, such as messages for the hash or MAC engines, fixed information for KDF, or the plaintext/ciphertext for the AES engines, it is much faster to send these through the DMA and AXI/AHB interface, rather than over the slower APB bus.

5.3.4. Sequencer (SQC)

PUFcc relies on a fixed state machine for logic control and sequencing. This allows for a lightweight implementation, as compared to using a programmable microcontroller for a controller. Thus, for users that only require the included crypto engines but not writing their own crypto algorithms, PUFcc would be a good fit for their applications. As it only run the hardwired states of its programming, this sequencer is resistant to the insertion of Trojan Horses and other malware. It is also good for installations that are “set and forget”, such as for remote edge IoT scenarios.

5.3.5. Extendable Enclave with add-on Module Support

PUFcc is self-contained in its protective shell of cryptography and anti-tampering features. However, with the addition of add-on

modules (see section 5.6 for more details), this protection may be extended to additional system-attached memories, such as an external NAND Flash.

5.4. PUFcc Operations

The operations supported by PUFcc can be broken down into five categories:

1. PUFrt
2. Key management
3. Integrity/authentication
4. Private key (symmetric) cryptography
5. Public key (asymmetric) cryptography

The supported functions of PUFcc control the root of trust PUFrt (category #1) and the set of standard cryptographic engines (categories #2~5). This allows PUFcc to support the most common cryptographic primitives such as random number generation, confidentiality, integrity, authentication, and non-repudiation.

5.4.1. PUFrt

The first set of functions supported by PUFcc are those related to the root of trust at the heart of PUFcc, called PUFrt. The operations of PUFrt can be broken into five categories. The *Table 5-1* lists these five categories along with some example operations for each category. For more details, please refer to section 4.4 for full descriptions of each category and operations within.

Table 5-1 PUFrt example operations

Category	Example Operations
Data access	Read/write OTP, get random numbers
PUF setup	PUF quality/health check, enrollment
Settings/options	Test mode lock, shuffle mode on/off
Access permissions	Secure/non-secure function access, zeroization of OTP/PUF
Hard macro	Sleep mode on/off, repair, margin read

5.4.2. Key Management

The key management functions supported by PUFcc include storage, wrapping, and derivation, which are implemented by the KA/KWP/KDF modules. As summarized in *Table 5-2,* the KA module is specifically for key storage. The KWP module is for wrapping or encrypting keys for secure key export to modules outside of PUFcc over potentially non-secure buses. The KDF module is for deriving new keys using keying material from either internal or external sources.

Table 5-2 PUFcc key management operations

Name	Algorithm	Related Standard	Types
Key Storage	n/a	n/a	OTP/ session/ private/ shared secret
Key Wrapping	AES-CBC CS2	SP800-38F	Import Export
Key Derivation	SHA2/HMAC	SP800-108	Expand only Extract-then-expand

Keys may be imported from an external source and stored either in the OTP within PUFrt or the KA within PUFcc. The KA has specialized slots for storing three types of keys: private, session, or shared secret. When the user issues the command to import a key into the KA, besides specifying which slot the key will be stored in, the option to wrap the key beforehand is also available.

The key wrapping module (KWP) allows users to import a key and store it as ciphertext within the KA, or to export a key from the KA as ciphertext [28]. The key encryption key (KEK) for key wrapping comes from the KA. After the key has been wrapped and is now ciphertext, it will be available for access by request from the APB interface.

Starting from a source key, called a key derivation key (KDK), the KDF module can derive a new and equally cryptographically strong key [29]. The two-step derivation process, also known as extract-then-expand, is used on KDKs from either an external source or the shared secret slot in the KA. The one-step process which expands only, has greater flexibility in its choice of KDKs. In addition to using source keys from external storage or the KA, a KDK may also come from the PUFrt, either stored on the OTP/PUF or generated from the TRNG.

As the function of privacy depends heavily on the uniqueness and randomness of the data encryption keys, it is important that there are enough keys available to protect large amounts of data. To

ensure an ample supply of quality keys, PUFcc employs a secure internal key provisioning process and has a built-in key derivation function (KDF) module to generate more keys on demand. PUFcc supports both one-step and two-step key derivation, as illustrated in *Figure 5-3*.

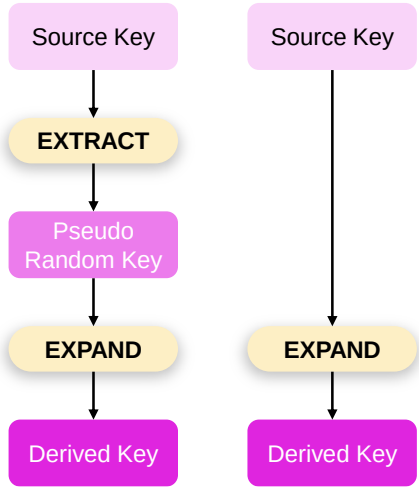


Figure 5-3 Key derivation function, one- and two-step operation.

5.4.3. Integrity and Authentication

PUFcc supports integrity and authentication checking through hashing and the creation of message authentication codes. *Table 5-3* summarizes two types of hash digests and two types of MACs that can be generated by the crypto engines of PUFcc, along with the related standards that may be referenced for further details.

Table 5-3 PUFcc integrity/authentication operations

Name	Algorithm	Related Standard	Output Type
Hash	SHA2	PUB180-4	Hash digest
Hash	SM3	GM/T 004-2012	Hash digest
CMAC	AES	SP800-38B	Message authentication code
HMAC	SHA2	PUB198-1	Message authentication code

PUFcc supports both SHA-2 and SM3 [27], [30] hash engines for the creation of digests that are used for integrity checking. By comparing the digest of the original dataset with the digest of the "same" dataset, it is quickly determined whether the second dataset is indeed a copy of the unaltered original, without the need to compare the datasets themselves. A comparison of the NIST SHA-2 and OSCCA SM3 approved hash engines are listed in *Table 5-4* [30].

Since the creation of a MAC relies on a secret key from the originator of the data, both integrity and authentication checking of a dataset is supported when the secret key and message digest are compared to the corresponding true known key and digest values. PUFcc supports both hash-based (following FIPS PUB 198-1) [25],[32] and cipher-based (following NIST SP800-38B) methods of MAC generation [33].

Table 5-4 Comparison of SHA-2 (256-bit) versus SM3

	SHA-2 (SHA-256)	**SM3**
Structure	Merkle-Damgard	Merkle-Damgard
Compression Function	Davies-Meyer	Davies-Meyer
Input (bits)	$0 \leqq len \leqq 2^{64}$	$0 \leqq len \leqq 2^{64}$
Output (bits)	256	256
Rounds	64	64
Operations	ADD, XOR, NOT, ADD (mod 2^{32}), SHR, CON (catenate), ROTR	ADD, XOR, NOT, OR, ADD (mod 2^{32}), CON (catenate), ROTL
Constants (words)	64	2

5.4.4. Private Key (Symmetric) Cryptography

PUFcc supports AES, SM4, and ChaCha cipher engines for symmetric key cryptography. It is named “symmetric” because the same private key is used for both encryption and decryption [34]. All parties who want to protect and access the encrypted data need to share the same private key. ChaCha is a stream cipher that generates a pseudorandom key stream to XOR with the plaintext, creating the ciphertext [35]. AES and SM4 are block ciphers that encrypt and decrypt the plaintext and ciphertext on a block-by-block basis. In addition, some AES modes can generate an authentication tag from the plaintext input, so the data and its integrity can be protected simultaneously [36], [37]. It is also known as authenticated encryption.

Table 5-5 summarizes the symmetric cipher operations, listing the engine name, modes of operation, related standard, and whether the cipher also generates an authentication tag during the encryption process.

Table 5-5 PUFcc private key cryptography operations

Engine	Mode(s)	Related Standard	Authenticated Encryption?
AES	ECB/CFB/OFB/CBC/CTR	SP800-38A	No
AES	CCM	SP800-38C	Yes
AES	GCM	SP800-38D	Yes
AES	XTS	SP800-38E	No
ChaCha	ChaCha20	RFC8439	No
ChaCha	ChaCha20-Poly1305	RFC8439	Yes
SM4	Standard	GM/T 002-2012	No

The AES engine supports three key sizes of 128/192/256 bits while SM4 only supports keys of 128 bits in length [31]. The ChaCha cipher engine supports key sizes of 128 and 256 bits [35]. The private keys for these symmetric cipher operations may come from a variety of sources, including external input, OTP, PUF, TRNG, or the KA. The architectural differences between a stream cipher such as ChaCha and a block cipher such as AES are illustrated in *Figure* 5-4, showing how a block cipher will divide up the original plaintext into discrete blocks of data, upon which the cipher will operate, creating blocks of ciphertext as output.

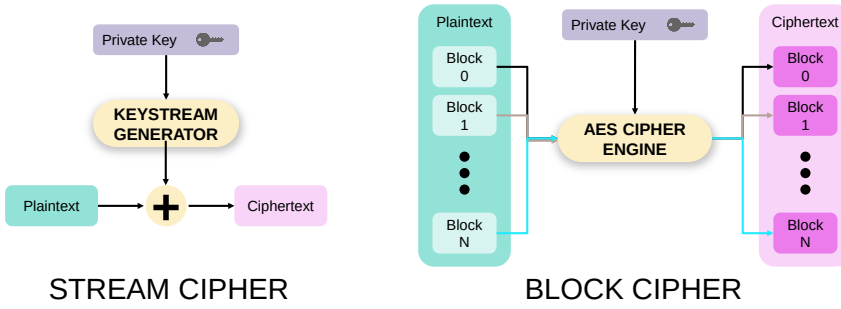


Figure 5-4 Block (AES) versus stream (ChaCha) cipher comparison.

5.4.5. Public Key (Asymmetric) Cryptography

In addition to private key cryptography, PUFcc also supports public key cryptography with its public key co-processor, otherwise known as the PKC module. The PKC supports three different asymmetric crypto engines, ECC, RSA, and SM2. The asymmetric crypto functions supported by PUFcc fall under the following categories:

1. Message Encryption/Decryption
2. Digital Signature Algorithm (DSA)
3. Key Agreement/Exchange
4. Public and Private Key Generation
5. Public Key Validation

As opposed to a symmetric cipher such as AES, an asymmetric cipher uses a key pair made up of a private and public key. The key pair allows for the public key to encrypt plaintext into ciphertext, while the private key decrypts the ciphertext back to plaintext. It

performs encrypted communications without the need to share one's private key with the sender.

Table 5-6 collects the asymmetric cipher operations, listing the algorithm name, the algebraic basis of the algorithm, the related standard[38], and the functions that use said algorithm, such as digital signing, signature verification, and encryption/decryption. The DSA functions of generating and verifying a digital signature are supported by PUFcc, such as the Elliptic Curve Digital Signature Algorithm (ECDSA) [32]. That is, a private key can be used to digitally sign a document, while the paired public key can be used to verify a digitally signed document.

Table 5-6 PUFcc public key cryptography operations

Algorithm	Algebra	Related Standard	Function
ECC	Elliptic Curve	FIPS 186-4	Digital signature sign/verify
ECC	Elliptic Curve	SP800-56A	Key agreement/exchange
ECC	Elliptic Curve	FIPS 186-4	Key generation
ECC	Elliptic Curve	SP800-56A	Public key validate
RSA	Galois Field	FIPS 186-4	Digital signature sign/verify
SM2	Elliptic Curve	GM/T 003-2012	Message encrypt/decrypt
SM2	Elliptic Curve	GM/T 003-2012	Digital signature sign/verify
SM2	Elliptic Curve	GM/T 003-2012	Key agreement/exchange

Since a private key is meant to be in the sole possession of one authorized party, digitally signed documents support non-repudiation. For example, user A creates a private/public key pair and shares the public key with user B. After user A signs a message with his private key and sends it to user B, user B can verify the authenticity of the signature by using the public key. It is impossible that any other party, not even user B, could have signed a document that can be verified by using the same public key from user A. That is, if a document's digital signature has been verified using user A's public key, it would be impossible for user A to claim later that he did not sign the document.

The next set of functions is related to establishing a shared key or secret, using an algorithm such as the Diffie-Hellman key exchange method. One of the very first public-key protocols ever created, Diffie-Hellman enables two parties to create a private and shared key only known to them, even if they have no prior knowledge of each other's secrets. Following the specification in the NIST SP800-56A document [39], a shared secret may be established between two parties by using two ephemeral private keys, or through two ephemeral keys and two static keys. Once generated, the shared secret is stored in the shared secret slot of the KA.

Private and public key generation for asymmetric ciphers is the fourth set of functions supported by the PKC module in PUFcc. Private keys are created from an entropy source, either from a dynamic source such as the TRNG, or from a static source such as the PUF. Private keys generated from the TRNG are called

ephemeral private keys, while those generated from the PUF are called static private keys. Next, after a private key has been created, its paired public key can be found by using the public key generation function. Finally, as public keys are meant for other parties to use, once the public key has been generated, it is available on the APB interface for authorized read requests. The public key is not stored in the KA of the PUFcc that generates it.

The last function supported by the PKC is public key validation. Public key validation determines if any public key transferred to PUFcc possesses the correct arithmetic properties for a proper public key [25]. For example, say we have two parties A and B who are interested in establishing a shared key between the two of them. They decide to use the Diffie-Hellman key exchange method to create a shared secret. However, party A decides to be sneaky and sends party B a non-standard public key A for the creation of shared secret S. Because this shared secret S was created with a suspect public key A, their shared secret S may not be as cryptographically strong. Therefore, it is recommended to check the properties of another party's public key with the public key validation function before using a public key from a party who has not yet been fully vetted.

5.4.6. Built-In Autoload Function to Support Secure Boot

The security and trust of the underlying system hardware form the bedrock for the security of the entire system operation. In other words, a hardware root of trust underpins every secure system. A

secure boot flow initializes and prepares hardware for trusted and correct system operation. Thus, every time it wakes up, a system depends on the secure boot flow to guarantee proper system operations. An example secure boot flow includes six steps:

1. Load Power/Clock parameter settings from OTP
2. Load and Verify the Boot Loader from OTP
3. Load and Verify the Boot Code from the external storage
4. Load and Verify the OS Loader
5. Load and Verify the OS Program
6. Load and Verify the APP/AI model

The first step is to load the trimming parameter settings so that the power and clock can operate as the system architect expects. The next step is to use the HRoT to verify boot code integrity, after which it can be loaded. Then the OS integrity and APP/SW integrity are verified, with the corresponding OS and APP/SW loading performed after each verification has passed.

However, as each loading and verification step is a potential target of an attack, we should make sure the boot flow is protected at every step along the way to guarantee system security. This must include the critical first step of trimming the power supplies (in particular the bandgap circuit) and clock. During the initial wake-up period, chip power levels and clock timings are not yet stable. Thus, the chip at this stage will not be able to function correctly, so even the secure boot up process must wait until the power levels and clock timings are trimmed to their proper operation settings.

If the secure boot process starts before the clock and power have been trimmed, it would improperly execute, leaving the chip and device in a state of unguaranteed security. Thus, it becomes critical to start the secure boot process before the power levels and to set the clock timing accurately. For a normal practice, the addition of extra circuits is applied. However, a new function called “autoload” is available without using any additional time or creating cost penalties.

Figure 5-5 illustrates the system integration of PUFcc, including connections to the power and clock modules, as well as the different stages of a secure boot flow, as listed on the right diagram. Note how this list includes the critical first step of loading the proper power and clock trim parameters from the OTP. Another view of the secure boot flow can also be seen in *Figure 5-2* after the power and clock values have been properly trimmed.

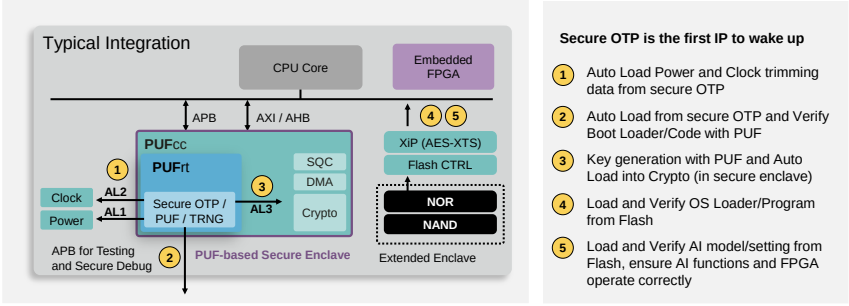


Figure 5-5 Secure boot with PUFcc.

During power-on, the secure OTP will be the first block to power on, so it can be used to store the power/clock trimming parameters.

The OTP's instant-ready-on characteristic and autoload function ensure that the power and clock can be quickly and correctly trimmed so that the system can finish the rest of the secure chip booting process.

The autoload function, triggered after the reset pin is released, loads the section of the OTP that has been pre-written with the power and clock trim values. Thus, it saves the system from needing to manually send read requests to the OTP upon power up. In this way, the system can start with a secure boot immediately after power-on, without the need for any additional circuitry to support the loading of the trimming values stored in the OTP.

After the power and clock have been trimmed properly, secure boot starts with verifying that the boot loader is authentic and has maintained its integrity. Next, the PUF-based root of trust can properly verify the boot code's integrity and validity. Then, the boot code may be loaded after it has been deemed safe to do so, through either the XIP or GCM/XTS Accessory Module (see the section 5.6 regarding Add-On Modules for more details). The OS integrity and APP/SW integrity can thus be verified in a secure environment until the system has finished booting up and is ready to accept user requests.

5.5. Add-On Modules

The available add-on modules extend the security boundary of PUFcc and PUFrt to protect any connected, external Flash

memories. These accessory modules can support three different types of Flash: NOR (XIP), embedded (eFlash), and NAND (GCM/XTS). By expanding the enclave of protection to encompass an external Flash using one of these modules, system integrators may now augment a system's capabilities with a secure and flexible non-volatile memory paired with either the essential PUFrt or PUFcc.

5.5.1. XIP Accessory

This XIP accessory module is created to extend the enclave of protection to include a NOR Flash attached to PUFcc. Specifically, the name "XIP" (eXecute In Place) refers to the feature that allows protected code to be directly executed from the attached NOR Flash. Unlike DRAM, it does not need to first move the code to another memory before execution. *Figure 5-6* shows the combination of PUFcc and the XIP Accessory module. This configuration is used for the protection of a system attached to NOR Flash while simultaneously supporting execution in place operations.

This add-on module to PUFcc merges the data-at-rest protection offered by the XTS mode of the AES block cipher with PUFcc. This combination of NVM and crypto co-processor allows the anti-tampering designs, secure/non-secure policy, and location access control of PUFcc to automatically carry over to the attached NOR Flash. In addition, the unified AHB or AXI interface of PUFcc makes system integration efficient – the NOR Flash becomes an extension of the memory-mapped register space of PUFcc, rather

than acting as a separate memory unit requiring its flash controller. Finally, by including PUFcc with the standard package of crypto engines, namely the CMAC/HMAC engines, authentication of the executable code stored in the NOR Flash is also supported.

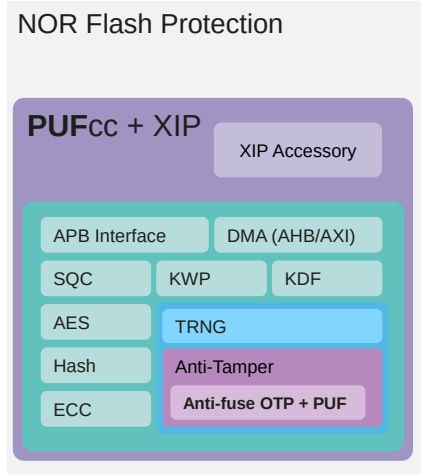


Figure 5-6 PUFcc and XIP accessory module.

The XTS mode (XOR-encrypt-xor based Tweaked codebook mode with ciphertext Stealing) is created to ensure the confidentiality of data on storage devices, particularly for sector-addressable storage [34]. As described in the NIST document SP800-38E and in the IEEE standard 1619-2007, the XTS mode is the natural choice to protect such data stored on the attached NOR Flash. Furthermore, by using an internally generated DEK from PUFcc, either one directly from the PUF or one derived from the PUF using the built-in KDF, this key automatically gains the protections afforded by never needing to leave PUFcc's circle of protection. Finally, by using a higher-speed AMBA-compliant bus,

such as AHB or AXI instead of APB, the XIP accessory module can guarantee the bandwidth necessary to maintain execute-in-place capabilities.

By unifying both co-processor and NOR Flash control under the same AHB/AXI system bus, the XIP module allows integrators and architects to work with a familiar interface. It eliminates the costs needed to integrate a separate NOR Flash and its corresponding controller into a system. Finally, with real-time encryption and decryption based on the unique randomness built into each on-board PUF, PUFcc with the XIP add-on module supports execution in place while offering secure data-at-rest protection for sensitive code and data stored on the attached NOR Flash.

5.5.2. eFlash Accessory

The eFlash accessory module allows systems to easily integrate an embedded Flash with PUFrt or PUFcc, which in turn securely integrates with the larger system. This offers the dual benefit of adding an essential Root of Trustor secure crypto co-processor to a system, along with the peace of mind brought by upgrading the security of the embedded Flash. *Figure 5-7* shows the combination of PUFrt, eFlash, its controller, and the eFlash accessory module. This configuration is used for protecting a system embedded eFlash.

This add-on module allows for the PUFrt or PUFcc and embedded Flash control signals to be taken care of by a single ARM peripheral bus interface, such as APB. As such, system integrators

can work with the same standard interface to execute both security and embedded Flash operations. In addition, the security boundary, including access control, secure or non-secure privileges, and anti-tampering designs, that have been built around PUFrt or PUFcc are automatically extended to cover the embedded NVM. Furthermore, to support high data throughput to or from the Flash, including burst read/write operations, the eFlash accessory module includes a second bus control module, allowing the embedded Flash to be accessed by a system's high-speed AXI or AHB data bus. This combination of eFlash and PUFrt or PUFcc solution eliminates many problems of integration, allowing system or product prototyping to commence as early as possible.

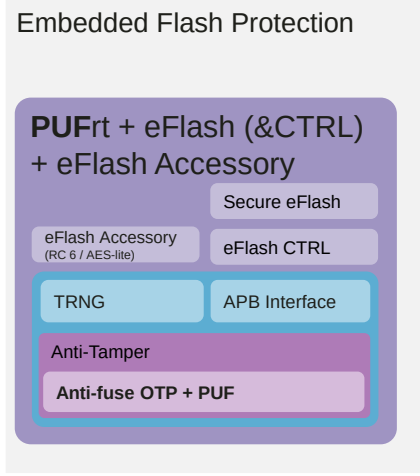


Figure 5-7 PUFrt with eFlash, controller and accessory module.

The eFlash module protects the embedded flash through data encryption and address scrambling. Flash addresses are scrambled by using an address protection key which is generated

through a key derivation function from the inborn entropy of the built-in PUF. The eFlash data itself is also encrypted by using a lightweight cipher such as RC6 or AES-lite that is a user-selectable configuration. A lightweight cipher is specifically chosen for performance and cost reasons, and to separate the embedded Flash data protection scheme from the AES data encryption scheme used by PUFrt or PUFcc. The option of implementing an error correction code through this accessory module is also available for users requiring the stability of the stored data.

By using the eFlash accessory module, users may combine an embedded Flash with the PUFrt or PUFcc. By doing so, this extends the security enclave of PUFrt or PUFcc to cover the embedded Flash. In addition, this allows the standard AMBA-compliant bus protocol interfaces (APB and AHB/AXI) of PUFrt or PUFcc to be used for accessing the embedded NVM. This solution enables designers to conveniently add both embedded NVM storage and security features to existing systems with a minimum of fuss. With the extendable enclave concept, secure embedded storage is guaranteed, as the built-in security of PUFrt or PUFcc is included, so that adding extra embedded storage does not come with the risk of opening a security back door for hackers.

5.5.3. GCM/XTS Accessory

The GCM/XTS add-on module is specifically designed to prevent the cloning or hacking of a company’s soft assets. Such assets include trained AI models, proprietary firmware and software, and Field Programmable Gate Array (FPGA) bitstreams of licensed

products. The modern supply chain has been stretched thin, into a collaboration between various vendors and service providers that can span the globe. With such a long supply chain, the opportunities to extract or tamper with a company's valuable soft assets are many.

Proprietary soft assets, such as firmware or boot code are vulnerable, especially after they have been written into storage, such as to an attached NAND Flash. This accessory module is created to protect soft assets throughout their lifetime, from when they are first programmed, through all authorized updates, and finally until the product's end of life. *Figure 5-8* shows the combination of PUFcc and the GCM/XTS accessory module. This configuration can then be used for protecting the soft assets stored on a system NAND Flash.

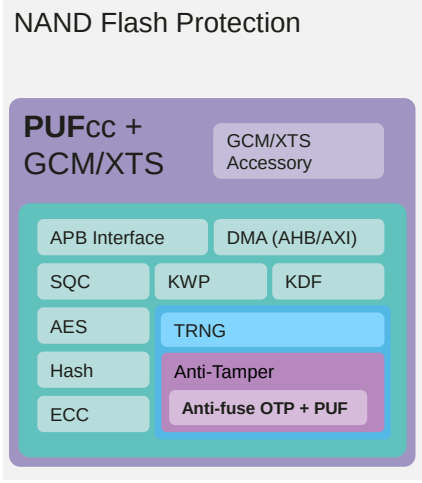


Figure 5-8 PUFcc and GCM/XTS accessory module.

The GCM/XTS accessory module protects soft assets through data encryption and message authentication codes by using the XTS and GCM modes of the AES block cipher. Continuing the above example of soft asset protection, they are initially encrypted with a “global” key and programmed at the factory. Information, such as the asset version number, planned region of usage, and factory number, can be utilized in creating a global key customized to each group of products that share the same defining characteristics.

For example, all products shipped to Europe from factory A use global key A-E, while products shipped to Asia from the same factory use global key A-A. Using such a global key, soft assets may be encrypted at the factory using AES-GCM, an authenticated cipher, so that an authentication tag is created alongside the ciphertext of the soft assets. By doing so, soft assets are protected from the start, in addition to allowing the end user a way to check for unauthorized tampering of the assets by comparing the authentication tags before the official onboarding process for the product’s in-field use.

After it has been determined that no malicious tampering has taken place, users can re-encrypt the assets by using their own unique “local” key which is derived from each user’s unique PUF values. Every soft asset is now protected by a key that is unique to each PUF on each die for each product. This re-encryption can use the AES-XTS mode that is well suited for protecting soft assets stored as sector-addressable data. If the soft assets have not been

updated or modified, no new authentication tag needs to be generated, hence GCM need not be called upon for this local key re-encryption process.

We can write out the above example in a list form.

1. Encrypt soft assets using a global key (GCM) at the factory
2. Distribute product to users
3. Examine the authorization tag to check for tampering (MAC from GCM)
4. Decrypt assets using a global key (GCM)
5. Re-encrypt assets using a local key (XTS)
6. Decrypt assets using the local key when needed (XTS)

The GCM/XTS accessory module adds the protection of dual AES-GCM and AES-XTS ciphers, as well as the ability to check for unauthorized modification through the authenticated encryption of AES-GCM. One such example of usage for this add-on module is the ability to protect soft assets from initial programming, all the way until end-of-life. Protection begins at the factory, using a global key to encrypt data during the first write of soft assets to the product. Before the products are put into use, the assets are written again (second write) with the individual part's local key.

In this way, the global key is no longer needed after the product has been deployed for in-field usage and after second write. The only exception would be during over-the-air (OTA) updates that need to be authenticated by using the factory key before updates

may take place. Then, after re-encryption of an update by using the local key, normal operations may resume using local key protection only.

6

The training data and models of AI are paramount today, influencing information accuracy and confidentiality. As devices, especially IoT devices, become more ubiquitous and intelligent, AI and IoT security will directly impact personal safety. This chapter delves into the vulnerabilities and threat models of AI and IoT, correlating with the five primary cryptographic functions from Chapter 1, to summarize the usage and essential features of hardware Root of Trust.

6. PUF-based Chip Security for AI and IoT

In this rapidly evolving landscape where the future hinges on the seamless integration of interconnected devices performing secure operations through extensive data collection, calculation, and analysis, the imperative for robust security measures becomes increasingly evident. Amidst the proliferation of AI-driven solutions and the pervasive connectivity of everyday devices to the Web, ensuring the integrity and confidentiality of data transcends the realm of network information security; it becomes a cornerstone for safeguarding life safety.

Physical Unclonable Function (PUF) emerges as a necessary component in this paradigm shift towards a more interconnected and AI-driven future. The uniqueness inherent in PUF provides a distinctive and unforgeable identity for each chip, forming the bedrock for secure communication and data exchange. As our lives become more intertwined with the digital realm, the need for safeguards against unauthorized access, data breaches, and device tampering becomes paramount.

Incorporating PUF into the architecture of AI and IoT chips becomes not just a security measure but a fundamental requirement for building trust in these interconnected ecosystems. PUF's ability to resist cloning, simulate physical characteristics,

and generate secure cryptographic keys ensures that the devices orchestrating our daily lives are not only intelligent but also inherently secure. It addresses the burgeoning challenges of protecting sensitive information and fortifying the foundations of AI and IoT technologies against potential threats. As we navigate towards a future where our reliance on AI and interconnected devices deepens, the incorporation of PUF becomes a linchpin in the pursuit of a secure and resilient digital ecosystem.

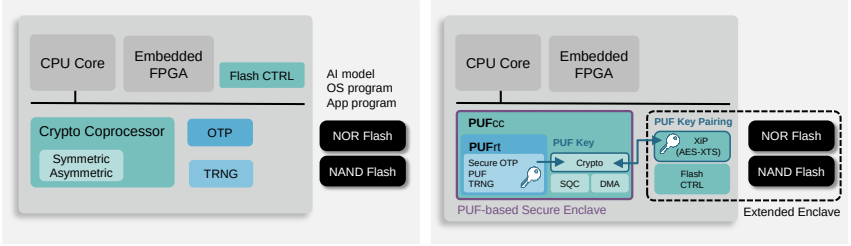


Figure 6-1 Protection by device pairing with PUFcc.

In many applications, sensitive data is typically stored in an external flash memory. *Figure 6-1* illustrates that when an application initiates, this data must be loaded onto the chip for execution. If the security of the flash memory is compromised, proprietary knowledge becomes susceptible to tampering or unauthorized extraction. Therefore, establishing a secure environment for any connected external flash is imperative, especially in mission-critical applications. A practical solution involves implementing PUFcc which we have introduced in chapter 5 for chip security. With PUFcc support, a PUF-derived

encryption key can encrypt the system assets stored in the flash memory, safeguarding them against theft. The uniqueness of each chip's PUF ensures that the data encryption key generated from it will be distinct for every PUF, achieving not only data protection but also a unique binding of each PUF to the specific flash it safeguards. This utilization of a distinctly defined static entropy source allows for the development of secure applications using PUFs and PUF-based solutions tailored for the realms of AI and IoT.

6.1. PUF-based Security for Artificial Intelligence

Given the expansive nature of AI, our focus in this book will be directed specifically towards the subfield of machine learning and its associated topic of neural networks. Modeled after the human brain, artificial neural networks undergo a training phase, also referred to as the processing phase, before being deployed for practical use. It is crucial for both the training and processing phases of AI networks to incorporate adequate safety and security measures, particularly as the impact of AI systems extends deeper into various aspects of our lives.

Recognizing the diverse applications of AI, varying levels of security are requisite for different use cases. For instance, a trained voice recognition system tasked with handling customer calls for a cable TV company may not necessitate the same rigorous security measures as an AI system deployed in a hospital

for diagnosing patients. However, regardless of the application, additional security layers are indispensable to meet the relevant safety and security standards. The potential consequences of a caller being directed to the wrong extension differ significantly from compromising the safety of patients.

This section on AI is structured into three subsections. Firstly, it provides descriptions of the threat models confronting AI systems. Secondly, there is a concise introduction to PUF-based solutions tailored for AI. Finally, a detailed examination of the targets within an AI system and their vulnerabilities is presented. Following these AI-specific subsections, the discussion shifts to IoT systems. Only after establishing a foundational understanding of the challenges faced by both AI and IoT systems do we delve into cryptographic functions employed to mitigate threats, recognizing that many security solutions and functions are shared between these two interconnected domains.

6.1.1. AI Threat Models

There exist four primary threat models for an AI system, namely poisoning, backdoor, evasion, and stealing [39]. Each of these threats is directed at different components of an AI system, encompassing the processor, training data, trained model, input data, or inference results, as outlined in *Table 6-1*. While the processor itself is not explicitly listed as a target, attacks on AI processors fall under the broader category of security threats to CPUs/processors.

Table 6-1 AI threat models and their targets

Threat Model	Target	Description of Vulnerability
Poisoning	Training Data	Training data is corrupted so AI model will produce incorrect results
Backdoor	AI Model	Backdoors placed inside AI model allow bad agents to influence the model's outputs
Evasion	Input Data	Alter input data so inference results are incorrect
Stealing	Training Data Input Data AI Model Inference Results	Targets can be stolen by bad agents and later reappropriated for unauthorized use

Poisoning targets the training data set used to instruct an AI system in making accurate inferences. If malicious actors manipulate the training data, they can influence the AI system's behavior post-deployment. For instance, in training a neural network to recognize stop signs, a poisoning attack may alter tags in the training data, causing the network to misidentify a stop sign as a speed limit sign during actual use.

Backdoor attacks aim to impact trained AI models, allowing a malicious agent to implant behaviors triggered later. Traditionally, visible triggers are placed in training data, teaching the neural network to make incorrect inferences when presented with specific input data. More sophisticated backdoor attacks hide triggers, making input data modification unnecessary, and concealing any visual indications of tampering.

Evasion attacks aim to disrupt proper classification or inference without directly attacking the AI model or training data. Instead, input data is altered to induce incorrect inferences. This attack method is akin to spammers altering messages to evade detection by anti-spam software.

Stealing threat model can target all forms of data in an AI system, from training data to inference results. Theft of a training dataset allows the creation of an identical AI model. Alternatively, the theft of input data and its corresponding inferences can be used to reverse engineer the trained AI model. This implies that real inputs and inferences can serve as a set of "training" data [40].

Figure 6-2 provides a graphical representation of the threat models and their targets. With four distinct threat models and five diverse targets, securing an AI system demands a robust solution supporting multiple security protocols, including integrity checking, user authentication, secure boot, and others. Leveraging a secure, PUF-based root of trust.

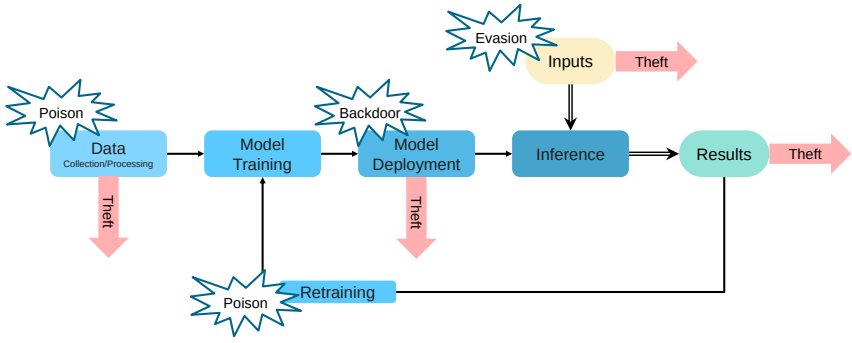


Figure 6-2 Threats to AI systems: Training to deployment.

6.1.2. PUF-based Solutions for AI

A trained AI model represents a substantial investment of both time and financial resources for a company, encompassing activities such as collecting and preparing training data, maintaining the model's integrity post-deployment, and conducting ongoing retraining. In the subsequent sections, we will delve into the paramount role of the primary cryptographic function, namely privacy, in safeguarding a company's investment in its AI model.

Encryption algorithms, commonly employed to uphold privacy, heavily rely on the secrecy of the data encryption key (DEK). The compromise or theft of a DEK poses a significant threat, eroding the privacy of the secured data. Hence, for AI systems, prioritizing key protection is of utmost importance when evaluating various security solutions. Comparing to other solutions, PUF-based solutions provide two primary advantages, ensuring that only authorized parties can access stored keys.

The first advantage lies in the storage cell technology itself. Traditional solutions often employ electrical fuse (eFuse) technology for key storage, which, while simple and reliable, is susceptible to reverse engineering. Unfortunately, eFuses can be deciphered even when the chip is powered down. A hacker can decap the integrated circuit sufficiently to reveal the fuse array and read the stored values using a powerful microscope.

As illustrated in *Figure 6-3 (a),* an electron microscope photo displaying the visible difference between intact and blown eFuse

cells. In contrast, *Figure 6-3 (b)* exhibits programmed and unprogrammed storage cells using anti-fuse technology, highlighting the advantage of a PUF-based solution.

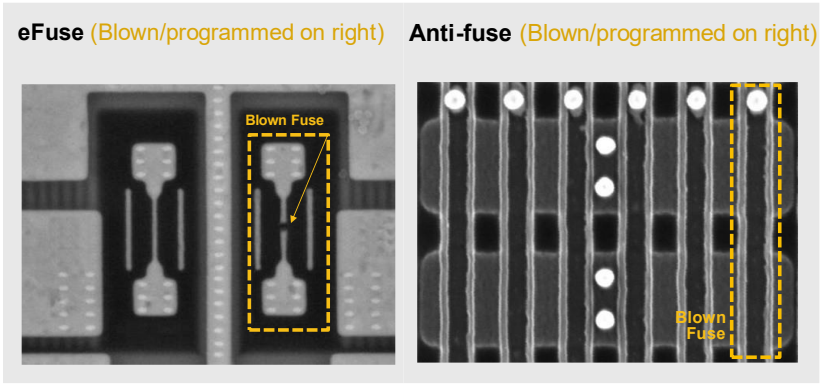


Figure 6-3 (a) eFuse microscope photo. (b) Anti-fuse microscope photo.

The second advantage of PUF-based solutions builds upon an array of anti-fuse storage cells, incorporating additional anti-tampering protections and discovery prevention designs to enhance the security of keys and other sensitive data, as outlined in *Table 6-2.* These measures encompass a range of safeguards, from facilitating ciphertext storage (#3) to introducing randomization in reads, thereby increasing the complexity of side-channel analysis (#13). PUF-based solutions employ a multi-layered protection scheme, effectively thwarting attempts to pilfer or intercept keys from secure storage.

Table 6-2 Anti-tampering measures

Types		Security Features
Invasive Attack	1	Intrinsic Physical Security
	2	Voltage Contrast Attack Countermeasures
	3	Data Address X-Y Scrambler and IO Shuffler using PUF
Semi-Invasive Attack	4	(Optional) Top Metal Shielding
	5	Security-oriented IP Layout
	6	Active Sense-Amplifier READ Protection
	7	Hidden and Obfuscated Data Interface (inside macro)
	8	Output Data Fault Detection
Non-Invasive Attack	9	Pin Integrity Protection on Mode and Array Selection
	10	Word Lock; Non-Accessible Post-Masking (on OTP)
	11	Zeroization and Post-Masking (on PUF)
	12	Built-in Secure Repair and Test-mode Lock
	13	(Optional) Random Dummy Insertion READ
	14	PUF Health Check
	15	Fault Injection Prevention (Mode/Address/Post-masking)
	16	Unified Write Power to Prevent Electrical Analysis
	17	Power Detection -VDD/VDDIO Floating

6.1.3. AI Target Vulnerabilities

To effectively safeguard AI systems, it is imperative to identify potential vulnerabilities and implement appropriate protective measures. In this subsection, we will present a concise introduction to countermeasures and corresponding PUF-based solutions designed to address the five key weaknesses of AI systems. As depicted in *Figure 6-4*. target the protection of the processor, training data, trained AI model, input data for inferences, and inference results. The following assets in an AI system are the focal points for protection:

1. Processor itself
2. Training data used for AI model training
3. Trained AI model
4. Input data for making inferences
5. Inference results

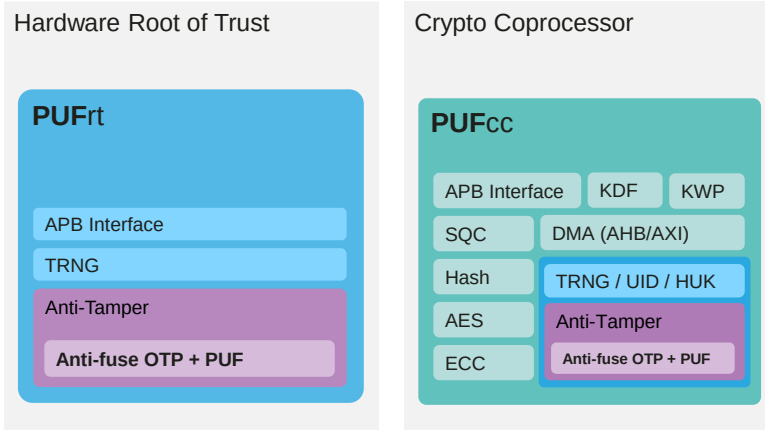


Figure 6-4 Root of trust and crypto coprocessor from PUFsecurity.

Brief insights into the functions providing corresponding protection will be touched upon. However, detailed descriptions of these functions will be provided in Section 6.3, as they will reappear in Section 6.2.

6.1.3.1. AI Processors

Among the five targets previously mentioned, the sole hardware asset is the AI processing unit itself. Although general-purpose CPUs or graphics processing units (GPUs) can serve AI applications, specialized processors optimized for AI, such as Tensor Processing Unit, Vision Processing Unit, Graph Streaming Processor, and Reconfigurable Dataflow Unit, have been designed by many companies.

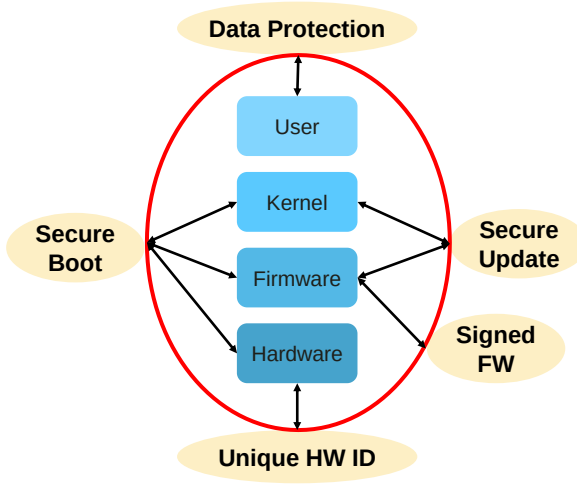


Figure 6-5 Shielding the system stack.

Despite the diverse names, AI processors fundamentally constitute hardware, making them susceptible to tampering of both the hardware itself and the firmware necessary for correct operation. Fortunately, anti-tampering and security designs developed for general-purpose CPUs can be adapted for AI processors. As the processor represents only one facet of the hardware layer within a system stack, a comprehensive protection strategy is essential to shield the entire system stack operations from potential attacks, as illustrated in *Figure 6-5*. While detailed coverage of PUF-based solutions and their supported functions will be presented in Section 6.3, *Table 6-3* outlines common threats to AI processors, countermeasures, and recommended solutions for threat mitigation.

Table 6-3 AI processor threats, countermeasures, and solutions

Threat	**Countermeasure**	**PUF-based Solution /function**
Boot Code Tampering	Secure Boot	PUFcc /non-repudiation
Firmware Tampering	Signed Firmware	PUFcc /non-repudiation
Key Leakage	Secure Key Management	PUFcc /key exchange
Cloned Hardware	Unique Hardware Identification	PUFrt, PUFcc /authentication
System Update Tampering	Secure Update	PUFcc /non-repudiation
Data Exfiltration	Data Protection (at rest/in transit)	PUFcc /privacy

In *Table 6-3*, three of the targets — boot code, firmware, and system update — are associated with tampering, signifying instances where a critical system executable has been altered to benefit hackers. If any of these three executables undergo unauthorized modifications, the AI system may, at best, fail to function correctly and, at worst, pose potential harm to human lives. This is especially critical in scenarios where the AI system is responsible for vital functions, such as monitoring vital signs in a hospital setting. As highlighted earlier, the security of a system hinges on keeping keys secure to protect private data meant solely for authorized users. A robust key management system is indispensable to ensure that these keys remain accessible only to secure system processes.

Uniquely identifying AI processors is crucial to thwart cloning and the production of counterfeit versions that could be illicitly circulated. The authorization function for the processor relies on UIDs for each processor, preventing cloning by enabling a means to verify the authenticity of each processor based on their IDs. These UIDs may be directly derived from the PUF.

6.1.3.2. AI Training Data

The next important asset to protect in an AI system is the training data. Training data can be tampered with to alter the resulting AI model or can be stolen to create a clone of the official AI model. The training data can also be mined for private and personal data that may be a part of the dataset, especially when it was not made anonymous when collected for training purposes. While fully

homomorphic encryption (FHE) promises to take care of keeping training data anonymous, fast, and practical, implementations of FHE have remained out of reach for modern day cryptographers [41].

Table 6-4 AI training data threats, countermeasures, and solutions

Threat	Countermeasure	PUF-based Solution /function
Poisoning	Encryption, Integrity Checking	PUFcc /privacy/integrity
Stealing	Encryption	PUFcc /privacy
Leaked Private Data	Encryption	PUFcc /privacy

Each of the above attacks listed in *Table 6-4* all targets the training data, but each successful attack will have different results. A poisoned AI model will yield wrong results when presented with input data. An adversary successfully creating a cloned AI model represents lost proprietary company information/resources. Leaked confidential data remains an ongoing risk for those people whose privacy has been compromised.

6.1.3.3. Trained AI Models

As the embodiment of the time, effort, and resources of the AI development team spent on training and creating an AI model, it is an equally important asset that requires protection throughout its deployment and operational lifetime. AI models are particularly

vulnerable to tampering and theft, as they are themselves data, such as the finalized weights and biases of a neural network after training. The below *Table 6-5* lists the threats to AI models, their corresponding countermeasures, and the recommended PUF-based solution to implement said countermeasures:

Table 6-5 AI model threats, countermeasures, and solutions

Threat	Countermeasure	PUF-based Solution /function
Backdoor	Encryption, Integrity Checking	PUFcc /privacy/integrity
Theft	Encryption	PUFcc /privacy
Replacement	Integrity Check with Signature	PUFcc /integrity/non-repudiation

A backdoor attack is one such form of tampering to be wary of. Either by modifying the training data to influence the development of the AI model or by inserting additional neurons into an AI model that will listen for a specific set of triggers, the model is nudged towards making inaccurate inferences.

Another attack called “replacement” will remove the trained original neural network and replace it with one of the hacker’s choosing. The original network is substituted with an entirely new one that can effectively disable the AI system by producing untrustworthy or harmful results. Finally, there is the always present threat of model theft. Instead of stealing just the input training data to create a clone of a competitor’s AI, a bad agent

may resort to the outright copying of an entire AI model, so that they may save the trouble of training a model themselves.

6.1.3.4. AI Model Input Data

Ensuring the protection of input data during the use of an AI model is crucial to prevent tampering that could compromise the accuracy of output results. A successful attack on input data can lead to incorrect inferences, as the output is derived from inaccurate inputs.

Furthermore, similar to training data, input data is susceptible to privacy invasions through analysis, necessitating measures to safeguard against the extraction of private information. *Table 6-6* outlines the threats to AI input data, corresponding countermeasures, and recommended PUF-based solutions for their implementation. These measures are essential for maintaining the integrity and privacy of input data during the operational phase of the AI model.

Table 6-6 AI input data threats, countermeasures, and solutions

Threat	**Countermeasure**	**PUF-based Solution /function**
Evasion	Encryption, Integrity Checking	PUFcc /privacy/integrity
Leaked Private Data	Encryption	PUFcc /privacy

6.1.3.5. AI Inference Results

The resulting inference output needs to be protected from alterations and theft, or else the wrong conclusions will be drawn from tampered results. Worse still, bad agents can reap the benefits of a well-trained AI model without needing to invest the time and resources to create one. *Table 6-7* lists these threats to AI inference results, their corresponding countermeasures, and the recommended PUF-based solution to implement said countermeasures.

Table 6-7 AI inference results threats, countermeasures, and solutions

Threat	Countermeasure	PUF-based Solution /function
Tampering	Encryption, Integrity Checking	PUFcc /privacy/integrity
Stealing	Encryption	PUFcc /privacy

6.2. PUF-based Security for the Internet of Things

The Internet of Things (IoT) is a vast field categorized into consumer, enterprise, and industrial applications based on intended usage. Despite challenges posed by the Covid-19 pandemic and a global semiconductor shortage, the IoT industry exhibited an 8% growth in 2021, with an estimated installation of over 27 billion devices by 2025 [42].

Given the diverse applications within the consumer, enterprise, and industrial categories, defining a universal "IoT architecture" applicable to all cases proves challenging. Nevertheless, when discussing the fundamental workings of IoT devices at a higher level, a widely accepted "three-layer IoT architecture" is often utilized in IoT literature [43]. Comprising the perception, network, and application layers, these levels represent the interface between the user and the IoT edge device, with the application layer situated at the user interface and the perception layer positioned at the edge *Figure 6-6*. provides a visual representation of this commonly employed three-layer IoT architecture.

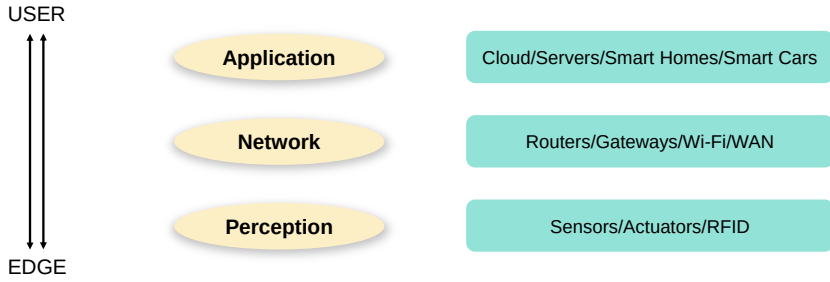


Figure 6-6 Three-layer architecture of IoT.

At the application layer of the IoT architecture, users interact with IoT-enabled appliances like smart homes or cars, as well as with the cloud that delivers requested data and applications. Given that the application layer is primarily implemented in software, discussions of security in this realm are beyond the scope of this book, which centers on PUF-based hardware security solutions.

The network layer, situated between the user (application layer) and the edge (perception layer), manages the communication of data and requests. Standard network hardware such as routers and gateways, along with established protocols like Wide Area Network (WAN), Wi-Fi, and LoRa (Long Range), facilitate this communication. As the hardware and protocols of the network layer are standardized with well-defined security measures, the forthcoming sections of this book will concentrate on PUF-based solutions for IoT, specifically targeting the perception layer.

The perception layer, where edge devices like sensors and cameras operate, serves as the initial connection between the physical world and the digital system. At this layer, IoT devices observe or measure a target and transmit the collected data back to the application layer.

This section on IoT is organized into three subsections: first, descriptions of threat models facing IoT devices at the perception layer; second, a brief introduction to PUF-based solutions for IoT devices at the perception layer; and finally, a closer examination of vulnerable targets within an IoT system. The order of these targets in subsection 6.2.1, consisting of data, hardware, and firmware, will mirror the same sequence used in subsection 6.2.3.

6.2.1. IoT Threat Models at the Perception Layer

Attacks can threaten all three layers of the IoT architecture: perception, network, and application. However, as threats in the

application layer are normally handled through software, and since the network layer is taken care of by already established industry standard communication protocols, such as WiFi Protected Access, or WPA, we will only focus on those threat models at the perception layer.

Threats at the perception layer will target one of three areas: data, hardware, and firmware. These three target areas are seen in *Figure 6-7*, which are especially vulnerable to hacking at the perception layer. As perception layer data is transmitted to the network layer, it is open to attack in both the network and perception layer, as well as at the boundary between the two.

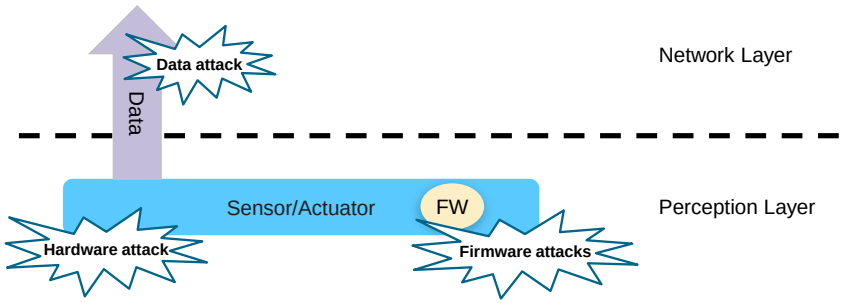


Figure 6-7 Threats at the perception layer.

Data attacks at the perception layer target the data entering or leaving IoT nodes. These attacks encompass eavesdropping and false data injection, with the goal of either stealing sensitive data and keys or forging the data received by sensor nodes. An example includes spoofing the location of a Global Positioning

System (GPS) sensor, leading to incorrect tracking about the device's location.

Hardware attacks focus on the node itself. The objective is to capture the node, subject it to the control of an unauthorized agent, and repurpose it for activities beyond its original design. In cases where capturing is unfeasible, hackers may attempt to clone the node, creating a counterfeit node automatically under their control. Additionally, if capturing or cloning isn't achievable, malicious actors might resort to attacks designed to deplete the battery life of an IoT node in the perception layer. This strategy enables hackers to deactivate enough installed nodes to effectively diminish the size of the deployed IoT network, rendering it non-operational or ineffective.

Firmware attacks involve attempts to modify or disable the firmware necessary for a node's proper operation. Firmware is vulnerable not only when at rest but also during installation or over-the-air (OTA) updates for devices already in the field. These attacks on firmware integrity can compromise the functionality and security of IoT nodes.

6.2.2. PUF-based Solutions for IoT

The focus of the solutions for IoT systems centers on the perception layer situated at the "edge." As the array of IoT

applications continues to expand, the proliferation of devices at the edge experiences explosive growth due to the increasing demand for enhanced control and monitoring of actuators and sensors. Given that each edge device is assigned to a specific location, phenomenon, unit, or machine, it is imperative for these devices to maintain distinct identities. This means that each device, along with its sensors or actuators, must possess a unique identifier.

Similar to the critical role of key protection in securing an AI system, the generation and security of unique identifiers for edge devices in an IoT system hold equal importance. To differentiate individual IoT devices at the edge, each device must possess its own unique identifier. The generation of this unique ID occurs through a "key provisioning" process, which can be implemented either internally via a PUF-based solution or externally through a traditional method. The distinctions between internal and external key provisioning are illustrated in *Figure 6-8*.

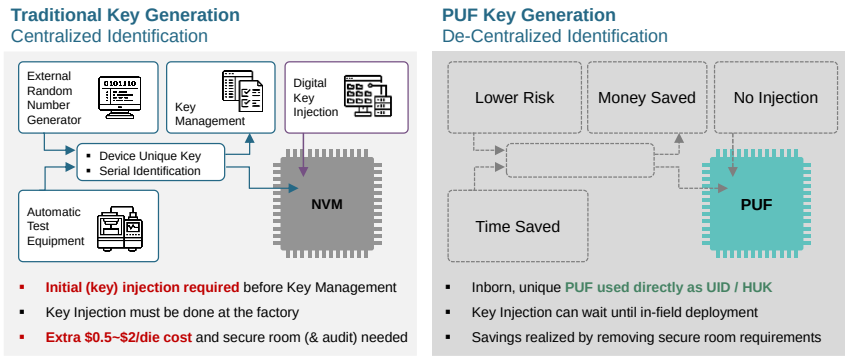


Figure 6-8 Internal versus external key provisioning.

Edge devices, situated farthest from the cloud and users at the IoT application layer, often reside in remote locations without access to a wired power supply, relying on battery power. Thus, these devices should minimize power consumption to extend the time between charges. PUF-based security solutions for edge devices offer pre-integrated analog and digital components in one unit, reducing the power cost that two separate analog and digital blocks would otherwise incur.

Traditional security solutions are implemented entirely in Register Transfer Logic (RTL) to create the required logic blocks. However, they still necessitate an analog memory, such as an array of electronic fuses, for storing keys and sensitive data. Digital security companies lacking expertise in analog design must engage the services of a memory specialist for these storage blocks and a True Random Number Generator (TRNG) missing from their designs. Consequently, at least two discrete modules—one analog and one digital—must be combined in the device to form a complete security solution.

The new PUF-based security solution combines all necessary components into a single unit. There is no need to integrate the analog and digital components first while applying a PUF-based solution. As illustrated in *Figure 6-9*, the integration of digital and analog components for PUFrt and PUFcc into edge devices is achieved in one setting, making the implementation process simpler and more efficient.

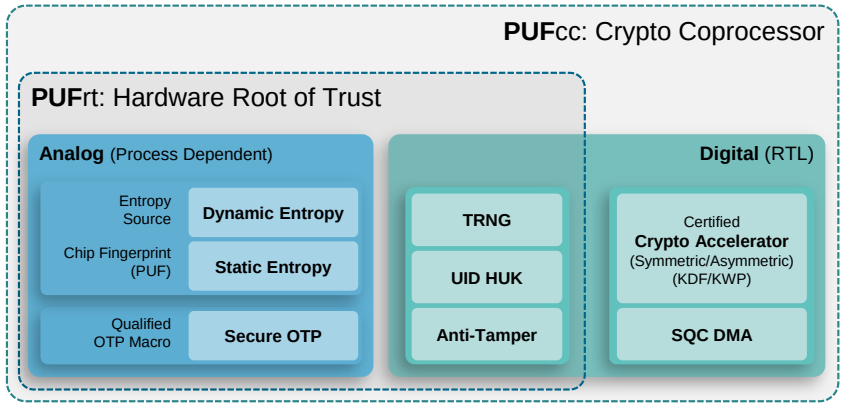


Figure 6-9 Digital-analog integrated solutions.

6.2.3. IoT Target Vulnerabilities

The upcoming sections of this book will delve into the weaknesses of data, hardware, and firmware, respectively, based on the section 6.2.1 mentioned threat models.

6.2.3.1. IoT Perception Layer Data

Data attacks in the perception layer can be categorized as either an attempt to steal data (eavesdropping) or to modify or tamper the data to affect normal operations. These two threats are shown in *Table 6-8*. IoT perception layer data may refer to either the data that is being collected from the sensors or the commands that are being sent to the actuators at the perception layer. In either case, if the perception layer data is tampered with, the wrong data will be reported back to the processor, or the wrong commands or

incorrect settings will be implemented. And for incidents of theft, a malicious actor will be able to steal the data from another's installed sensor network, thus saving the trouble and costs of deploying their own devices.

Table 6-8 IoT data threats, countermeasures, and solutions

Threat	Countermeasure	PUF-based Solution /Function
Theft	Encryption	PUFcc /privacy
Hacking/Tampering	Integrity Check, Auth Encryption	PUFcc /integrity

6.2.3.2. IoT Perception Layer Hardware

IoT hardware at the perception layer is vulnerable to capture, cloning, or disabling through battery exhaustion. Given that these attacks primarily target the hardware, it is imperative that anti-tampering designs are standard features for all PUF-based solutions, as outlined in section 6.

Please note that in *Table 6-9*, "authentication" encompasses both user and device authentication, each of which will be addressed separately. User authentication is employed to prevent the capture and misuse of an IoT edge device, ensuring that unauthorized users cannot gain control. Meanwhile, to thwart cloned devices from accessing IoT system resources, devices must undergo authentication before being granted access.

Table 6-9 IoT hardware threats, countermeasures, and solutions

Threat	Countermeasure	PUF-based Solution /function
Capture	User Authentication	PUFcc /authentication
Clone	Unique ID	PUFrt, PUFcc /authentication
Exhaust	User Authentication	PUFcc /authentication

6.2.3.3. IoT Perception Layer Firmware

Attacks targeting firmware exhibit similarities to those directed at data within the perception layer, given that firmware is inherently a form of data. Consequently, safeguarding against theft or unauthorized alterations of firmware, both at rest and during update processes, becomes paramount. In *Table 6-10* , a comprehensive enumeration of threats confronting firmware at the perception layer is presented.

The firmware governing edge IoT devices remains susceptible to tampering and theft across various phases, encompassing pre-deployment, deployment to the field, and instances during firmware updates. An adversarial entity gaining access to an IoT device's firmware assumes a potent position to manipulate its functionality or potentially render the device entirely inoperable. Hence, it is imperative that firmware at the perception layer remains untainted by malware or any malicious code, whether introduced during the manufacturing stage, in-field operations, or update procedures.

Table 6-10 IoT firmware threats, countermeasures, and solutions

Threat	Countermeasure	PUF-based Solution /function
Theft	Encryption	PUFcc /privacy
Hacking/Tampering	Integrity Check, Digital Signature	PUFcc /non-repudiation

6.2.4. Cryptographic Functions for Mitigation of Vulnerabilities

In this subsection, we systematically categorize the five foundational cryptographic functions employed for mitigation, offering further elaboration on the functions delineated in sections 6.1 and 6.2. Additionally, this section serves as an extension of section 1.3, serving as a point of reference for readers engaged in the formulation of protective measures. The targets, threats, and solutions from the relevant tables in the previous sections will be thoroughly illustrated in the following sections.

6.2.4.1. Privacy/Confidentiality Function

The primitive function of privacy is likely the one that is most well-known for people not familiar with cryptography. Stemming from the need to keep secret messages private, this function relies heavily on ciphers to prevent unauthorized users from accessing or stealing sensitive information.

Table 6-11 System threats mitigated with privacy

System	Target	Threat	Crypto Function for Mitigation
AI	Processor	Data exfiltration	Privacy
AI	Training data	Poisoning	Privacy
AI	Training data	Stealing	Privacy
AI	Training data	Leaked private data	Privacy
AI	Trained model	Backdoor	Privacy
AI	Trained model	Theft	Privacy
AI	Input data	Evasion	Privacy
AI	Input data	Leaked private data	Privacy
IoT	Data	Theft	Privacy
IoT	Firmware	Theft	Privacy

Table 6-11 lists the threats that are handled by the function of privacy. Utilizing a robust cipher and a highly secure data encryption key, like AES or ChaCha, sensitive data is encrypted by PUFcc. Even if the data is stolen or intercepted, it remains inaccessible without the correct encryption key. This privacy function operates securely within PUFcc's secure boundaries, fortified by built-in anti-tampering designs. This process acts as a strong deterrent against hackers attempting to pilfer data from the protected system, as depicted in *Figure 6-10*.

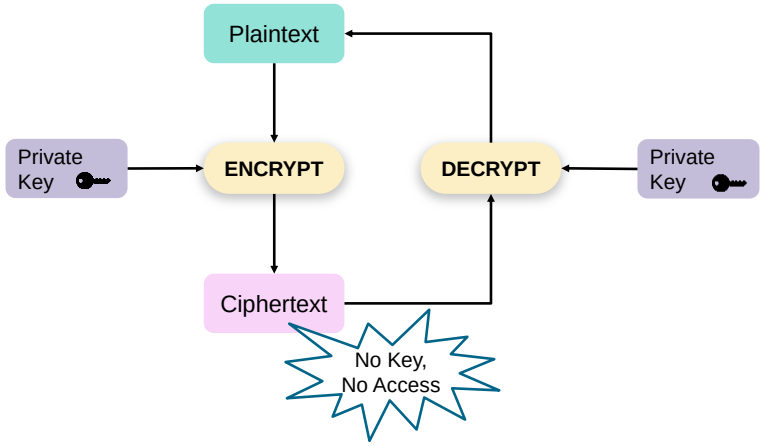


Figure 6-10 Privacy through encryption.

6.2.4.2. Authentication Function

The function of authentication encompasses both user and device authentication. In the context of IoT edge devices, user authentication serves as a preventive measure against capture or misuse, ensuring that unauthorized users are unable to seize control. Simultaneously, device authentication plays a pivotal role in thwarting cloned devices from gaining access to IoT system resources, necessitating devices to authenticate themselves before being granted entry. *Table 6-12.* succinctly enumerates the threats to AI/IoT systems effectively addressed by the authentication function.

Examining device authentication in detail, this process relies on unique identifiers assigned to each device or processor. These

identifiers serve to forestall cloning by establishing a mechanism to verify the authenticity of each device based on its individual ID.

Table 6-12 System threats mitigated with authentication

System	Target	Threat	Crypto Function for Mitigation
AI	Processor	Clone	Authentication
IoT	Hardware	Capture	Authentication
IoT	Hardware	Clone	Authentication
IoT	Hardware	Exhaust	Authentication

The internal key provisioning process utilized in PUFrt and PUFcc allows each instance to self-create its own unique ID. This all-internal process eliminates the need for a secured, clean-room environment traditionally required for the key injection process, saving time and money over the conventional key provisioning flow. PUFrt and PUFcc support the primary function of authentication by providing an economical and secure method for the generation of individual IDs for each processor. The contrast between an authentic device with an installed PUF versus a cloned device that does not have a PUF, is seen in *Figure 6-11*.

Examining user authentication in the context of preventing hackers from taking control of an edge IoT device or engaging in malicious acts such as depleting the device's battery, a certificate-based authentication process proves effective. PUFcc adeptly handles

this scenario through its incorporation of a public key co-processor (PKC) with support for the digital signature algorithm (DSA).

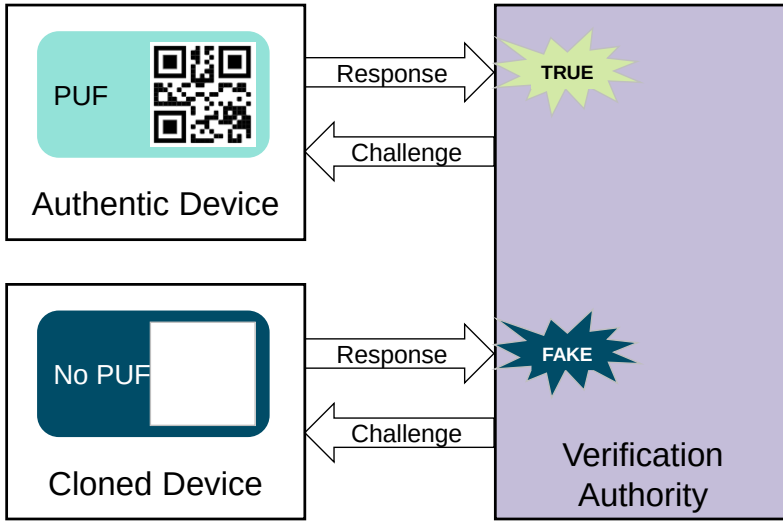


Figure 6-11 Unclonable PUF-based IDs for authentication.

In this process, a user possessing a previously issued digital certificate from a trusted authority, affirming the user's identity and public key, signs a piece of data using their private key. Subsequently, the user forwards the signed data and digital certificate to PUFcc for verification. Following the successful verification of the user's digital signature using their public key derived from the digital certificate, authorization is granted. Consequently, PUFcc leverages the core function of authentication to exclusively permit access to authorized users, concurrently thwarting unauthorized access attempts by hackers.

Figure 6-12 illustrates how pre-established digital certificates by a certificate authority (CA) and the DSA can authenticate users.

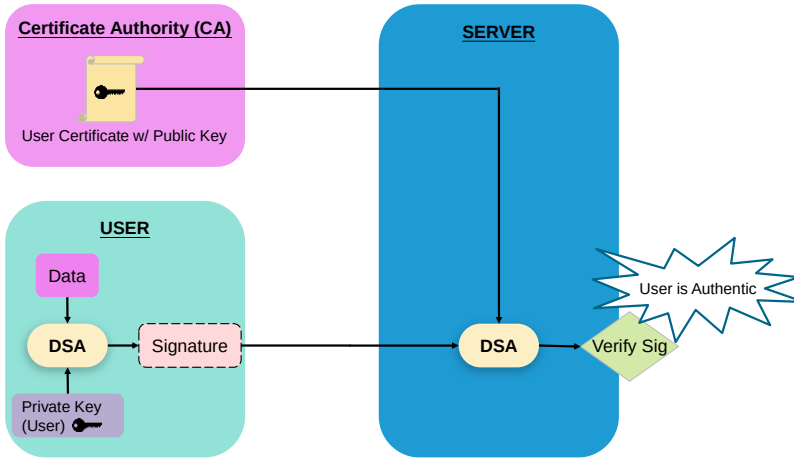


Figure 6-12 Authentication through CA and digital signature algorithm.

6.2.4.3. Integrity Function

The integrity function plays a crucial role in enabling a system to detect evidence of data tampering. In the realm of AI and IoT systems, upon detecting any indication of data tampering, the system can promptly trigger an alert and cease the utilization of the modified data or executable. This rapid response mitigates the potential harm caused using compromised information, code, or firmware. The integrity function is instrumental in addressing the threats outlined in *Table 6-13*.

Table 6-13 System threats mitigated with integrity

System	Target	Threat	Crypto Function for Mitigation
AI	Training data	Poisoning	Integrity
AI	Trained model	Backdoor	Integrity
AI	Trained model	Replacement	Integrity
AI	Input data	Evasion	Integrity
IoT	Data	Hacking/tampering	Integrity

To check for tampered data, such as through a poisoning attack, the hash digests of the dataset are compared to check for signs of unauthorized modification, seen in *Figure 6-13*. With either its SHA-2 (secure hash algorithm) or SM3 hash engine, PUFcc generates the “original data” digest and the “to check data” digest and compares the two.

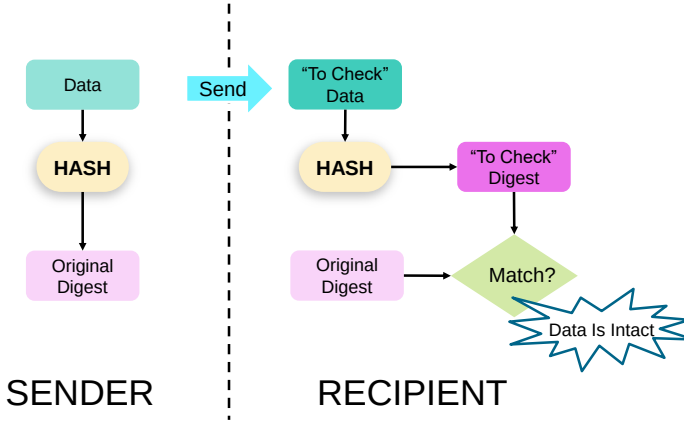


Figure 6-13 Integrity through secure hash.

If the digests do not match, then the current training dataset has been modified from the original version, indicating a possible poisoning attack has taken place. In this way, PUFcc supports the primary function of integrity to detect when a poisoning attack may have taken place, alerting the main system to make the appropriate response.

Alternatively, employing authenticated encryption (AE) methodologies, such as Galois Counter Mode (GCM) or Counter with CBC-MAC (CCM) modes of an AES cipher engine, enables the simultaneous generation of "original" and "to check" authentication tags during the creation of ciphertext for protected data. The comparison of authentication tags mirrors the process with hash digests, allowing for the straightforward detection of any signs of data tampering or hacking. *Figure 6-14* provides an example of utilizing authenticated encryption for integrity checking, wherein the co-generated tag during encryption serves as a mechanism for tamper detection.

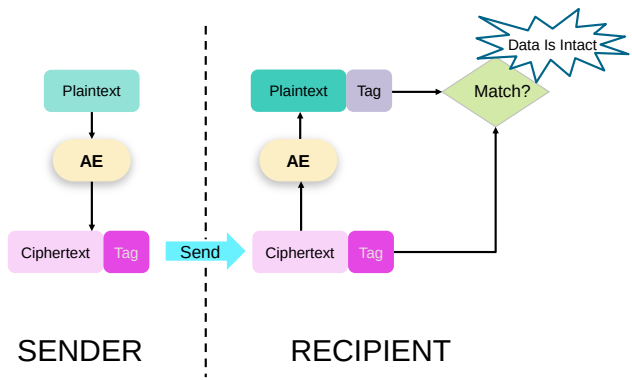


Figure 6-14 Integrity through authenticated encryption.

6.2.4.4. Non-Repudiation Function

The primary cryptographic function of non-repudiation allows confirmation that the claimed author of the data in question is in fact the one who created it. For example, to ensure that the firmware of an edge IoT device is safe to use with no malware hidden in the source code, non-repudiation guarantees the source/originator of the firmware is who they claim they are. Thus, so long as the true author is the same as the official factory source, it is a strong guarantee that the firmware has not been modified from the original, official factory's design. Non-repudiation mitigates the following threats listed in *Table 6-14.*

Table 6-14 System threats mitigated with non-repudiation

System	Target	Threat	Crypto Function for Mitigation
AI	Processor	Boot code tampering	Non-repudiation
AI	Processor	Firmware tampering	Non-repudiation
AI	Processor	System update tampering	Non-repudiation
AI	Trained model	Replacement	Non-repudiation
IoT	Firmware	Hacking/tampering	Non-repudiation

To protect against tampering attacks, the function of non-repudiation is used to ensure that the origin of those executables is from a trustworthy source and that they have not been modified.

As a simple illustration of implementing non-repudiation, a digital signature of the executable is created by the originator, which includes a hash digest of the executable and signed using the originator's private key. After it is sent to the user, along with the originator's public key, the user can verify that the originator's signature is valid using the public key (with DSA).

Subsequently, the received executable undergoes hashing to generate an additional digest through the HASH engine. This newly created digest is then compared with the originally received hash digest. In the event of tampering, the two hash digests will not align. PUFcc, serving as the secure crypto co-processor developed by PUFsecurity, facilitates such a workflow. *Figure 6-15* provides an illustrative example of implementing non-repudiation using a secure hash and digital signature.

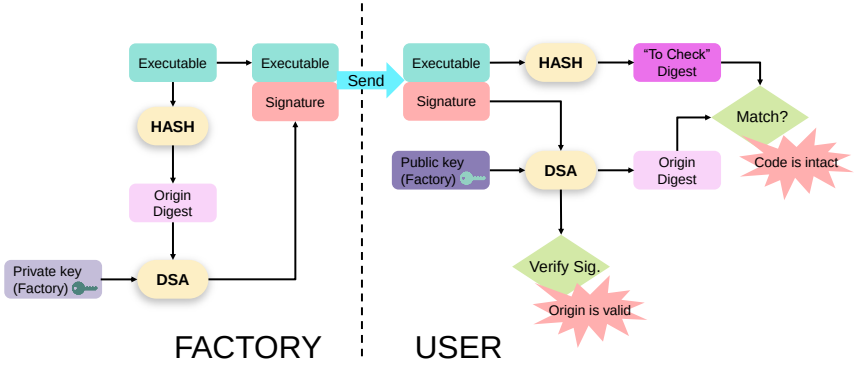


Figure 6-15 Non-repudiation through digital signature algorithm.

6.2.4.5. Key Exchange Function

The primary function of key exchange plays an important role in a system's key management system. Allowing for the secure transfer of secret keys between the included parties, key exchange ensures that secret keys may remain private to all others when said keys need to be shared among the authorized users. The function of key exchange helps prevent any key leakage during transit, as shown here in *Table 6-15*.

Table 6-15 System threats mitigated with key exchange

System	Target	Threat	Crypto Function for Mitigation
AI	Processor	Key leakage	Key exchange
IoT	Network layer	Key leakage	Key exchange

A private key, like any other sensitive data, needs to be protected while at rest (in storage) and while in transit to prevent it from being leaked. Having a secure way of exchanging keys is an important part of any good key management system for the safety of private keys. Offering three different modes to securely wrap keys for safe transport, all based on the NIST AES cipher, PUFcc's key wrapping module (KWP) supports the primary cryptographic function of key exchange, which includes the cipher block chaining (CBC) mode, as seen in *Figure 6-16.*

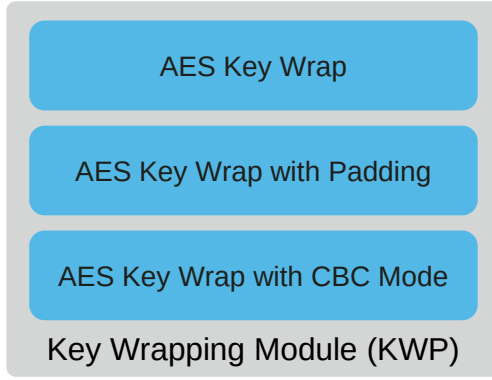


Figure 6-16 Key wrapping module of PUFcc.

7

Conclusion of the book. This chapter provides a concise review of the key points from all preceding chapters.

7. Conclusion

This book serves as an extensive and comprehensive guide, aiming to equip readers with a robust foundation in PUF-based security solutions. It commences its exploration by anchoring itself in the fundamental principle that security is intricately linked to trust. The overarching goal of the content is to lead readers through a nuanced understanding of the potential challenges and various forms of attacks that can undermine system trust, empowering them to develop sophisticated strategies for the establishment and perpetual maintenance of trust. The narrative intricately delves into the five primitive cryptographic functions—confidentiality, authentication, integrity, non-repudiation, and key exchange.

Before immersing readers in the intricacies of PUF-based solutions designed for these cryptographic functions, the book adopts a panoramic approach by providing readers with an exhaustive background on security. This includes a profound focus on the contemporary trends shaping hardware security. The narrative highlights how modern security proposals are embracing a holistic direction, necessitating the safeguarding of all three forms of system data—data at rest, data in use, and data in transit. Furthermore, the discourse underscores the imperative that security considerations extend across the entire operating lifecycle of a product, challenging the conventional post-deployment focus. The discussion on hardware security trends culminates in a comprehensive overview of prevailing security standards and certifications, such as FIPS 140-2 and PSA Certified, which play a

pivotal role in sustaining a consistent and universally accepted level of hardware security.

Pivoting to the core theme, the book meticulously defines the concept of a Root of Trust (RoT). With this foundational understanding in place, the spotlight turns to the two prominent PUF-based solutions—PUFrt (Root of Trust) and PUFcc (Crypto Co-processor). Dedicated sections meticulously cover their underlying philosophy, design principles, architectural intricacies, functional blocks, and supported operations. The section on PUFcc extends its purview by providing insightful details about auxiliary modules that effectively broaden PUF-based security, encompassing other critical system blocks, including non-volatile system memory.

The concluding section of the book functions as a bridge, effectively connecting the abstract discussions on security with the practical intricacies of PUFrt and PUFcc. This is achieved through an in-depth exploration of real-world applications in the realms of AI and IoT. Readers are guided through a detailed examination of how security safeguards crucial data in these applications, covering diverse aspects such as threat models, vulnerable targets, and illustrative examples demonstrating how PUF-based solutions seamlessly thwart attacks through the effective implementation of the five primitive cryptographic functions.

In closing, the authors express their sincere hope that readers not only emerge from this book with a heightened understanding of trust and security but also with a renewed interest in the dynamic

and evolving realm of PUF-based hardware security. Constructive comments and suggestions to enhance the overall readability and depth of this book are not only welcomed but strongly encouraged.

8

Appendix of the book, including Abbreviations, References, and Index.

Abbreviations

A	AE	Authenticated Encryption
	AES	Advanced Encryption Standard
	AHB	Advanced High-performance Bus
	AI	Artificial Intelligence
	AMBA	Advanced Microcontroller Bus Architecture
	APB	Advanced Peripheral Bus
	API	Application Programming Interface
	AXI	Advanced eXensible Interface
B	BER	Bit Error Rate
C	CA	Certificate Authority
	CAVP	Cryptographic Algorithm Validation Program
	CBC	Cipher Block Chaining
	CCA	Confidential Compute Architecture
	CCM	Counter with CBC-MAC
	CFB	Cipher Feedback
	CMAC	Cipher-based MAC
	CMOS	Complementary Metal-Oxide Semiconductor
	CPU	Central Processing Unit
	CRYPTO	Cryptographic
	CTR	CounTeR
D	DEK	Data Encryption Key
	DMA	Direct Memory Access
	DRBG	Deterministic Random Bit Generator
	DSA	Digital Signature Algorithm
E	ECB	Electronic Code Book
	ECDH	Elliptic Curve Diffie-Hellman
	ECDSA	Elliptic Curve Digital Signature Algorithm
F	FAB	FABrication plant

	FPGA	Field Programmable Gate Array
	FW	Firmware
G	GCM	Galois Counter Mode
	GDSII	Graphic Design System, II (Roman numeral)
	GE	Gate Equivalent
	GPS	Global Positioning System
	GM/T	GM Standard for “Cipher Code” (China)
	GPU	Graphics Processing Unit
H	HD	Hamming Distance
	HMAC	Hash-based MAC
	HSM	Hardware Secure Module
	HUK	Hardware Unique Key
	HW	Hamming Weight
I	ID	IDentification
	IEEE	Institute of Electrical and Electronic Engineers
	IETF	Internet Engineering Task Force
	IID	Independent and Identically Distributed
	IoT	Internet of Things
	IP	Intellectual Property
	ISSCC	International Solid-State Circuits Conference
K	KA	Key Array
	KBKDF	Key Based KDF
	KDF	Key Derivation Function
	KDK	Key Derivation Key
	KEK	Key Encryption Key
	KW	Key Wrap
	KWP	Key WrapPing
M	MAC	Message Authentication Code
	MHz	Mega Hertz
N	NA	No Access
	NIST	National Institute of Standards and Technology

	NVM	Non-Volatile Memory
O	OFB	Output Feedback
	OSCCA	Office of State Commercial Cryptography Administration
	OTA	Over The Air
	OTP	One-Time Programmable
P	PBKDF	Password Based KDF
	PKC	Public Key Cryptography
	PMP	Physical Memory Protection
	PRF	Pseudo Random Function
	PRTC	PUFrt Interface Control
	PUF	Physically Unclonable Function
	PUFcc	PUFsecurity Crypto Coprocessor
	PUFrt	PUFsecurity Root of Trust
Q	QT	Quantum Tunneling
R	REE	Rich Execution Environment
	RO	Read Only
	RoT	Root of Trust
	RSA	Rivest-Shamir-Adleman
	RTL	Register Transfer Logic
	RW	Read Write
S	SDK	Software Development Kit
	SHA	Secure Hash Algorithm
	SoC	System on Chip
	SQC	Sequencer Logic Control

References

[1] P. Kocher, J. Jaffe, and B. Jun, “Differential power analysis,” *Annual International Cryptology Conference*., Springer, Berlin, Heidelberg, 1999.

[2] G. Piret, and J.-J. Quisquater, “A differential fault attack technique against SPN structures, with application to the AES and KHAZAD.” *International workshop on cryptographic hardware and embedded systems,* Springer, Berlin, Heidelberg, 2003.

[3] D. Nedospasov, et al., “Invasive PUF analysis.” *2013 Workshop on Fault Diagnosis and Tolerance in Cryptography,* IEEE, 2013.

[4] D. McCandless. “The Internet of Things – A Primer.” informationisbeautiful.net. https://informationisbeautiful.net/visualizations/the-internet-of-things-a-primer

[5] A. Ali, W. Hamouda, and M. Uysal, “Next Generation M2M Cellular Networks: Challenges and Practical Considerations,” *IEEE Commun. Mag.*, vol. 53, Jun. 2015.

[6] D. Takahashi. “SoftBank believes 1 trillion connected devices will create $11 trillion in value by 2025.” venturebeat.com. https://venturebeat.com/business/softbank-believes-1-trillion-connected-devices-will-create-11-trillion-in-value-by-2025

[7] G. Kessler. “An Overview of Cryptography.” www.garykessler.net. https://www.garykessler.net/library/crypto.html

[8] Microsoft. “Enable TPM 2.0 on your PC.” support.microsoft.com. https://support.microsoft.com/en-us/windows/enable-tpm-2-0-on-your-pc-1fd5a332-360d-4f46-a1e7-ae6b0c90645c

[9] Arm. “Arm Confidential Compute Architecture.” www.arm.com. https://www.arm.com/en/architecture/security-features/arm-confidential-compute-architecture

[10] Arm. “PSA Certified Level 2.” www.psacertified.org. https://www.psacertified.org/getting-certified/silicon-vendor/overview/level-2

[11] *Security Requirements for Cryptographic Modules*, FIPS 140-2, May 2001. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.140-2.pdf

[12] CA State Legislature. 2017-2018 Session. (2018, Aug. 29). *SB-327, Information privacy: connected devices.* [Online] Available: https://leginfo.legislature.ca.gov/faces/billTextClient.xhtml?bill_id=201720180SB327

[13] Department for Science, Innovation and Technology. “Code of Practice for Consumer IoT Security.” www.gov.uk. https://www.gov.uk/government/publications/code-of-practice-for-consumer-iot-security

[14] GlobalPlatform. “TrustCB SESIP Scheme.” trustcb.com. https://trustcb.com/iot/sesip

[15] *Transitioning the Use of Cryptographic Algorithms and Key Lengths*, SP800-131A Rev. 2, Mar. 2019. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-131Ar2.pdf

[16] *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*, SP800-22 Rev. 1, Apr. 2010. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-22r1a.pdf

[17] Microsoft. “Secure boot and device encryption overview.” learn.microsoft.com. https://learn.microsoft.com/en-us/windows-hardware/drivers/bringup/secure-boot-and-device-encryption-overview

[18] M.Y. Wu *et al*. (2018). A PUF scheme using competing oxide rupture with bit error rate approaching zero. Presented at 2018 IEEE Int. Solid-State Circuits Conf. – (ISSCC*)*.

[19] PUFsecurity. “PUFrt.” www.pufsecurity.com. https://www.pufsecurity.com/products/pufrt/pufrt

[20] *Recommendation for Block Cipher Modes of Operation: Methods and Techniques*, SP800-38A, Dec. 2001. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a.pdf

[21] *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*, SP800-22 Rev. 1, Apr. 2010. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-22r1a.pdf

[22] *Recommendation for the Entropy Sources Used for Random Bit Generation*, SP800-90B, Jan. 2018. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90B.pdf

[23] *Recommendation for Random Number Generation Using Deterministic Random Bit Generators*, SP800-90A Rev. 1, Jun. 2015. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf

[24] Arm. “AMBA Specifications.” www.arm.com. https://www.arm.com/architecture/system-architectures/amba/amba-specifications

[25] PUFsecurity. “PUFcc.” www.pufsecurity.com. https://www.pufsecurity.com/products/pufcc (accessed Dec. 11, 2023).

[26] National Institute of Standards and Technology. “Cryptographic Algorithm Validation Program.” csrc.nist.gov. https://csrc.nist.gov/projects/cryptographic-algorithm-validation-program

[27] guanzi. “GM Standards.” github.com. https://github.com/guanzhi/GM-Standards/blob/master/README.md

[28] *Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping*, SP800-38F, Dec. 2012. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38F.pdf

[29] *Recommendation for Key Derivation Using Pseudorandom Functions*, SP800-108 Rev. 1, Aug. 2022. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-108r1.pdf

[30] *Secure Hash Standard (SHS)*, FIPS 180-4, Aug. 2015. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf

[31] L.B. Martinkauppi, Q. He, and D. Ilie. (2020). On the Design and Performance of Chinese OSCCA-approved Cryptographic Algorithms. Presented at 13th Int. Conf. on Commun. (COMM).

[32] *The Keyed-Hash Message Authentication Code (HMAC)*, FIPS 198-1, Jul. 2008. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.198-1.pdf

[33] *Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication*, SP800-38B, Oct. 2016. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38B.pdf

[34] *Recommendation for Block Cipher Modes of Operation: the XTS-AES Mode for Confidentiality on Storage Devices,* SP800-38E, Jan. 2010. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38e.pdf

[35] *ChaCha20 and Poly1305 for IETF Protocols*, RFC8439, Jan. 2020. [Online]. Available: https://datatracker.ietf.org/doc/rfc8439/

[36] *Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality*, SP800-38C, July 2007. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38c.pdf

[37] *Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*, SP800-38D, November 2007. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38d.pdf

[38] *Digital Signature Standard (DSS)*, FIPS 186-4, July 2013. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf

[39] *Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography*, SP800-56A Rev. 3, April 2018. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Ar3.pdf

[40] "AI Security White Paper," Huawei, Shenzhen, P.R. China, White Paper, 2018. [Online]. Available: https://www-file.huawei.com/-/media/corporate/pdf/trust-center/ai-security-whitepaper.pdf

[41] K. Sooksatra and P. Rivas. (2020). A Review of Machine Learning and Cryptography Applications. Presented at 2020 Int. Conf. on Comput. Sci. and Comput. Intell. (CSCI).

[42] IoT Analytics. "State of IoT 2021." iot-analytics.com. https://iot-analytics.com/number-connected-iot-devices

[43] R.R. Krishna *et al*, "State-of-the-Art Review on IoT Threats and Attacks: Taxonomy, Challenges and Solutions," *Sustainability*, vol. 13, no. 16, Aug. 2021.

Index

Our Thanks

We dedicate this book to our friends, families and, of course, the whole team at **PUFsecurity** that made this project possible.

Together, we gave our time and effort to develop this from only a nascent concept into a fully realized series of books about PUF based technology. This would have never happened without your continued support and commitment.

So, to each of you, we say thank you!

PUFsecurity Corporation.

8F-1, No. 5, Tai-Yuan 1st St., Jhubei City,
Hsinchu County,
302082, Taiwan
Tel: +886-3-560-1010
www.pufsecurity.com