BLIDA 1 UNIVERSITY

Faculty of Science

Computer Science Department

DOCTORAL THESIS

Specialty: Information Systems Security

A ROBUST AND SECURE AUTHENTICATION PROTOCOL FOR IoT SYSTEMS BASED ON PHYSICAL UNCLONABLE FUNCTIONS

By

Fahem ZERROUKI

In front of the Jury:

N. Boustia	Professor,	Blida 1 University	President
H. Bouarfa	Professor,	Blida 1 University	Reporter
S. Ouchani	A.Professor,	CESI, France	Co. Director
M. Bensalah	A.Professor,	EMP, Algeria	Examiner
F. Boumahdi	A.Professor,	Blida 1 University	Examiner

Blida, February 2023

DECLARATION

I here by declare that the thesis entitled “A Robust and secure Authentication protocol for IoT systems based on Physical Unclonable Functions” submitted by me, for the award of the degree of *Doctor of Computer science* to Blida 1 University is a record of bonafide work carried out by me under the supervision of Pr. BOUARFA Hafida and Dr OUCHANI Samir.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Blida, Algeria

Date: **Signature of the Candidate**

ABSTRACT

Internet of Things (IoT) is the interaction of a huge amount of devices (vehicles, buildings, etc.) integrating electronics, software, and sensors. They also support network connectivity that enables them to collect and share huge amounts of data. It has enormous potential for enhancing the quality of life, healthcare, manufacturing, and transportation, among other things. However, IoT devices must continuously manage and maintain a number of security challenges, including authentication, privacy, access control, and data gathering and management. Nevertheless, ensuring secure communication between these devices requires a robust authentication protocol that verifies identities and prevents malicious entities from accessing the trusted IoT network, as well as a secure session key for securing transmission following a successful authentication phase. Traditionally, the secret keys used by these devices as unique identifiers are inserted in the integrated circuits' non-volatile memory immediately after manufacturing. This renders them susceptible to numerous types of attacks, and it is costly, difficult, and sometimes impossible to resist these assaults using traditional cryptographic solutions. Traditional symmetric or asymmetric cryptographic algorithms, on the other hand, necessitate increased computing power, huge memory, and high energy sources for authentication and communication security. However, the limited memory capacity, processing power, and energy resources of IoT devices prevent the use of typical authentication procedures in IoT networks. In the last decade, Physical Unclonable Functions (PUF) have risen to prominence as a promising low-cost security primitive. A PUF eliminates the need to store secret keys in the device's memory, allowing it to replace more secure and less expensive authentication mechanisms for IoT systems. Consequently, IoT limitations and challenges lead to the introduction of lightweight authentication techniques that consider the unique characteristics and constraints of these devices. Therefore, any identity and authentication mechanism created for the IoT must be secure against physical threats, computationally efficient, and robust. Therefore, the purpose of this thesis is to design a strong and lightweight authentication mechanism for IoT-based systems using PUF by exploiting the characteristics of IoT chips. This thesis develops three innovative lightweight mutual authentication and key exchange protocols (MAKEP) employing PUFs for IoT devices. The first protocol, T2S-MAKEP,

ensures secure communication between Things and Servers. T2T-MAKEP enables two resource-constrained IoT devices to interact via an embedded PUF circuit, while LT2S-MAKEP is a lightweight version of T2S-MAKEP.

Keywords: *Physical Unclonable function*, *entropy*, *security metrics*, *performance*, *cryptography*, *haching function*, *authentication protocols*, *hardware security*, *attacks*.

ملخص

إنترنت الأشياء هي شبكة من الأشياء المادية التي تدمج المستشعرات والبرامج والتقنيات الأخرى من أجل الاتصال بالأجهزة والأنظمة الأخرى على الإنترنت وتبادل البيانات معها. توفر هذه التقنية نفوذًا هاما جدا لإنترنت الأشياء. هذه التكنولوجية تنمو وتعد بإمكانيات هائلة لتحسين نوعية الحياة في مجالات مختلفة مثل الصحة والصناعة والنقل وما إلى ذلك. لا يتغير ظهور إنترنت الأشياء كثيرًا أثناء استخدام نفس التكنولوجيا ونفس الاتصال ونفس تطبيقات الهاتف المحمول. ومع ذلك، يجب أن توفر أجهزة إنترنت الأشياء جمع البيانات وإدارتها مع ضمان أساسيات الأمن، بما في ذلك المصادقة، السرية، النزاهة والتحكم في الوصول وما إلى ذلك. إلا أن، ضمان الاتصال الآمن بين هذه الأجهزة يتطلب بروتوكول مصادقة قويًا يتحقق من الهويات ويمنع المهاجمين من الوصول إلى شبكة إنترنت الأشياء الموثوقة. أيضًا، يجب أن يوفر هذا البروتوكول مفتاح جلسة آمنًا لتأمين الإرسال بعد مرحلة مصادقة ناجحة. تقليديًا، تعتمد المفاتيح السرية التي تستخدمها هذه الأجهزة على حلول التشفير التقليدية. هذه المفاتيح المستخدمة كمعرف فريد يتم إدخالها في ذاكرة الدوائر المتكاملة فور تصنيعها، مما يجعلها عرضة للعديد من أنواع الهجمات وفي نفس الوقت تكون باهظة الثمن، صعبة وحتى مستحيلة المقاومة لهذه الهجمات باستخدام حلول التشفير التقليدية. في المقابل، تتطلب خوارزميات التشفير التقليدية المتماثلة أو غير المتماثلة قوة معالجة متزايدة، ذاكرة ضخمة ومصادر طاقة عالية للمصادقة وأمن الاتصالات. ومن جهة أخرى، فإن سعة الذاكرة المحدودة وقوة المعالجة وموارد الطاقة لأجهزة إنترنت الأشياء تعرقل استخدام إجراءات المصادقة النموذجية في شبكات إنترنت الأشياء. تعتبر الوظائف المادية غير القابلة للاستنساخ (PUFs) بديلاً مهمًا وواعدا للأمان وبتكلفة منخفضة. تلغي هذه الوظائف الحاجة إلى تخزين المفاتيح السرية في ذاكرة الجهاز، مما

يسمح له باستبدال آليات المصادقة بشكل فعال في أنظمة إنترنت الأشياء وبأقل تكلفة. أدت القيود والتحديات التي تمت مناقشتها في مجال إنترنت الأشياء إلى إدخال تقنيات مصادقة خفيفة الوزن تأخذ بعين الاعتبار الخصائص والقيود الفريدة لهذه الأجهزة. لذلك، أي آلية هوية ومصادقة لإنترنت الأشياء يجب أن يكون إنشاؤها مبني على ضمان الأمن ضد التهديدات المادية، فعالة من الناحية الحسابية وقوية. من أجل هذا، فإن الهدف من هذه الأطروحة هو تصميم آلية مصادقة قوية وخفيفة الوزن للأنظمة القائمة على إنترنت الأشياء باستخدام وظائف مادية غير قابلة للاستنساخ. تقترح هذه الرسالة ثلاثة بروتوكولات خفيفة الوزن للمصادقة المتبادلة وتبادل المفاتيح (MAKEP) لأجهزة إنترنت الأشياء باستخدام الوظائف المادية غير قابلة للاستنساخ. البروتوكول الأول، T2S-MAKEP ، يضمن الاتصال الآمن بين الأجهزة والخوادم. يسمح T2T-MAKEP لجهازين IoT المحدودة الموارد بالتفاعل، بينما LT2S-MAKEP هو Lightweight-T2S-MAKEP .

الكلمات الرئيسية: الوظيفة غير القابلة للنسخ الفعلية ، إنتروبيا ، مقاييس الأمان ، الأداء ، التشفير ، وظيفة التظليل ، بروتوكولات المصادقة ، أمان الأجهزة ، الهجمات .

RÉSUMÉ

L'IoT est le réseau d'objets physiques qui intègrent des capteurs, des softwares et d'autres technologies en vue de se connecter à d'autres terminaux et systèmes sur Internet et d'échanger des données avec eux. Cette technologie fournit l'effet de levier indispensable à l'IoT. Elle est en pleine croissance et promet un immense potentiel d'amélioration de la qualité de la vie dans différents domaines tels que, la santé, l'industrie, le transport, etc. L'essor de l'IoT ne change pas beaucoup tout en utilisant la même technologie, la même connectivité et les mêmes applications mobiles. Cependant, les appareils IoT doivent assurer la collecte et la gestion des données tout en garantissant les principes fondamentaux de la sécurité, notamment l'authentification, la confidentialité, l'intégrité, le contrôle d'accès, etc. Néanmoins, assurer une communication sécurisée entre ces appareils nécessite un protocole robuste qui vérifie les identités et empêche les personnes malveillantes d'accéder au réseau IoT de confiance. Aussi, ce protocole doit fournir une clé de session sécurisée pour sécuriser la transmission après une phase d'authentification réussie. Traditionnellement, les clés secrètes utilisées par ces dispositifs sont à base de solutions cryptographiques traditionnelles. Ces clés utilisées comme identifiant unique sont insérées dans la mémoire non volatile des circuits intégrés immédiatement après la fabrication, ce qui les rend vulnérables à de nombreux types d'attaques et dans le même temps, il est coûteux, difficile et voire impossible de résister à ces attaques à l'aide de solutions cryptographiques traditionnelles. En revanche, les algorithmes cryptographiques traditionnels symétriques ou asymétriques nécessitent une puissance de calcul accrue, une mémoire énorme et des sources d'énergie élevées pour la sécurité des communications. Cependant, la capacité de mémoire, la puissance de traitement et les ressources énergétiques limitées des appareils IoT empêchent l'utilisation de procédures d'authentification typiques dans les réseaux IoT. Les fonctions physiques non clonables (PUF) sont une alternative importante en tant que primitive de sécurité prometteuse à faible coût. Une PUF élimine le besoin de stocker des clés secrètes dans la mémoire de l'appareil, ce qui lui permet de remplacer efficacement les mécanismes d'authentification dans les systèmes IoT et avec le moindre coût. Les limites et les défis discutés de l'IoT conduisent à l'introduction de techniques d'authentification légères qui prennent en compte les caractéristiques et

les contraintes uniques de ces appareils. Par conséquent, tout mécanisme d'identité et d'authentification créé pour l'Internet des objets doit être sécurisé contre les menaces physiques, efficace sur le plan informatique et robuste. Par conséquent, l'objectif de cette thèse est de concevoir un mécanisme d'authentification fort et léger pour les systèmes basés sur l'IoT utilisant des fonctions physiques non clonables. Cette thèse propose trois protocoles légers innovants d'authentification mutuelle et d'échange de clés (MAKEP) utilisant des PUF pour les dispositifs IoT. Le premier protocole, T2S-MAKEP, assure une communication sécurisée entre les Objets et les Serveurs. T2T-MAKEP permet à deux appareils IoT à ressources limitées d'interagir via un circuit PUF intégré, tandis que LT2S-MAKEP est une léger version de T2S-MAKEP.

Mots clés: *Fonction physique non clonable*, *entropie*, *métriques de sécurité*, *performances*, *cryptographie*, *fonction de hachage*, *protocoles d'authentification*, *hardware security*, *attack*.

ACKNOWLEDGEMENT

First of all, I would like to express my sincere gratitude to my supervisory team, **Pr. BOUARFA Hafida**, research professor at the University of Blida 1 (Blida, Algeria) and **Dr. Samir Ouchani**, associate professor at CESI Engineering School (Aix-en-Provence, France). I am deeply grateful for their endless support, enormous efforts, and huge amounts of time devoted to this research throughout the last three years. The achieved academic record including this thesis, two high ranked articles, and five international conference papers, could not have been completed without their careful guidance, strict requirements, and constant encouragement. They consistently allowed this thesis to be my own work but set me on the right track. Their profound knowledge, strivings for perfection, insights into academic research, sincerity and integrity illuminated my way forward

I would be remiss in not mentioning the jury president Pr. Narhimene Boustia, and jury members Pr. Mustapha Bensalah, and Pr. Fatima Boumahdi.

I wish to extend my profound sense of gratitude to **my parents** for all the sacrifices they made during my research and also providing me with moral support and encouragement whenever required.

Last but not the least, I would like to thank **my wife** and **my daughter Hiba** for their constant encouragement and moral support along with patience and understanding.

Place: Blida

Date: February 21, 2023 **Fahem Zerrouki**

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

List of Terms and Abbreviations

Notation	Indication	Notation	Indication
PUF	Physical Unclonable Functions	IoT	Internet of Things
CRP	Challenge Response Pairs	FE	Fuzzy Extractor
SPUF	Silicon Physical Unclonable Functions	DoS	Denial-of-Service
AES	Advanced Encryption Standard	GPS	Global Positioning System
CNN	Convolutional Neural Network	IC	Integrated Circuit
HMAC	Hash Message Authentication Code	MITM	Man-In-The-Middle
IoMT	Internet of Medical Things	UAV	Unmanned aerial vehicles
MAC	Message Authentication Code	HD	Hamming Distance
SRAM	Static Random Access Memory	HW	Hamming Weight
NFC	near-field communication	LTE	Long Term Evolution
GSM	Global System Mobile communication	APUF	Arbiter PUF
CIA	Confidentiality, Integrity and Availability	M2M	machine-to-machine
RFID	Radio Frequency Identification	IoV	Internet of Vehicles
VANETs	Vehicular Ad-hoc Networks	CSA	Climate-Sensitive Agriculture
IoMT	Internet of Medical Things	CRC	Cyclic Redundancy Checks
VLSI	Very Large-Scale Integrated	SVMs	Support Vector Machines
ES	Evolution Strategies	LR	Logistic Regression
PAC	Probably Approximately Correct	ML	Machine Learning
SCA	Side-Channel Analysis	EM	Electromagnetic
TERO	Transient Effect Ring Oscillator	MUX	Multiplexer
FPGA	Field Programmable Gate Array	SG	smart grid
ECC	Elliptic Curve Cryptography	IP	Intellectual Property
PRNGs	Pseudo-Random Number Generators	T2T	Thing-to-Thing
HRNGs	Hardware Random Number Generators	T2S	Thing-to-Server

Notation	Indication	Notation	Indication
LT2S	Lightweight T2S	P2P	Peer-to-Peer
IP-PUF	Intellectual Property PUF	GPUF	Glitch PUF
CMOS	Complementary Metal Oxide Semiconductor	DD-PUF	Delay Difference PUF
CLK-PUF	Clock PUF	GS	Ground Station
RO PUF	Ring-Oscillator PUF	MPUF	Multi PUF
CMA-ES	Covariance Matrix Adaptation Evolution Strategy	PPUF	Pico PUF
MAKEP	Mutual Authentication and Key Exchange Protocols	5G	5^th Generation
TLS	Transport Layer Security	Vth	Threshold Voltage
CPU	Central Processing Unit	APUF	Arbiter PUF
WLAN	Wireless Local Area Network	IPUF	Interpose PUF
FF APUF	Feed-Forward Arbiter PUF	LSPUF	Lightweight secure pufs
DL	Deep Learning	DAPUF	Double Arbiter PUF
ICID	Integrated Circuit IDentification	RPPUF	Reconfigurable Pico PUF
Gen	Generation	Rep	Reproduction
CRO PUF	Configurable Ring Oscillator PUF	BPUF	Butterfly PUF
SNP	Static-Noise Margin	PC	Personal Computer

CHAPTER 1

GENERAL INTRODUCTION

This chapter presents the motivation behind the presented work, overall this Ph.D. thesis, the research method, as well as the goals and contributions of this dissertation thesis. Finally, this chapter describes the thesis structure.

1.1 Motivation

The use of smart technologies in our daily lives has made our lives smarter and more intelligent. Electronic smart devices try to make our lives easier by doing or helping us do things like route planning, navigation, transportation decisions, monitoring traffic and health, and keeping an eye on our children. [46]. For instance, a user can remotely connect to his refrigerator and check on food availability, and smart bracelets can provide us with our heart rate. The ”Internet of Things” (IoT) is a term for this type of connection between a user and a smart object or between smart devices [47].

IoT is a network of interconnected things that are capable of sensing, acting, and communicating with one another and with their environment (i.e., smart things or smart objects), as well as the ability to share information and act autonomously in response to real-world events by initiating processes and creating services with or without direct human intervention [48]. Computers, vehicles, smart phones, home appliances, toys, cameras, medical instruments and industrial systems, as well as animals, people, and buildings are all connected and capable of communicating and sharing information via predefined protocols [49].

Such devices are widely used in many applications, starting from smart homes to public health. Wireless technology is the most means of communication used by these devices to transfer a large amount of data, making them a prime target for cyber-attacks. Further, IoT devices face several security issues that must be continuously managed and maintained, including authentication, privacy, access control, and data collection and management [50]. However, ensuring secure communication between these devices requires a robust authentication protocol that refers to verifying identities and preventing malicious ones from accessing the trusted IoT network and a secure session key for securing transmission after a successful authentication phase. Unfortunately, any defect

in the authentication protocol allows an unauthorized thing to communicate, inject false data, get access to confidential data, and launch dangerous attacks with other things.

Traditionally, the secret keys, which are used by those devices as a unique identifier, are embedded immediately after manufacturing into the integrated circuits in non-volatile memory. This makes them vulnerable to many kinds of attacks such as invasive, semi-invasive, and side-channel attacks [3]. This allows an attacker to steal the secret key or to make a full copy of the device and use it in identity theft attacks. On another side, it is expensive, difficult, and maybe not possible to avoid these attacks with the classical cryptography systems that are based on the concept of a secret binary key.

From another side, traditional symmetric or asymmetric cryptographic algorithms require more processing power, large memory, and high energy sources to secure authentication and communication. Nevertheless, the limitations in memory capacity, processing power, and energy resources of the IoT devices impede deploying conventional authentication protocols in IoT networks [51].

Recently, a more attractive alternative has become a hot topic in research and development that relies on the physical disorder by giving birth to *Physical Unclonable Functions* (PUFs) [52].

A PUF is a one-way function that is derived from the behavior of a complex physical object. When a challenge (input) is presented to a PUF, a corresponding response (output) will be generated. The latter is determined by a complex physical function that is unique to each device and it is impossible to be duplicated because they have uncontrollable physical parameter variations that occur during hardware device manufacture. Nowadays, PUFs are widely used in identification and authentication. Due to the physical disorder of integrated circuits (ICs) caused during its fabrication by the manufacturing process. Silicon PUFs [53] are one of the most proposed and discussed PUF classes to generate a unique digital signature used as the fingerprint of an IC.

PUFs can generate unique secret information from the physical characteristics of the IoT device and use it as a unique device fingerprint, making PUF a very efficient solution for IoT authentication protocol. Silicon PUF eliminate the need to store secret keys in device memory, making them a potential alternative to deploying more secure and low-cost authentication protocols for IoT systems.

The PUF-based authentication mechanism is a very efficient solution for IoT authentication protocol since it eliminates the need for storing secret keys in the IoT memory. Based on the randomness found in the physical characteristics of the physical things, PUFs extract unique secret information and use it as the IoT device identity, as a unique device fingerprint. Therefore, the latter can be used for IoT device authentication. Also, it is low-cost security primitive, which makes it robust against physical attacks [51] and suitable for the constraint of IoT appliances.

1.2 Problem Statement

The existing security techniques are not longer sufficient for current IoT systems, especially for IoT constraint devices, as it is challenging to deploy conventional authentication protocols since they require more processing power, large memory, and high energy sources [50, 3]. Further, IoT devices operate in open and public places without being physically protected or guarded, which makes them vulnerable to physical attacks [51]. Despite all of the benefits of IoT, their applications face many security requirements that must be assured, including confidentiality, integrity, and authentication for users, devices, and data. In particular, authentication is the foremost essential and indispensable security mechanism that can guarantee secure communication between the entities of an IoT network [44]. The discussed limitations and issues related to the IoT led to the emergence of lightweight authentication schemes by considering the specific nature and constraints of these devices. Therefore, any identification and authentication protocol designed for the IoT needs to be robust, computationally efficient, and secure against physical attacks.

Therefore, the goal of this thesis is to use physical unclonable function to design a robust and lightweight authentication protocol suitable for IoT based system. This solution needs to work in a resource-constrained environment, and the system needs to be end-to-end secure.

1.3 Thesis Goals

The goals of this dissertation thesis are:

- Learn and comprehend the concepts of PUF and IoT.
- Identifying the security issues, and more precisely, the authentication in IoT systems.
- Reviewing the recent contributions dedicated to IoT PUF-based authentication protocols and comparing and judging the reviewed state-of-the-arts.
- Proposition of a PUF-based authentication protocol for IoT systems.
- Validation of the proposed schemes using formal and informal security analysis.
- Validation of the proposed schemes through experience and simulation.

1.4 Research Method

A detailed strategy of steps has been followed in this thesis to achieve the goal: 1) understanding the required background, 2) studying the literature, 3) developing a theoret-

ical framework for IoT protocols, 4) analyzing security, and 5) simulating the proposed schemes.

1.4.1 Needed Background

The first task was to study the general area of interest. Which is Physical Unclonable Function (PUF) and the Internet of Things (IoT), a starting point was investigated, and analyses of the concept of PUF were conducted by collecting all the needed background to fully understand its technology in terms of principles of application, architecture, and vulnerabilities. It was discovered that the PUF technology is a popular security primitive for IoT systems. In a second time, we have searched and collected the needed articles related to IoT. This is to study the IoT technology applications, architecture, and challenges. Authentication was found to be one of the biggest open issues in terms of security with IoT.

1.4.2 Literature Review

After receiving the needed materials and information related to our topic, the second task of this project was reading literature on internet of things-based authentication protocols, where it was discovered that PUF was one of the newest solutions found in the literature used as an IoT security primitive. Based on these results, a deep survey related to IoT PUF based authentication protocols was achieved, and the most recent schemas and approaches were explored and analysed. From this analysis, gaps were identified.

1.4.3 Theoretical Work

Following the literature review, an IoT PUF-based authentication protocol was proposed. Through the threat analysis and practicality of the existing work, it is recognized that a given PUF-based authentication protocol for the IoT must withstand and avoid known attacks, particularly physical and modeling attacks. From another point of view, most of the existing PUF-based protocols do not take into consideration the environment variation when using PUF rather than most of the thing-to-thing authentication protocols are impractical in a real situation. At the conclusion of this stage, three novel protocols were proposed: 1) thing-to-server (T2S) authentication; 2) thing-to-thing authentication; and 3) store-less authentication.

1.4.4 Security Analysis

Upon completion of the design phase, the goal of this phase of our research was to evaluate the security and performance of the proposed protocols. Two methodologies are

used to conduct the security analysis: informal analysis and formal verification. Analyzing the protocols against the security requirements stated for each set of protocols was the objective of the first step. The formal verification was then used to test the protocols' correctness and offer systematic verification. Verifpal [45] was used to do the formal validation. Once the security analysis has been performed, the protocols' performance will be evaluated. In addition to protocol execution time, computational and communication overheads are considered while evaluating protocols.

1.4.5 Simulation Modelling

The next task was to evaluate the performance of the proposed protocols. The first step was to implement an Arbiter PUF design using the PyPuf simulator, which generated the necessary data, which was then evaluated in PUF performance metrics. Following that, the Fuzzy Extractor Technique is used to eliminate noise in the PUF response and generate a stable, reproducible response. Finally, to demonstrate the efficacy of the new authentication schemes, the performance evaluation results are compared to the most pertinent literature.

1.5 Scientific contributions

The contributions achieved during this dissertation thesis are presented in the following list:

- **Conference papers**

  1. Fahem Zerrouki, Samir Ouchani, Hafida Bouarfa. "Quantifying security and performance of physical unclonable functions".IEEE 7th International Conference on Internet of Things: Systems, Management and Security (IOTSMS 2020). Paris, France.
  2. Fahem Zerrouki, Samir Ouchani, Hafida Bouarfa. "Towards an automatic evaluation of the performance of physical unclonable functions". Springer, the fourth International Conference in Artificial Intelligence in Renewable Energetic Systems (ICAIRES 2020). Tipaza, Algeria.
  3. Fahem Zerrouki, Samir Ouchani, Hafida Bouarfa. "Towards a Foundation of a Mutual Authentication Protocol for a Robust and Resilient PUF-Based Communication Network". Elsevier, the 18th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2021). Leuven, Belgium.
  4. Fahem Zerrouki, Samir Ouchani, Hafida Bouarfa. "A Low-Cost Authentication Protocol Using Arbiter-PUF Springer". Springer, the 10th International Conference on Model and Data Engineering (MEDI 2021). Tallinn, Estonia.

5. Fahem Zerrouki, Samir Ouchani, Hafida Bouarfa. “Generation and Recovery Framework for Silicon PUFs based Cryptographic Key”, Springer, the 10th International Model and Data Engineering (MEDI 2021) Workshops, Symposium on Intelligent and Autonomous Systems (SIAS 2021). Tallinn, Estonia.

- **Journal papers**

1. ZERROUKI, Fahem, OUCHANI, Samir, et BOUARFA, Hafida. A survey on silicon PUFs. Journal of Systems Architecture, 2022, vol. 127, p. 102514.
2. Zerrouki Fahem, Samir Ouchani, and Hafida Bouarfa. ”PUF-based mutual authentication and session key establishment protocol for IoT devices.” Journal of Ambient Intelligence and Humanized Computing (2022): 1-19.

- **Paper under review:**

1. “Physical Unclonable Function-based Authentication Protocols for Internet of Things: A Review”. Software and Systems Modeling, Impact factor=2.211, Classe=Q1.
2. “T2T-MAKEP: A PUF-Based Thing-to-Thing Mutual Authentication and Key Exchange Protocol for IoT devices”. IEEE Internet of Things Journal, Impact factor=10.238, Classe=Q1.
3. “A Robust and Low-Cost Authentication Protocol for a Secure IoT Communication”. International Journal of Information Security, Impact factor=2.427, Classe=Q2.

1.6 Thesis Outline

This thesis will be organized as follows:

- Chapter 2 will be the **concepts and background**, which will include all the important concepts that are related to our work, especially IoT systems and PUFs, as they are the main parts of our thesis.
- Chapter 3 will be the **stat of the art**, where we will present, discuss and compare the existing IoT authentication techniques and from another side, the existing PUF-based authentication protocol schemes.
- Chapter 4 will be the **contributions**, which is the most important part of this thesis because it is where we will present and outline the proposed protocols.

- Chapter 5 will be the **verification and analysis**, which is dedicated for approving and studying the robustness of the proposed PUF-based authentication protocols for IoT systems using formal and informal verification tools.
- Chapter 6 will be **Implementation, experimentation and simulation**, which is dedicated for approving the proposed authentication protocol through simulation tools.
- Chapter 7 will be **General Conclusion and Future Work**, where we highlight the key conclusions and contributions of our study. We also provide some prospective research directions.

CHAPTER 2

CONCEPTS AND BACKGROUND

In recent years, the Internet of Things has become one of the most well-known names to achieve new heights and establish a global benchmark (IoT). Things in the real world have become intelligent due to the future of communication. IoT's functional element is to connect every object in the world to a single infrastructure, allowing humans not only to manage those objects but also to get regular and timely status reports. IoT principles were introduced a few years ago, and it would not be incorrect to state that this phrase has become the standard for creating communication among items. However, specific security measures should be taken to protect the communicated information to and from these devices. However, the existing conventional security primitives require large amounts of memory capacity, processing power, and energy resources that contradict the specific nature of devices. On the other hand, they store secret keys on the devices for future use, making them vulnerable to physical attacks. A new concept, known as Physically Unclonable Functions (PUFs), has been recently investigated to mitigate this problem. A PUF is a hardware-specific security primitive uses the randomness found in the disorder of physical media caused by the manufacturing variation process to provide cryptographic functionalities. Consequently, PUFs are inexpensive to fabricate, prohibitively challenging to duplicate, admit no compact mathematical representation, and are intrinsically tamper-resistant. The main focuses of this chapter are:

1. Introducing the background and concepts related to the Internet of Things.
2. Identifying the main ongoing open problems that IoT is facing.
3. Providing the necessary context and knowledge to understand PUFs and their applications.
4. Showing how to analyze the performance of PUF.
5. Categorizing the existing attacks proper to PUFs.

2.1 Internet of things

In 1999, Kevin Ashton coined the term ”Internet of Things” to refer to the network that connects physical objects to the Internet. But, the concept of the Internet of Things dates back to the 1980s, when David Nichols, a graduate student in the computer science department at Carnegie Mellon University, craved a soda in his office, which was a considerable distance from the building's coke machine, and given his classmates' propensity for caffeine, Nichols knew there was a good chance it was empty or, if recently refilled, the sodas inside would be tragically hot. From this concept, a group of programmers could connect to the Coca-Cola machine via the Internet to check its status and determine whether a cold drink was present or not if they went to the machine [54].

The Internet of Things (IoT) typically refers to a world of networked smart items, in which any physical ”thing” with a digital component is interconnected. According to [47] IoT is defined as follows: *A dynamic global network infrastructure with self-configuring capabilities based on standard and interoperable communication protocols where physical and virtual ”things” have identities, physical attributes, and virtual personalities, uses intelligent interfaces, and is seamlessly integrated into the information network.* IoT enables people to have complete and intelligent control over their life by granting them access to their data from anywhere, at any time, and on any device, and by enhancing the quality of their businesses' services. As an example, a home automation system employs IoT to monitor, regulate, and automate the interconnected electrical systems in a building. Smartcities provide limitless opportunities for residents to cut waste and energy usage. IoT has numerous uses in today's world, such as in healthcare, banking, retail, society, the environment, and manufacturing, among others. The this first part of this chapter, we provides the necessary background to understand Internet of things technology including characteristics, application, architecture and open challenges.

2.1.1 Characteristics

Fig. 2.1 shows the fundamental characteristics of the IoT [55, 49] :

- **Connectivity:** It is one of the most important and standard features of the IoT. With the global information and communication infrastructure, anything can be linked to everything else. It helps keep things accessible by connecting the objects to each other through a network. Also, it helps with compatibility, which is the ability to send, receive, and make data without any problems or conflicts [49].
- **Heterogeneity:** IoT devices are different from each other because they use different hardware platforms and networks and can talk to other devices on different

networks. The lack of a single security service is the biggest problem with heterogeneity. Heterogeneity makes it harder for systems to work together and costs more money and time to interpret each other. Also, it makes security policies and updates challenging to comprehend [49].

- **Sensing and Intelligence:** The IoT environment is mostly based on sensing technologies that detect, measure, and generate status data about our complex physical world. These technologies give us a true understanding of it or even let us interact with it in a smart way, thanks to algorithms and computations that make it smart [55].

- **Resources Constraint:** The majority of IoT devices lack performance and battery life. Consequently, older security services such as TLS and AES cannot be implemented directly to IoT devices. Therefore, these services or algorithms should be built to be lightweight and simple in order to maximize CPU, memory, and battery performance [55].

- **Dynamic Environment:** ToT environments are pretty dynamic. At any time, devices can be roused, terminated, linked, or disconnected from a network. Devices, software, and networks are susceptible to malfunction or compromise. In IoT contexts, device volatilities are extremely prevalent [49].

- **Enormous scale:** The number of devices that must be managed and communicate with one another will be at least an order of magnitude greater than the number of devices linked to the Internet today. The management of created data and its interpretation for application purposes will be even more crucial. This relates to the semantics of data as well as the efficient management of data [55, 49].

- **Self-organized Network:** The dynamism of the IoT network implies that the network cannot be statistically organized for its whole life period. It has to be able to adapt to the constant changes in its environment. It has to think about how mobile the devices are and how likely it is that new devices will connect or old devices will leave the network [55].

- **Safety:** As we obtain benefits from the Internet of Things, we must not neglect safety. We must design for safety. As both creators and recipients of the IoT, this involves the protection of our personal information and physical health. Developing a scalable security paradigm is necessary for securing endpoints, networks, and the data that flows across them all [49].

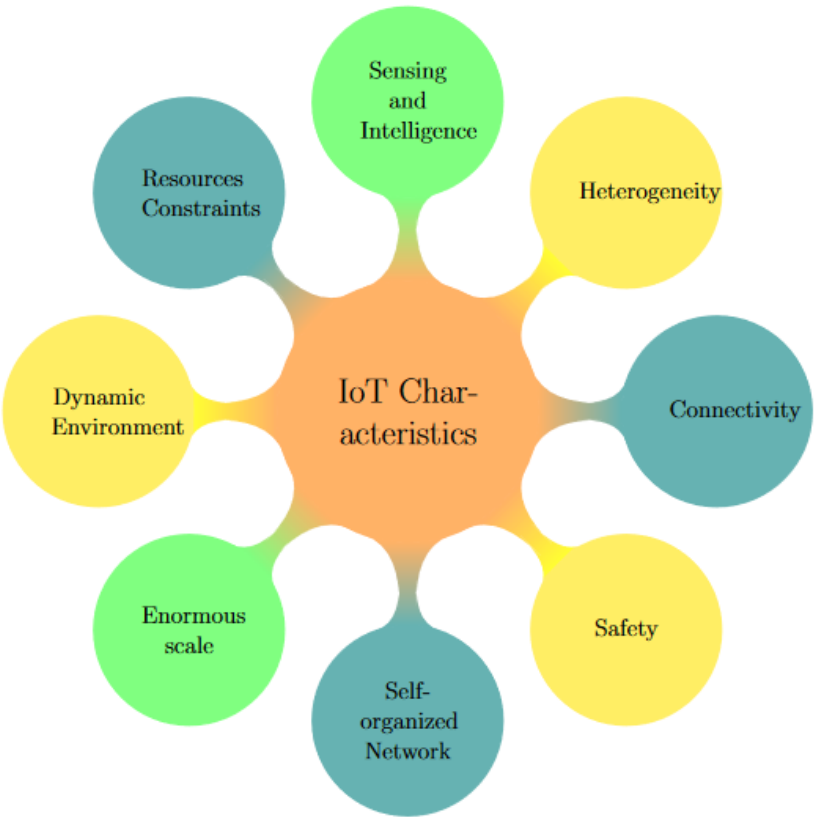

Fig. 2.1 IoT Characteristics.

2.1.2 Architectures

Due to its constant growth and development, the Internet of Things needs a universal architecture that can be changed to fit its many different types of devices and applications. At the moment, there is no architecture that everyone uses. Several researchers have come up with different IoT architectures based on three-layer, four-layer, five-layer, six-layer, and seven-layer architectures [56]. As illustrated in Fig. 2.2 The three-layered architecture shows the main idea behind the Internet of Things, which is broken up into three basic layers and their functions. Next, the layers are shown and talked about according to the definition given in [47].

1. **Application layer:** A variety of intelligent IoT application solutions make up this layer. Due to the immense potential of the IoT market, smart applications are being developed in practically every area. Numerous Internet of Things (IoT) applications have already been implemented in a number of industries, including smart homes, offices, and cities, as well as wearable bands for monitoring one's health, traffic, environment, alarm system, and personal assistant. The IoT application layer, which acts as an interface between networks and objects, is the top layer in the IoT architecture. It provides a range of functionalities, including data formation, presentation, monitoring of device conditions, notifications, and alerts,

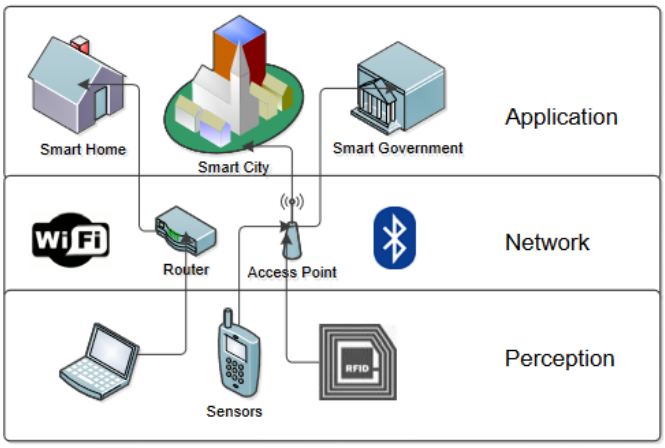

Fig. 2.2 IoT three layer architecture [1].

controlling device functions, management and processing of data, device performance optimization, and autonomous operations, all of which are aimed at ensuring end users receive a high level of service. A service support platform, middleware, and computer and communication software are common components of an application layer. The IoT application layer's primary objective is to offer various application services to end users. By protecting applications from illegal access, assuring software/log integrity, and maintaining the application services' availability at all times, data confidentiality, integrity, and availability (CIA) should be maintained at this layer. Sensitive data processing can lead to problems like unauthorized access and harmful data manipulation. Additionally, this layer may be susceptible to various security assaults, such as spoofing, message forging, viruses, and worms.

2. **Network layer:** Software, protocols, and technologies that enable object-to-object and object-to-internet connectivity make up this IoT layer. Local area networks such wireless and wired networks, personal area networks like ZigBee, near-field communication (NFC), and Bluetooth, as well as wide area networks like GSM, LTE, and 5G, as well as cloud computing, are mostly used to create it. The M2M communications, machine-to-gateway model, machine-to-cloud model, and a back-end data-sharing model have been listed as versions of the IoT communication paradigm. This layer's primary mission is to send digital signals made up of data that has been acquired from platforms' physical layers via a network connection. This layer is exposed to several security risks and assaults. Denial-of-Service (DoS), Sinkhole, Hello Flood, and Blackhole are frequent assaults in this layer. Secure communication at the network layer is crucial for secure data transfer over a public network.

3. **Physical layer:** The physical layer is the core of the Internet of Things architec-

ture. This layer is also known as the perception layer in the Internet of Things. Both real-world items and virtual beings are included. This layer's primary duty is to gather environmental data using a variety of sensors. Electronic and mechanical hardware elements including sensors, antennas, actuators, and CPUs are embedded in IoT devices. Data processing, identification, connectivity, communication, and storage capabilities are available in smartphones, RFID technology, and wearable gadgets. The sensors at the perception layer translate the physical object data that has been collected into readable digital signals. IoT devices perceive and collect information from the physical world, including proximity, humidity, and temperature. But many security attacks, such jamming and tampering attacks, can target this layer of the IoT.

2.1.3 Applications

Through adaptation, IoT has a lot of potential to have positive social, environmental, and economic effects. Some of the IoT-based concepts include mobility, smart grid, smart homes and buildings, public safety and environment monitoring, healthcare and medicine, industrial processing, agriculture and breeding, and independent living [2]. In some way or another, each of these applications relates to us. The use of these apps and their striking advantages play a significant role, and there is now a great deal of reliance on their continued existence. Their existence and usability have recently reached a visionary level and have taken the utmost significance. It may not be erroneous to say that the Internet's future is solely built on the idea and vision of the Internet of Things, which virtually propels us into the future [57]. Fig. 2.3 illustrates a number of IoT application areas, that are presented, following [2, 57], as follow:

1. **Smart Mobility:** Smart Mobility is the methodology that enables efficient, flexible, and seamless mobility across many modalities. VANETs (Vehicular Ad-hoc Networks) have attracted considerable interest. Thus, it represents a paradigm change towards a more flexible and multimodal transportation system. It is a pillar of the Internet of Vehicles (IoV) that aims to improve road safety by preventing or minimizing accidents and giving new solutions for optimized forms of mobility. The Smart Traffic System enhances traffic flow based on traffic data collected by IoT-enabled devices. Intelligent traffic management systems necessitate vehicle identification and traffic factor monitoring. By providing all information regarding congestion, such as the shortest route and least congested area, intelligent traffic monitoring improves the travel experience. In addition, it provides road toll collection, traffic accident reporting, vehicle theft detection, and reduced pollution .

2. **Smart Grid:** To improve the unwavering quality, reduce the cost, and simplify

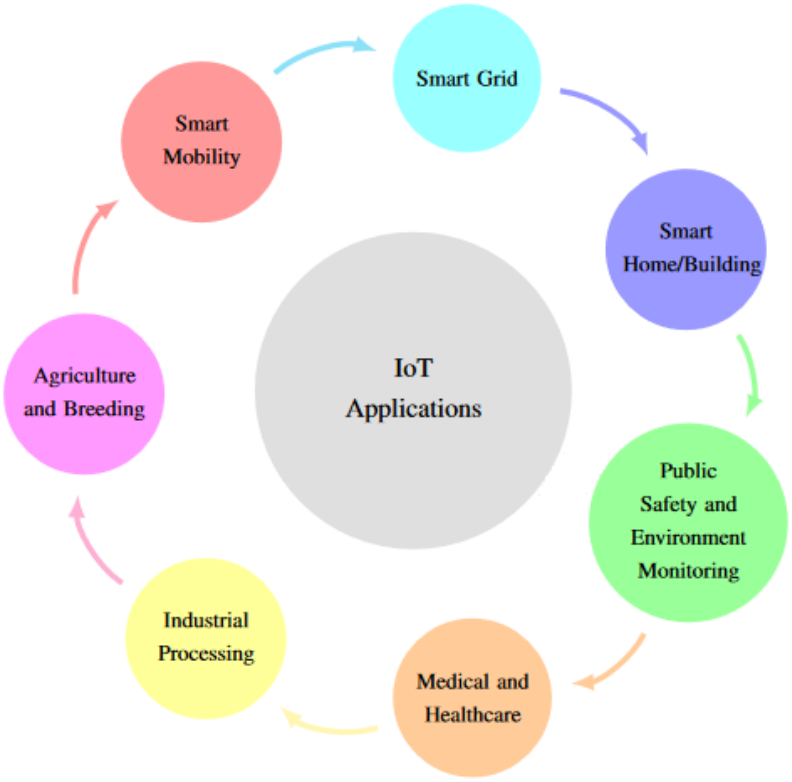

Fig. 2.3 IoT application [2].

the implementation of the traditional power matrix methodology, the Smart Grid has been planned and implemented. It is anticipated that it will be able to efficiently increase the quality and power matrix while incorporating increasingly green and sustainable energy sources including solar power, geothermal heat, and wind power. It gathers information on energy use and shows the state of the smart network system. Depending on the specific knowledge and communication structures of the framework, different applications can be created. The automated metering framework is one of the techniques that have been suggested to produce the clever matrix correspondence systems. Security is also one of the most crucial issues in the development of these frameworks because of the vast volume of information that passes through this particular structure of the project. Attacks on integrity and the introduction of fake data will annoy the heavy layer charging structure, the matrix state estimation, the power stream, and delay reaction requests.

3. **Smart Home/Building:** A Smart Home or Smart Building is an environment outfitted with heating, lighting, and other technological gadgets, unlike any other living environment. The fact that they can be remotely operated by a smart phone or a computer is a key distinction. In recent years, the notion of ”smart houses” and ”smart buildings” has emerged by integrating Internet-connected items, such

as machinery, door locks, surveillance cameras, furniture, and carport entrances, and managing them by communicating with existing digital frameworks. This advancement in understanding gives a variety of advantages for improved human comfort, well-being, safety, and productive use of defining assets, hence enhancing the quality of life.

4. **Public Safety and Environment Monitoring:** Public safety and environmental monitoring is the process of observing weather conditions, endangered species protection, water quality monitoring, and a variety of other parameters directly or indirectly associated with our environment. Various sensors and other observational tools are integrated into applications to monitor environmental changes in real time. IoT technologies aid in weather forecasting and the prediction of natural disasters, such as earthquakes, floods, incessant rains, cloud bursting, thunderstorms, high speed winds, tides, volcanic eruptions, tornadoes, hail storms, hurricanes, tsunamis, forest fires, and fires, among others. The IoT plays a significant role in predicting the occurrence of natural disasters. This facilitates the timely evacuation of affected areas. Humans are rescued and relocated to safe locations. The Internet of Things facilitates the monitoring of air, water, and soil pollution in a smart environment.

5. **Medical and Healthcare:** Often referred to as the Internet of Medical Things (IoMT), this application mode connects healthcare services to the IT system via multiple computer networks. RFID tags are embedded in RFID-enabled, wearable, Internet of Things (IoT)-enabled devices found in smart hospitals. These wearables are given to hospitalized patients upon their arrival. The doctor and nurses will be able to monitor the patient's health by taking note of blood pressure, heart rate, temperature, and pulse, among other conditions. They can be monitored both on hospital grounds and from the doctor's home. In emergency situations such as a stroke, heart attack, or cardiac arrest, IoT is incredibly useful. Ambulances equipped with IoT-enabled devices arrive on time, and the patient can be monitored from the hospital while in the ambulance. Prior to reaching the hospital, treatment can begin considerably earlier. Drone ambulances can transport emergency supplies to the patient, allowing for proper patient monitoring. Doctors are able to track patients and administer prompt medical care until an ambulance arrives. IoT is incredibly useful for differently abled, physically or mentally challenged individuals. The Internet of Things enhances the quality of life by automating mundane tasks. The machines are capable of monitoring and making decisions. The IoT is extremely useful in healthcare. Sensors on health monitoring equipment collect medical information from patients and transmit it to physicians.

6. **Industrial Processing:** In recent years, the IoT concept has also flourished in the industrial sector. Modern industrial equipment and requirements are so demanding that the IoT's functional capabilities are either molded or designed to meet the industry's needs. Utilizing recognizable evidence and remote, flexible, and sensor devices to establish the universality of radio-recurrence, IoT has created the potential for the accumulation of incredible current industrial systems and applications. In recent years, a vast selection of mechanical IoT applications have been developed. In general, the combination of sensors/actuators, RFID labels, and communication constitutes the establishment of the Internet of Things (IoT) and clarifies how a variety of physical objects and devices in our environment can be connected to the Internet and enables such objects and devices to communicate with one another in order to achieve common goals. There is a growing enthusiasm for using the advances of IoT in various businesses. Different mechanical IoT undertakings have been carried out in territories, such as agriculture, food preparation, ecological observation, security observation, and others.

7. **Agriculture and Breeding:** Climate-Sensitive Agriculture (CSA) is a method to reforming and reorienting agricultural production in light of climate change's practical implications. There have been substantial developments in the techniques and methods used for agricultural activity. Modern farmers have transitioned from conceptual farming to modernized notions. Researchers in this field have developed theories and techniques that include smart devices to analyze the characteristics that contribute to the growth of plants, and agricultural activities are conducted based on these observations. Automatic temperature management and other parameters for optimizing production would improve agriculture. It will monitor soil nutrition, sunshine, and humidity, as well as assure proper watering and accurate fertilizer application to increase crop yield. This translates in water and fertilizer savings. Certainly, IoT improves the quality of life for farmers. The sensor nodes are installed in the ground. Exact soil and agricultural conditions data and information are collected and analyzed. This analysis facilitates effective agricultural planning. They can maintain the health of their crops by the application of fertilizers and insecticides. Crops will be free of illnesses, weeds, insects, and other pests if the correct amount of hydration and the right quality and quantity of pesticides are utilized. It affects the farmer's life, as he can earn substantial rewards from his produce. Utilizing innovative technologies, intelligent agriculture improves the quality and quantity of agricultural products.

2.1.4 Open Issues

As shown in Fig. 2.4 There are various issues and challenges with IoT that need to be resolved to ensure wide adoption of IoT. In a smart home, the main door opens using face recognition, biometric thumb impression, and voice recognition security features. The heaters and air conditioners monitor and regulate the temperature automatically. The air controller monitors the humidity and air and regulates it automatically as per the climatic conditions and as required by users. There is always a risk of security as all these systems are connected through the Internet and can easily be hacked. Security is a major concern for unmanned IoT devices, which are geographically distributed. The sensors consume more energy and are costly, so they are not widely used in all applications. Various networks and systems are not upgradeable and scalable. This results in compatibility issues in interoperability and data exchange. Dedicated resources are required for a single IoT application like smart grids or smart homes. IoT can mend the usage of the Internet the way it is required by its end-users. This feature also has to face some of the open issues that are directly or indirectly associated with the concept of IoT. The most common of them [58] have been discussed as under:

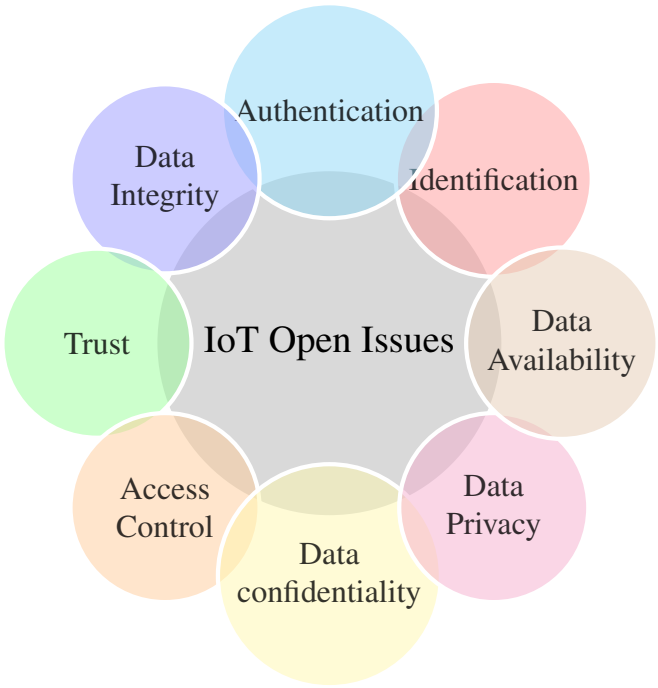

Fig. 2.4 IoT Open Issues.

1. **Identification:** It is crucial for a smart gadget to know when to expose its identify and when not to. Disclosing your identify to an attacker can pose a grave risk. Nonetheless, we must acquire a system that simultaneously gives device identification to other authorized devices. Devices that interact with individuals (users) must recognize their identities and distinguish amongst them.

2. **Authentication:** Authentication is challenging because it often necessitates a suitable authentication infrastructure and servers that accomplish their aim by exchanging the necessary messages with other nodes. Such strategies are not viable in the IoT because passive RFID tags cannot exchange too many messages with authentication servers. The identical logic applies to the sensor nodes.

3. **Data Integrity:** Cybercriminals are susceptible to a range of external circumstances, such as data modifications during the transition, server outages, and electromagnetic interference. Data integrity is the application of standard surveillance techniques to secure this valuable data from cybercriminals and prevent external intervention during transmission and reception. Consequently, the system cannot alter the data without first identifying the threat. As error detection mechanisms, checksums and cyclic redundancy checks (CRC) are employed to ensure that data is accurate and trustworthy.

4. **Trust:** Trust is a complex, transdisciplinary, and multifaceted notion. Trust encompasses a broader scope than security, making its establishment more complex and challenging. It is also related to the notion of privacy, which refers to an entity's ability to select whether, when, and to whom personal information may be shared. Numerous studies strive to enhance identity trust and achieve privacy protection in pervasive systems such as the Internet of Things. Many individuals believe that users will adopt and utilize IoT technology since manufacturers have ensured the safety of the gadgets.

5. **Data confidentiality:** Protects the user and ensures that confidential information can be relied upon by employing a number of techniques to prevent unauthorized disclosure. Data encryption, which prevents data from unwanted access, two-stage authentication, which provides authentication by two dependent components, and biometric authentication, each of which is uniquely recognized, are security measures that secure data privacy. This assures that, for IoT-based devices, sensor networks do not reveal the data of sensor nodes to nearby nodes or transfer label data to an unauthorized reader.

6. **Access Control:** Access control refers to the permissions granted to various actors in a big IoT network to utilize resources. Access control should prioritize IoT capabilities above per-device granularity, as context-dependent factors substantially influence access control decisions. Several steps must be completed in order to give the smart device with the required access control requirements and authentication.

7. **Data Privacy:** Due to the proliferation of larger volumes of data in an IoT context, addressing the risk that data will be used for purposes other than or in addi-

tion to those initially specified becomes even more crucial. As users walk across IoT environments, devices, sensors, readers, and applications are equipped to capture a variety of data kinds. There is a chance that individuals could be identified using the aggregated data. The information obtained based on object identifiers, sensor data, and the connectivity capabilities of IoT systems may consequently reveal information about individuals, their habits, location, interests, and other personal information and preferences maintained for system convenience.

8. **Data Availability:** When required, IoT provides data to its users. The availability of data enables the authorized individual to have immediate access to information sources in both normal and catastrophic circumstances. Implementing a firewall avoids denial of service (DoS) attacks and prevents end user data from being accessible. Multiple system failovers are provided by backups and backup methods to provide system component replication in the case of a system failure or to assure data integrity and availability.

2.1.5 Summary

In this section, we have defined the IoT concept and its expansion in our lives by presenting its characteristics and applications, and we have also introduced the basic architecture. In addition, we have listed the existing open issues related to IoT security, where we found that authentication is one of the most important security issues facing IoT. In the next section, we present physical unclonable functions that represent a fundamental part of our project.

2.2 Physical Unclonable Functions

Integrated Circuits (ICs) and electronic devices have become an integral part of daily human life (mobile, home, car, etc.). However, specific security measures should be taken to protect the communicated information to and from these devices. However, the existing conventional security primitives require large amounts of memory capacity, processing power, and energy resources that contradict the specific nature of devices. On the other hand, they store secret keys on the devices for future use, making them vulnerable to physical attacks. A new concept, known as Physically Unclonable Functions (PUFs), has been recently investigated to mitigate this problem. A PUF is a hardware-specific security primitive that uses the randomness found in the disorder of physical media caused by the manufacturing variation process to provide cryptographic functionalities. Consequently, PUFs are inexpensive to fabricate, prohibitively challenging to duplicate, admit no compact mathematical representation, and are intrinsically tamper-resistant. This section gives the needed background to understand

PUF's ideas by talking about the concepts of randomness and variability, as well as their classes and properties. Then, the principal metrics used to evaluate the PUFs' performance and present some related attacks were given. Finally, a short summary of all the presented background was given.

2.2.1 PUFs background

Some of the most common terms and measurements that describe PUFs are shown in this section. They help us understand PUFs and how they work.

2.2.1.1 Physical disorder

Physical disorder refers to the random imperfections found in the structure of physical objects. This phenomenon is typically observed at the nano-scale level of the physical objects' structures. Many fascinating randomnesses exist around us, taking various forms such as biological, physical, chemical entities, and so on, caused by nature or any manufacturing process [3].

As a naturally physical disorder example, the surface with three-dimensional random structures of a coffee bean is presented as a microscopic image as shown in Fig. 2.5. (a). Fig. 2.5. (b) represents the microscopic image of a biological physical disorder example of a human tooth. Fig. 2.5. (c) shows the irregular structure of the metal conductors in a semi-conductor chip fabricated using 90 nm technology.

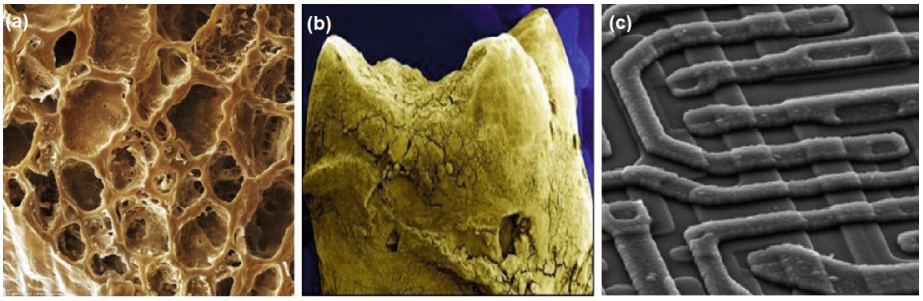

Fig. 2.5 Examples of physical disorder: (a) a coffee bean, (b) a tooth, and (c) an integrated circuits [3].

This physical disorder is unique to each object and is hard or impossible to replicate, and it can be used as an identity for this object or the device embedded in it.

2.2.1.2 Manufacturing Variation

As a main principle, the manufacturing process of any product should be identical in its shape and structure to the needed product design. However, this is not the case in most modern chips and integrated circuits due to the manufacturing process and continuous scaling of semiconductor technologies [3].

The manufacturing process variability is affected mainly by four factors: physical geometric structure, internal material parameters, interconnect geometry, and interconnect material structures [3]. The impact of variability on the electrical parameters of very large-scale integrated (VLSI) circuits is expected to be significant. Fig. 2.6 shows the magnitude of variation in device threshold voltage (Vth) and the performance of VLSI circuits. We observe that the impact of variations on threshold voltage increases significantly compared to the performance evaluation of the VLSI circuit, which endangers the stability of the circuit operations. However, these variations can be exploited to design a physically unclonable function.

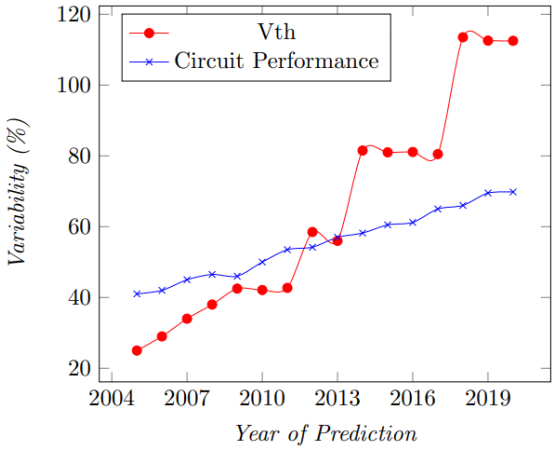

Fig. 2.6 The impact of variability on the electrical parameters of VLSI circuits [3].

2.2.1.3 Challenges and Responses Pair (CRP)

Challenges are entries given as inputs to an instance of a PUF. When a challenge stimulates a device where an instance of a PUF is embedded, the latter will interpret it in its internal system using the complex physical function unique to each device or PUF instance. Then, the PUF will produce unpredictable but repeatable data, called a response. The PUF's design determines the forms of the challenges and responses.Also, as a PUF is derived from the concept of one-way function, it should be impossible to revert the system, meaning that an adversary cannot predict a response as an entry to find the original challenge or vice versa. Finally, as a PUF will always produce the same response to a given challenge, we will talk about the Challenge-Response Pair (CRP), representing the link between a challenge and its response [4].

However, the CRP will change if we build another instance of a PUF (meaning that we take the same design and the same blueprints but build another one with random components and in another environment). Indeed, the way the PUF works is always the same, but due to the manufacturing variation, its internal components are never identical, causing each PUF to (ideally) always produce different responses compared to

other instances. This uniquely allows the PUF to play the role of a perfect identification system, where the set of CRPs is the fingerprint of the PUFs or the device embedded in [59]. The particular dependence of responses on physical parameters and challenges for a given PUF was generally called the challenge-response behaviour of that PUF [60]. Fig. 2.7 shows the PUF's challenge-response behaviour.

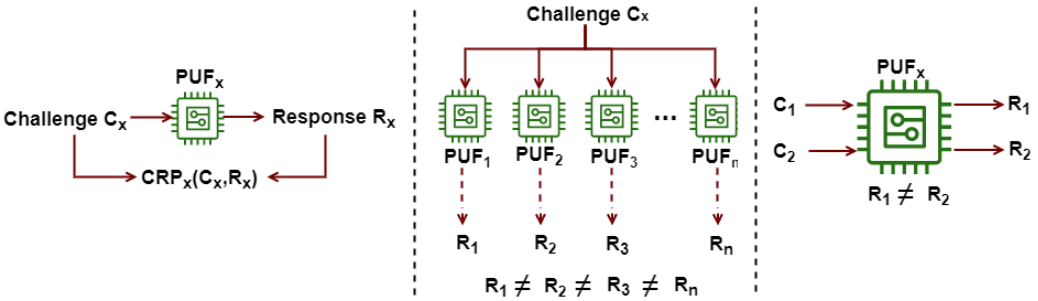

Fig. 2.7 The challenge response behaviour [4].

2.2.1.4 Intra-distance

The intra-distance, also called intra-chip or intra-die of a PUF response, is described by a random variable describing the distance between two responses from the same PUF instance and the same challenge [61]. By taking two evaluations $R_i(c)$ and $R'_i(c)$ of the same PUF instance i and the same challenge c, let dist [. , .] to be any distance metric over the response set R, the intra-distance of a PUF i is given by Equation 2.1 [61].

$$Intra - distance_i \triangleq dist\left[R_i(c), R'_i(c)\right] \tag{2.1}$$

In this survey, responses are always considered as bit vectors, and the hamming distance (HD) is used as a distance metric. Therefore, Equation 2.1 will be:

$$Intra - distance_i \triangleq HD\left[R_i(c), R'_i(c)\right]$$

For a range of [0,1], when the $Intra - distance_i$ result is close to "zero", that means the PUF is highly reliable. Conversely, if the result is close to "one, " the PUF is least reliable. This, due to the environmental conditions under which responses are generated, such as temperature variation and supply voltage [61]. Where the intra-distance between two responses generated from the same challenge with the same PUF instance under the same environmental condition is less than the intra-distance between the same responses generated under two different conditions [62]. Fig. 2.8 shows the intra-distance of a PUF.

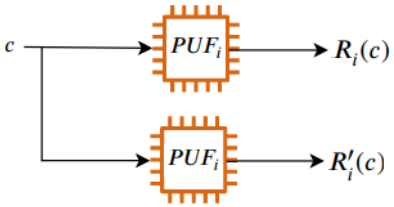

Fig. 2.8 PUF's Intra-distance.

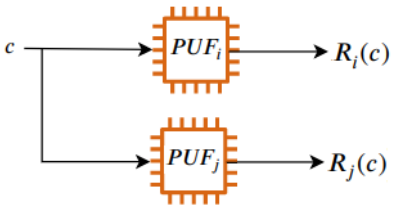

Fig. 2.9 PUF's Inter-distance

2.2.1.5 Inter-distance

The inter-distance, inter-chip, or inter-die of a PUF response is described by a random variable [61]. It is the distance between two responses generated by two different PUF instances, PUF_i and PUF_j, stimulated by the same challenge c. For two responses $R_i(c)$ and $R_j(c)$ of two different PUF instances, i and j, for the same challenge c; Equation 2.2 uses HD as a distance metric to measure the inter-distance of $R(c)$.

$$Inter - distance_{R(c)} \triangleq HD\left[R_i(c), R_j(c)\right] \tag{2.2}$$

If the result of Equation 2.2 is close to "zero" for a range of [0,1] that means the PUF is less unique. Conversely, if the result is close to "one" the PUF is highly unique. The inter-distance between PUF responses is also susceptible to variations in environmental conditions. Fig. 2.9 depicts the inter-distance of a PUF.

2.2.1.6 Environmental effects

In addition to the manufacturing process, which makes the integrated circuits physical disorder objects, environmental variation or variability in the environmental conditions plays a significant role in the circuit operating conditions, and it has a significant impact on the stability and the reliability of the output of the PUF or the device where it is embedded in. The factors that cause this variation can be temperature, power supply, ground bounce, crosstalk, radiation hits, or even aging[1] [3].

2.2.1.7 PUF properties

To show a PUF's strength and robustness, we use its CRP, which acts as a signature or fingerprint. The function $\sqcap : C \rightarrow R$ such that $\sqcap(c) = r$ expresses the relationship between the challenge and the response, where $c \in C$ and $r \in R$. Fig. 2.10 describes the basic PUF properties that we consider [63, 61, 64, 62].

- **Realizable:** A given PUF is realizable if it is easy to invoke its creation procedure and produce a random and unclonable PUF instance given its physical properties.

[1] In some literature, aging is not considered an environmental effect.

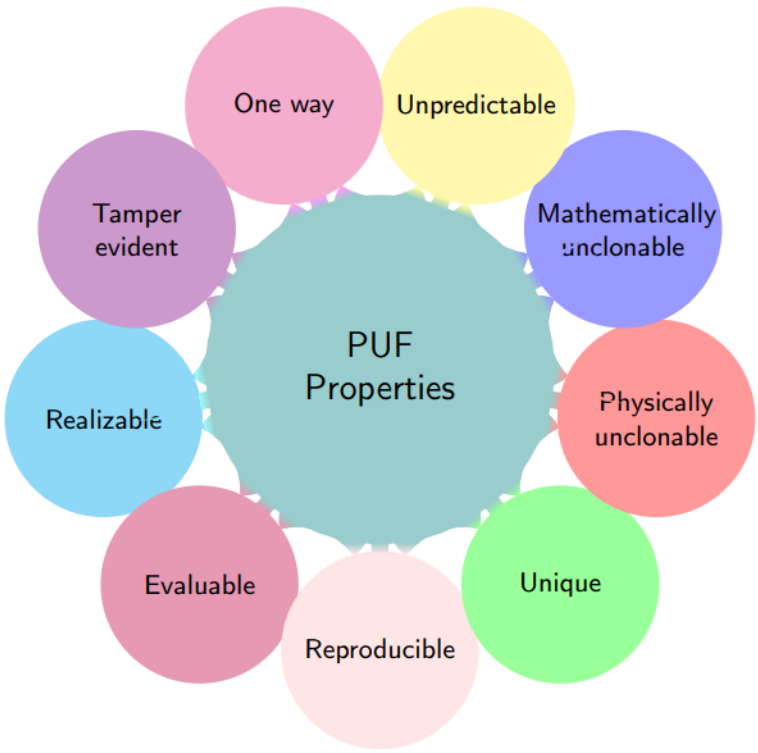

Fig. 2.10 The basic properties of PUF [5].

- **Evaluable:** It means a PUF can[scale=0.5] easily produce a response to a random challenge. For a given $\sqcap$ and c, a PUF should be easy to evaluate according to the function $r = \sqcap(c)$ since it does not need any specific requirements.

- **Reproducible:** For a given challenge, the response may diverge due to the physical environment or the PUF characteristics. Hence, reproducibility means that the PUF must be able to correct this divergence to generate the same response at any time. Thus, a response $r = \sqcap(c)$ can be reproduced with a small error.

- **Unique:** The function $\sqcap$ contains the identity-related information about the physical entity embedding the PUF, which means the CRPs can be used as a unique identifier of the PUF.

- **Physically unclonable:** A PUF was considered unclonable when it was not possible to find a corresponded response r of challenge c without the physical PUF. Even if an adversary has the PUF, it is not possible to make a PUF copy. For a given $\sqcap$, it was difficult to fabricate a physical element containing another PUF $\sqcap$' where $\forall c \in C : \sqcap'(c) \simeq \sqcap(c)$.

- **Mathematically unclonable:** For a given PUF $\sqcap$, it is hard to construct a mathematical procedure $f_{\sqcap}$ such that $\forall c \in C : f_{\sqcap}(c) \simeq \sqcap(c)$.

- **Unpredictable:** For a set of CRPs $Q = \{(c_i, r_i) : \ i > 0 \ \wedge \ r = \sqcap(c)\}$, it is hard

to predict $r = \sqcap(c)$ up to a small error ($r \approx \sqcap$(c)) for a random challenge c which did not appear in Q.

- **One way:** For a given r and $\sqcap$, it is not possible to find $c \in C$ such that $\sqcap(c) = r$.
- **Tamper evident:** Since a PUF is embedded into a physical entity, any alteration of this entity will convert $\sqcap$ into $\sqcap$' and with high probability we got $\exists c \in C$: $\sqcap(c) \neq \sqcap'(c)$ even with a small error ($\sqcap(c) \not\approx \sqcap'(c)$).

2.2.1.8 PUF classes

As shown in Fig. 2.11, we classify PUFs into four classes concerning the implementation technology, the size of challenge-response pairs, the response's dependency, and the physical construction properties.

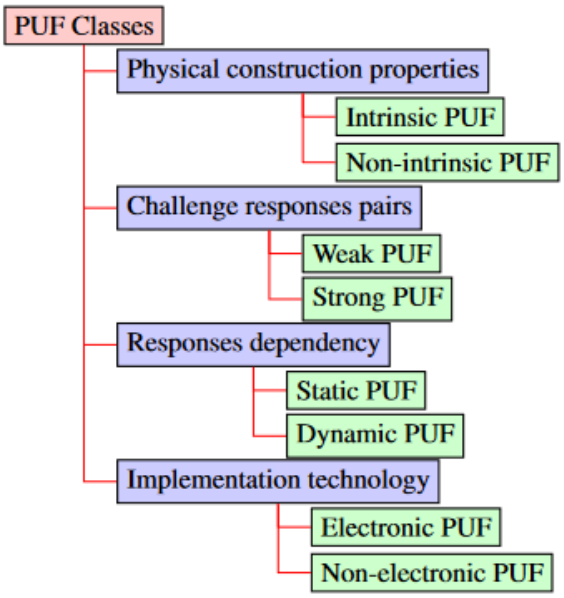

Fig. 2.11 The classification of PUFs [5].

- Physical construction properties: This class is based on the physical structure properties of the PUF that can be intrinsic or non-intrinsic. In the first case, PUF's construction needs to meet at least two conditions: its uniqueness must be assured during the manufacturing processes, and it must internally evaluate itself from embedded measurement equipment. Otherwise, it is considered non-intrinsic [65].
- Challenge-response pairs (CRPs): The size of challenge-response pairs (CRPs) directly impacts PUF applications among metrics that determine their strength and quality. For the size of CRPs, the results exhibited strong or weak PUFs [19]. The weak have a small number of CRPs due to the lower number of symmetric

component blocks used to create the PUF [63]. Thus, an attacker can observe the pairs if he gains physical access to the PUF. Responses from a weak PUF are not public and not unpredictable [66]. Strong PUFs support a massive number of CRPs that grow exponentially with the primary cells or the symmetric component blocks, forming PUFs [63]. This property makes it robust against physical attacks if an attacker has physical access to the PUF. In this case, it is impossible to read all the CRPs since an adversary cannot derive a response to an unknown challenge even with the reverse engineering modeling attacks [67].

- Response dependency: This class is based on the response generation dependency by taking into account the time factor. Practically, the most existing and used PUFs are static, meaning that the generated response is independent of the generation time. In addition to the challenges and the physical features, dynamic PUFs use time as a third dependency, which means dynamic PUFs give different responses to the same challenge at different time slots. Hence, two categories exist in this class: static and dynamic.

- Implementation technology: Various materials and technologies such as glass, plastic, paper, electronic components, and silicon integrated circuits are used to construct PUFs. Thus, each type of material that can be either electronic or not was considered a class of PUFs. The non-electronic PUFs can use electronic subsystems to accomplish their secondary functions [65]. Whereas electronic PUFs use electronic components for their essential operation, such as resistance and capacitance [68].

2.2.2 PUFs Performances

To evaluate the performance of a given PUF, we consider the metrics shown in Fig. 2.12: uniqueness, steadiness, randomness, correctness, bit aliasing, uniformity, reliability, diffuseness and security [69, 70, 71].

For a better mathematical formulation of these metrics, we first use the notation shown in Table 2.1.

2.2.2.1 Uniqueness

Let us consider two PUFs with the exact implementation embedded into two devices d_1 and d_2 that generate respectively responses $R_{d1,m}$ and $R_{d2,m}$ to the same challenge c under the same measurement m where both responses must be very different. Thus, the uniqueness requirement measures how much a PUF instance is different from others by evaluating the uncorrelated responses across dying. When the same challenge sets are presented to different PUFs, the response of each PUF is expected to be different.

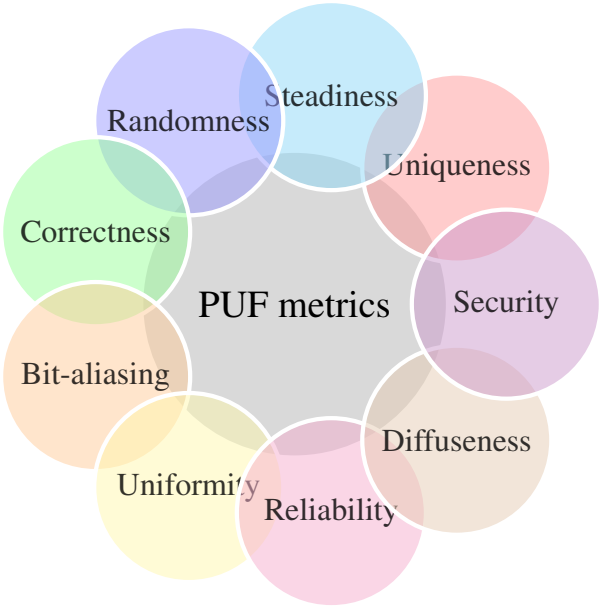

Fig. 2.12 The metrics of PUFs [5] .

Table 2.1 The notation symbols.

Symbol	Description
d	Index of a device ($1 \leq d \leq D$).
D	Number of devices.
c	Index of a challenge ($1 \leq c \leq C$).
C	Number of challenges.
p	Index of a PUF bit within a response vector ($1 \leq p \leq P$).
P	Length of a PUF response vector.
M	Number of measurements.
m	Index of a measurement under an environmental condition ($1 \leq m \leq M$).
R	Number of responses generated by a device.
r	Index of a response $1 \leq r \leq R$.
$r_{p,d,m}$	Binary response bit p of a device d within a measurement m.
$R_{d,m}$	Response vector of a device d for a measurement m with $R_{d,m} = \{0,1\}^p$.

Uniqueness indicates that responses resulting from evaluating the same challenge on different PUF instances should be dissimilar with a high probability.

Since the generated response can be (or be transformed into) a vector of bits, the Hamming distance (HD) will compare two-bit vectors. The HD is the number of positions in which two PUF responses are different, e.g. "11011" and "11011" is 0, while the HD of "11011" and "10101" is 3. The device uniqueness is defined as follows:

$$Uniqueness = \frac{2}{D(D-1)} \frac{1}{P} \sum_{d_1=1}^{D-1} \sum_{d_2=d_1+1}^{D} HD(R_{d_1,m}, R_{d_2,m})$$

The main function in this formula is the hamming distance calculation given by HD. It calculates the sum of XOR operations between each binary response bit $r_{p,d_1,m}$ and $r_{p,d_2,m}$ of two responses $R_{d_1,m}$ and $R_{d_2,m}$ in the m measurement. Ideally, the uniqueness should be close to 50%.

$$HD = \sum_{p=1}^{P} (r_{p,d_1,m} \oplus r_{p,d_2,m})$$

As illustrated in Fig. 2.13, the two instances d_1 and d_2 of a given PUF are assumed to be implemented on two different chips. When a challenge (c_1= 1010) is presented in both instances, with the same measurement m, each PUF generates a unique response. In this case, the HD between both of them is 2 which means the uniqueness of this PUF is 37.5%.

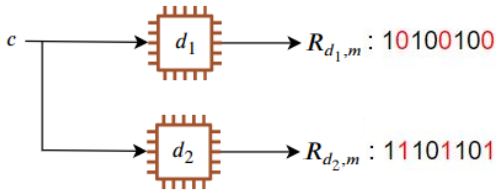

Fig. 2.13 An example of the uniqueness evaluation of a given PUF design.

2.2.2.2 Reliability

This requirement shows how stable a PUF design is when the same challenge values are stimulated for a given PUF instance while the latter should generate the same response values. It measures the repeatability and the consistency with which a PUF generates its response across environmental variations such as ambient noise and aging. To measure the reliability of a PUF, we evaluate the deviation/bias degree of a response generated from the same challenge across different measurements.

For a given device d_1 which has a response R_{d,m_1} of P-bit reference at normal operating conditions m_1 and the response R_{d,m_2} of P-bit at different conditions m_2

for the same challenge c, the reliability is defined as follows using Hamming distance, where less reliability means more changes and instability. The optimal value of the reliability indicator should be 100%.

$$Reliability = 1 - \frac{1}{RMP} \sum_{m_2=2}^{M} \sum_{p=1}^{P} (r_{p,d,m_1} \oplus r_{p,d,m_2})$$

By taking the example of the device d_1 depicted in Fig. 2.14, when a challenge ($c_1 = 1010$) is applied on this device at two different temperatures, m_1 (30k) and m_2 (80k), we observe that the initial response '10100101' differs from the second one '10101101' only on one bit. Then, the reliability of the PUF design embedded on d_1 is 93.75, which means it is not a very reliable design.

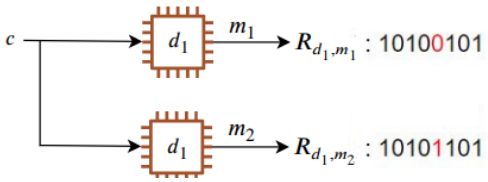

Fig. 2.14 An example of the reliability evaluation of a PUF design.

2.2.2.3 Uniformity

This metric estimates how uniform the n-bit of a response $R_{d,m}$ are distributed by measuring the percentage of '0's and '1's in the response bits. For an excellent response, the proportion of '0's and '1's in its responses should be equal to 50%. The PUF instance is biased towards '0' or '1' in its responses. In this case, the attacker can guess that response. The uniformity of the response bits $R_{d,m}$ is defined as the percentage of Hamming weight (HW) of the n-bit response. So, the uniformity of a response $R_{d,m}$ for a PUF instance d, generated with m measurement, is defined by:

$$Uniformity = \frac{1}{P} \sum_{p=1}^{P} (r_{p,d,m})$$

Taking for example the 8-bit response of '01010101'. The HW of this response is 4 and its uniformity is 50%, which makes it uniform. This due to the same ratio of ones and zeros in the given 8-bit response.

2.2.2.4 Randomness

The P-bit response of a PUF is expected to be uniformly distributed, so the randomness measures the balance of ones and zeros of the response bits value $r_{p,d1,m}$. The optimal

value of the randomness metric is 100%, and for a device d, it is calculated by:

$$Randomness = -log_2 max(p_d, 1 - p_d)$$

p_d is the relative frequency of '1' appearing in all the response bits R generated in a device d at different measurements m, and it is given by:

$$p_d = \frac{1}{RMP} \sum_{r=1}^{R} \sum_{m=1}^{M} \sum_{p=1}^{P} r_{p,d,m}$$

2.2.2.5 Correctness

This requirement gauges how well the PUF responses are accurate. Imagine that a part of the device where a PUF is embedded is broken for some reason after the correct response r for a given challenge c is determined. Then, compared with the correct response, a PUF instance on the device could always generate a wrong response bit value for the same challenge c. In this case, the new response r' becomes stable but incorrect. The correctness requirement is to determine if such a device is defective or degraded by aging. Correctness is similar to reliability, and its ideal value is 100%. It is calculated as follows.

$$Correctness = 1 - \frac{2}{RMP} \sum_{m_2=2}^{M} \sum_{p=1}^{P} (r_{p,d,m_1} \oplus r_{p,d,m_2})$$

The relationship between reliability and correctness is defined by:

$$Correctness = (2 * Reliability) - 1$$

2.2.2.6 Bit-aliasing

This metric estimates the bias of a particular response bit among the set of PUF instances. The bit-aliasing of $r_{p,d_1,m}$ for a challenge c is estimated as the average Hamming weight of the p^{th} bit across different PUF instances. Ideally, this value should be around 50%. The bit-aliasing of the p^{th} response bit generated on the same measurement m, across D different devices is given by:

$$Bit - Aliasing = \frac{1}{D} \sum_{d=1}^{D} r_{p,d,m}$$

Taking for example the challenge ($c_1 = 1010$) applied to three different devices d_1, d_2 and d_3 with the same measurement m (see Fig. 2.15), the HW value of the 1^{th} bit is 2. so, the bit-aliasing of this bit is 67% which means that this bit is biased towards

binary value 1.

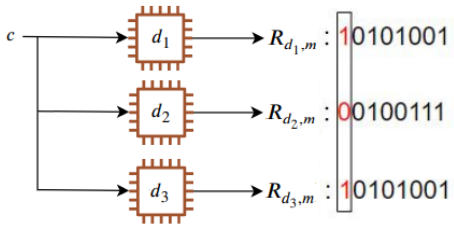

Fig. 2.15 An example of bit-aliasing evaluation of the 1^{th} bit.

2.2.2.7 Steadiness

When applying the same challenge c to the same device d_1 with different measurements m, the output responses R are expected to be identical. The steadiness measures the degree of bias of the p^{th} response bit $r_{p,d_1,m}$ for the given challenge. Steadiness is how strongly $r_{p,d_1,m}$ is biased toward 0 or 1, and its optimal value is 100%. That is calculated as follows.

$$Steadiness = 1 + \frac{1}{RP}\sum_{r=1}^{R}\sum_{p=1}^{P} log_2 max(p_d, 1 - p_d)$$

where

$$p_d = \frac{1}{M}\sum_{m=1}^{M} r_{p,d,m}$$

Fig. 2.16 represents the steadiness metrics of the 1^{th} bit of the response generated for the same challenge c_1 on the same device d_1 with two different measurements, m_1 (30k) and m_2 (80k). The steadiness of this bit challenge is 25% which means that this bit is biased towards binary value 0.

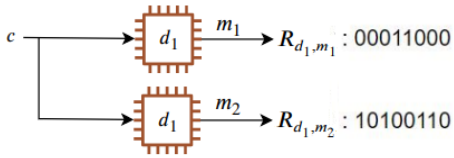

Fig. 2.16 An example of the steadiness evaluation of the 1^{th} bit.

2.2.2.8 Diffuseness

As a result of applying different challenges to the same PUF instance, the generated responses should be different. The diffuseness represents the difference among the generated responses from different challenge sets in the same PUF instance. Using the mean Hamming distance (HD) of the generated responses R from the challenges C, on

the same device d_1 and with the same measurement m. The diffuseness of the device d is defined by:

$$Diffuseness = \frac{4}{P.R^2} \sum_{r_1=1}^{R-1} \sum_{r_2=r_1+1}^{R} \sum_{p=1}^{P} (r_{1(p,d,m)} \oplus r_{2(p,d,m)})$$

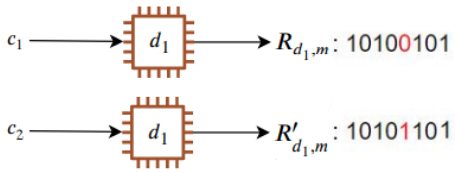

Fig. 2.17 An example of the diffuseness evaluation of a PUF design.

Any cryptosystem is exposed to classical cryptosystem attacks like trying to read out secret keys from memory and communication attacks, in addition to two new threats: Side-Channel Attacks, where an attacker has physical access to the device, and modeling attacks, where the adversary has a large number of CRPs. In addition to these metrics, some researchers consider security as a metric too[72].

2.2.2.9 Security

It is the ability of a PUF to resist all attacks. In contrast to the previous metrics, there is no specific formula to evaluate the security of PUFs.

2.2.3 PUFs Attacks

PUFs are a great solution to replace the actual protection mechanisms, such as hash functions and secret-key algorithms. Unfortunately, they are not entirely secure and suffer from some vulnerabilities. Strong PUFs are very difficult to break, and the current technologies are not advanced enough to manage to break them compared to weak PUFs that can be easily broken by different types of attacks. Security attacks are generally classified into invasive, semi-invasive, and non-invasive attacks [3].

2.2.3.1 Non-invasive attacks

Non-invasive are low-cost attacks based only on observation and speculation about the device without harming it. This type of attack does not need much equipment as the field of action is restricted. The attackers extract secret information by exploiting only the observed data (e.g., power consumption, delay time, and CRPs) without direct access to PUF components. The two most celebrated types of non-invasive attacks are machine learning (ML), and side-channel attacks (SCAs) [73].

- **Machine learning attacks:** Strong PUF's challenge-response behaviour is vulnerable to modeling attacks, which use machine learning algorithms to predict PUF responses. The principle of machine learning is to create a specific algorithm and give it some sample data to train the algorithm and create a statistical model that will simulate the PUF's behavior. Then, it will be easy to request the model to predict new data. After constructing an adversary machine and collecting a subset of all the CRPs of the target PUF, the attacker can build a numerical model from the collected CRP data. Thus, the model can be used for future response prediction to arbitrary challenges with high probability. In [74], **(author?)** showed that several PUF architectures [7, 75, 8, 16, 6, 76] can be broken using various machine learning techniques, including Support Vector Machines (SVMs), Evolution Strategies (ES), Logistic Regression (LR), and also briefly Neural Nets and Sequence Learning [77]. In [78], probably approximately correct (PAC) learning has been used to develop attack models for Arbiter, XOR Arbiter, RO-PUF, and BR-PUFs. To check the robustness of PUFs towards ML attacks, **(author?)** [79] developed a testing environment, called PUFmeter, to evaluate the security of PUFs under ML attacks. Also, **(author?)** [80] relied on the PAC-learnability of PUFs to derive the PAC-learnability bound from the representation of a PUF architecture described in the PUF-G language [80]. Then, they verify the robustness and resilience of the given PUF against ML-based attacks.

  As a countermeasure against ML attacks, **(author?)** [81] proposed raising the number of XORs in an XOR Arbiter PUF and a Lightweight PUF. In order to mitigate the PUF modeling vulnerability, **(author?)** [66] proposed to encrypt challenge-bit via the AES algorithm, where the encryption key is generated using a weak PUF. Similarly in [82], instead of storing the response, the hash value of the PUF response is stored on the server. To resist against ML attacks, **(author?)** [83] proposed a new PUF construction called a CRC-PUF where the input challenges are de-synchronized from the output responses. **(author?)** [84] proposed a challenge permutation and substitution techniques to increase the ML-attack resistance of Strong-PUFs. They showed that the predictability of Arbiter-PUF and TCO-PUF responses could be reduced to less than 70%. **(author?)** [85] showed that challenge obfuscation schemes implemented on a standard arbiter-PUF makes it secure against modeling attacks.

- **Side-channel attacks:** Another non-invasive example that is (hopefully) more complex is the side-channel attack [86], well-known in cryptanalysis. The idea is to exploit leaked information from the physical implementation of a cryptographic primitive of the device running the algorithm or the physical system regarding PUFs and extract information as much as possible to understand the

algorithm's behaviour. For example, when attempting to break RSA, some attackers tried to measure their CPU usage to understand when the most extensive computation occurs, which one, and even the produced data. This type of attack is classified into two groups: passive and active attacks. The attacker observes side channels in the first case, such as timing delays, power consumption, temperature, and electromagnetic noise. In the second, the attacker requires information on the internal structure and operation of the PUF, such as fault injection methods [87]. Various s-channel techniques have been applied to PUFs, such as Helper Data Leakage, Power Analysis Attacks, and Fault Injection Attacks [3]. **(author?)** [88] exploited the information leaked through the power side-channel in the initial step in the syndrome decoding phase of BCH and Reed-Solomon decoder fuzzy extractor implementations to recover the fuzzy extractor's input that refers to the PUF's response. **(author?)** [89] have demonstrated how RO PUFs can be attacked using electromagnetic (EM) attacks. In [90], **(author?)** analyzed side-channel vulnerabilities of the Loop PUF and showed that it is vulnerable to side-channel analysis (SCA) attacks. **(author?)** [91] proposed a power consumption and time-side channel attack method for XOR PUF and lightweight security PUF. **(author?)** [85] showed that arbiter-PUFs based challenge obfuscation schemes are vulnerable to power side-channel attacks. In [73], authors have also proposed a combined side-channel and modeling attack.

As a countermeasure against PUF Side-channel attacks, **(author?)** [89] proposed a measurement path randomization by randomizing the RO selection logic and the interleaved placement to disguise RO EM emission as two countermeasures. Also, to mitigate the attack presented in [90], they introduced a countermeasure based on temporal masking to thwart side-channel analysis that requires only one bit of randomness per a PUF response bit. **(author?)** [85] proposed dual flip-flop mitigation and randomized response settings to improve the resiliency of challenge obfuscation PUFs against power trace attacks.

2.2.3.2 Invasive attacks

This type of attack is among the most expensive ones where the attacker can extract information from the system, understand the internal behavior, and access the device. However, the required equipment to achieve this type of attack is expensive and requires more knowledge and time. However, invasive attacks are less popular, and they need more sophisticated equipment and precise expertise. Micro-probing and reverse engineering are two types of invasive attacks [92]. The first one requires a micro-probing station, a gigantic microscope with some probes to get information, such as the electric signals, from the circuits to understand the interactions between the different compo-

nents and when they need to communicate. This station can also be used to alter the device, as we can manipulate the system. And the second one requires observing and manipulating the device or the software to derive some information about its behaviour. Then, the attacker can attempt to reproduce it.

2.2.3.3 Semi-invasive attacks

This class is a compromise between the attacks categories mentioned above. In terms of requirements (affordability and knowledge), it is between invasive and non-invasive. In addition, there are many other types of attacks appropriate to a specific target of a PUF design. They have access to some parts of the internal devices in a system without damaging them. For example, an attacker can add extra circuitry with malicious functionality into a PUF design. When the PUF is used, the attacker uses this circuit to access the PUF. This attack is called 'Trojan insertion'. Also, the PUF can be attacked by exploiting the vulnerabilities of the application domain, like the man-in-the-middle attack, where the attacker tries to intercept the transport data used in the authentication protocol [3].

2.2.4 Summary

In this section, we have defined physical unclonable functions in general. First, we presented the different aspects and concepts that allowed the birth of PUFs. We also provided the needed PUF properties and their classifications regarding the implementation technology, the size of challenge-response pairs (CRPs), the response's dependency, and the physical construction properties. After that, we have presented the nine metrics used to evaluate PUFs by giving the mathematical formula and the graphical representation for each metric, except the security one. Finally, we have classified the existing PUF attacks into non-invasive, invasive, and semi-invasive.

2.3 Conclusion

In this chapter, we present the necessary context and concepts for the subject of this thesis, such as the internet of things and physical unclonable functions. In the first section, we discussed the IoT by discussing its characteristics, applications, architecture, and open concerns. Second, we have described the various factors and ideas that led to the development of PUFs. In addition, we included the necessary PUF features, their classifications, and the existing assaults.

In the following chapter, we will conduct a literature review pertaining to our issue, beginning with a discussion of the silicon PUF architectures and applications, followed by a survey of IoT-based authentication mechanisms. IoT PUF-based authentication

mechanisms will next be examined, discussed, and compared. The next chapter is vital to the description of the proposed protocol.

CHAPTER 3

STATE-OF-THE-ARTS

With the wide range of IoT applications, vast amounts of data are collected, extracted, managed, communicated, analyzed, and stored. Furthermore, because of the physical characteristics of IoT devices, such as memory capacity, processing power, and energy resources, it is difficult to apply traditional cryptosystems and symmetric/asymmetric algorithms to constrained applications. Their complexity makes them unsuitable for ensuring the security of IoT systems, including authentication, privacy, access control, as well as data collection and management. Authentication is a critical requirement and the main security challenge in the IoT. Any weakness in the authentication process leads to a compromised IoT network. Authentication refers to how to verify a device's identity and prevent malicious devices from accessing the IoT system. In the last decade, PUFs became a perfect and ideal security primitive used in IoT systems, principally during the authentication phase, based on the PUF's behavior.

The main focus of this chapter are:

1. Surveying the recent silicon PUF architectures and applications.
2. Comparing the existing architectures and applications related to silicon PUFs.
3. Surveying the existing techniques used in IoT-based authentication protocols.
4. Given a review of recent existing contributions to IoT PUF-based authentication protocols.
5. Classify, compare and discuss the reviewed work.

3.1 Silicon PUF

This section focuses more the existing architectures proper to Silicon PUF as well as their related applications.

3.1.1 Silicon PUF Architectures

The PUF is a one-way function that exploits the unique random imperfections found at the nano-scale level of the structure of physical objects [3]. A PUF could be defined as a “digital fingerprint” that is derived from a complex physical object. It is like a black box that takes a challenge as input and produces a response that can be used as a unique identity of the subject or as a cryptographic key.

The term ”silicon PUF” has been introduced in [93], which refers to physical unclonable objects built using conventional integrated circuits. Silicon PUF forms a major subclass of electronic PUFs considered integrated circuits (ICs). They can be embedded in silicon chips to accomplish PUF’s goals by exploiting their manufacturing process [68]. The Silicon PUF is certainly the simplest PUF as it does not require any modification in the manufacturing process to be used. It exploits the inherent manufacturing variations of transistors and wires that differ from one circuit to another, even if they are part of the same silicon wafer. The Arbiter PUF is the first silicon PUF, introduced by **(author?)** [76].

According to the different sources of variation, silicon PUFs can be categorized into three major classes: delay-based PUFs, memory-based PUFs, and analog electronic PUFs.

3.1.1.1 Delay-based PUFs

The response generated by the delay PUFs depends on the propagation delay between the different delay paths of the PUF’s circuits, and it can be affected by the temperature changes of the circuit [94]. Mainly, this type of PUF includes:

- **Arbiter PUF:** Due to the inherent manufacturing variations of transistors and wires, each IC has its own unique delay characteristics, **(author?)** [6] used this property to build secret information unique to each IC, which is called arbiter-based PUF or multiplexer (MUX) PUF.

  The idea behind the arbiter PUF is to explicitly introduce a race condition between two digital paths on a silicon chip. It consists of the two delay paths as chains of switch blocks (multiplexers) and an arbiter block at the end of the chain. As shown in Fig. 3.1, the switch block has two possible configurations depending on the challenge bit; straight if the challenge bit is 0 and crossed if it is 1. Each switch block has three outputs: the two outputs from the previous stage and a single bit

of the challenge. The inputs of the first switch block are connected to a common enable signal, and the outputs of the last switch block are connected to the arbiter block, which determines which signal arrived first. The arbiter generates a single bit known as the response bit based on this result.

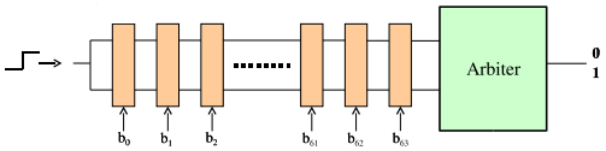

Fig. 3.1 First structure of Arbiter PUF [6].

The same authors of [6] showed that by exploiting the linearity of delay paths, an arbiter PUF was not secure against machine learning attacks. To introduce non-linearity into the PUF scheme, they proposed the feed-forward arbiter PUF (FF APUF) [95], which is an extension of their primary arbiter PUF, where an intermediate arbiter internally generates some challenge bits. Then, these challenges are hidden from an adversary.

Fig. 3.2 depicts the concept of a feed-forward arbiter PUF scheme with one feed-forward arbiter.

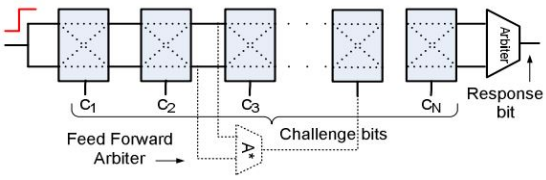

Fig. 3.2 The feed forward arbiter PUF [7].

In the same direction, several constructions based on the Arbiter PUF have been proposed, such as: XOR PUF or XOR-Arbiter PUF [16], Feed-Forward XOR PUFs [75, 9], Lightweight PUF [7], $m-n$ APUF [96], Multiplexer-based arbiter PUF [11], multi-PUF (MPUF) [12], multi-PUF [97], and Interpose PUF (IPUF) [14].

XOR PUF or XOR-Arbiter PUF [16] combines several rows of the basic arbiter PUF by XORing the outputs of each arbiter PUF into a one-bit response. The length of this implementation is measured by two factors (the length of the challenge's number of switch blocks and the number of rows that indicate the input size of the XOR). Fig. 3.3 shows a 2-XOR PUF with two rows and n switch blocks.

Recently, **(author?)** [75, 9] proposed Feed-Forward XOR PUF, which is a combination between Feed-Forward APUF and XOR PUF. Instead of using APUF as a component of XOR PUF, FFXOR PUF uses FF APUF as a new component.

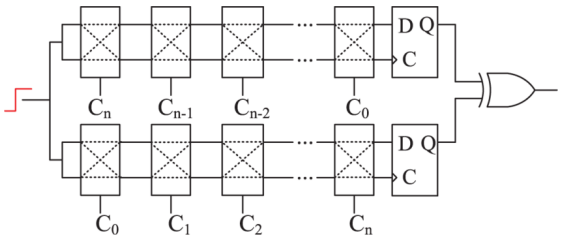

Fig. 3.3 An example of 2-XOR PUF [8].

According to [75] [75, 9], FFXOR PUF has shown good reliability, uniqueness, and resistance against attacks compared with the classical XOR PUF. However, no document has proposed or analyzed the safety and reliability aspects of this proposed PUF. Fig. 3.4 shows the general architecture of the Feed-Forward XOR PUF.

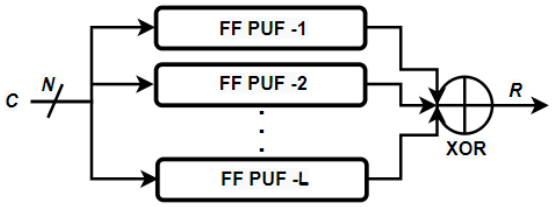

Fig. 3.4 The architecture of Feed-Forward XOR PUF [9].

Lightweight Secure PUFs or Lightweight PUFs have been introduced by **(author?)** [7]. It is a variant of the XOR APUF based on several APUF arranged in parallel. However, the challenge bits are rearranged and modified for each chain. Also, the output response bits of each chain are XORed to obtain a multi-bit response. Fig. 3.5 shows the general architecture of the LSPUF. Since LSPUF outputs are generated using x-XOR PUF, most of the attack strategies developed for XOR PUF can also be applicable to LSPUF, which consequently makes it vulnerable to LR[98] when x$\leq$9.

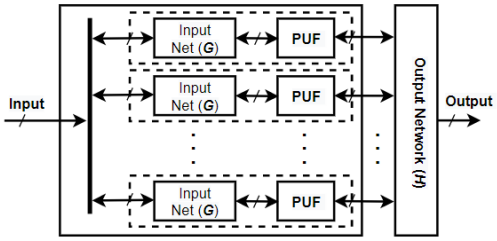

Fig. 3.5 The architecture of LSPUF [7].

From a security perspective, **(author?)** [74] showed that all the previously presented PUF implementations can be attacked using ES and LR machine learning attacks, and recently, in [99] authors proved that APUF, XOR APUF, and FF APUF are vulnerable to Deep Learning (DL) modeling attacks. To enhance the

unpredictability of APUF's responses, **(author?)** [96] proposed $m - n$ APUF or double arbiter PUF (DAPUF). Like $n-$XOR PUF, it is based on APUF, where m refers to the number of chains and n to the length of the response. Instead of

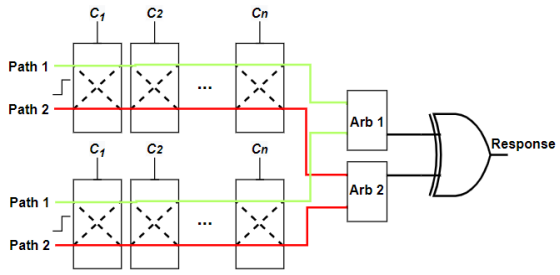

Fig. 3.6 The structure of the $2 - 1$ Double Arbiter PUF [10].

comparing the propagation delays of two paths of the same chain like APUF do, DAPUF compares the propagation delays of the same paths across *m* chains. The response of DAPUF is obtained by XORing all the results of the last comparison process. The experimental results showed that the uniqueness of the proposed 3-1 DAPUFs was approximately 50%, which is much superior to that of 3-1 APUFs. In [100], **(author?)** proposed a 4-1 Double Arbiter PUF and compared 3-1 DAPUF with $3-$XOR PUF. This comparison showed that 85% of the responses from the second design could be predicted with machine learning. Contrarily, a 3-1 DAPUF resulted in a prediction rate of 57%, and recently, modeling attacks have been successful against different DAPUFs architectures [10] except for the 4-1 DAPUF. Fig. 3.6 shows the structure of the 2-1 DAPUF.

(author?) [11] proposed a Multiplexer-based arbiter PUF (MPUF) built with multiplexers and APUFs. An $(n, k)-$MPUF consists of a $2^k - 1$ MUX and $2^k + k$ APUFs where each APUF receives n bit challenge. The outputs of $2^k + k$ APUFs are used as inputs of MUX, where each MUX of the $2^k - 1$ MUXs has three inputs, two data inputs from 2^k APUFs, and one selection input from k APUFs. The $2^k - 1$ MUX selects one of the data inputs as the final response. The robustness of this PUF is that an adversary does not have access to responses of $2^k + k$ APUFs. Fig. 3.7 shows the architectural overview of an $(n, 3)-$MPUF which generates a one-bit response to an *n* bit challenge by using 7 MUXs and 11 APUFs.

Based on PUF composition principles, two major challenges have been identified to overcome vulnerability against modeling and statistical attacks and lack of reliability. In the same paper, **(author?)** [11] proposed two other variants, rMPUF and cMPUF, to ensure reliability and to resist respectively to ML-based attacks and linear cryptanalysis (LC) attacks. Unfortunately, MPUF and its variants can be broken by two recently proposed attacks: logical approximation method and filter-based global approximation attacks [101]. Fig. 3.8 shows an example of

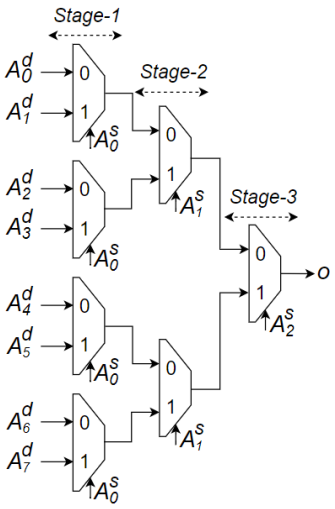

Fig. 3.7 The structure of $(n, 3)-$MPUF [11].

(n,3)-rMPUF and (n,2)-cMPUF MPUF variants.

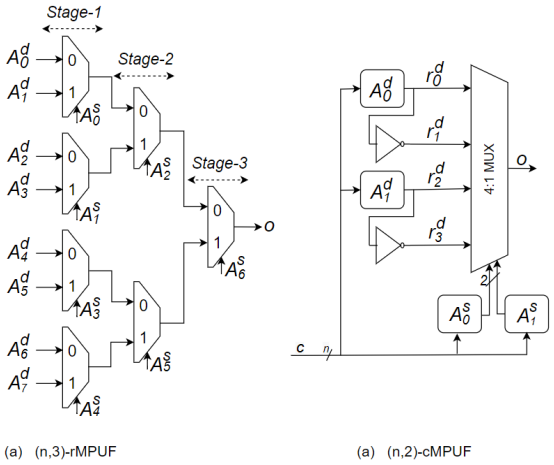

Fig. 3.8 Example of MPUF variants: (a) the basic (n,3)-rMPUF and (b) (n,2)-cMPUF. [11].

Using the same names but with different implementations, **(author?)** proposed a new arbiter-based multi-PUF (MPUF) [12] as a combination of weak and strong PUF. As shown in Fig. 3.9, MPUF is composed of n PicoPUF [102] and one Arbiter PUF with n switch blocks. To mask the original challenge bit C_i, it is XORed with the response k_i of the i^{th} PicoPUF to generate the new challenge $C^*{}_i$, which is used as the challenge for APUF. As the input of the strong PUF is depending on the output of weak PUF(s), the response of this strong PUF has a strong uniqueness and reliability. MPUF is vulnerable to Deep Learning (DL) modeling attacks [99].

(author?) [13] showed that the uniqueness and the reliability of PUFs could not be guaranteed due to the low hardware resources and the small CRP space. Thus, to enhance the performance of PPUF, they proposed a reconfigurable Pico-PUF

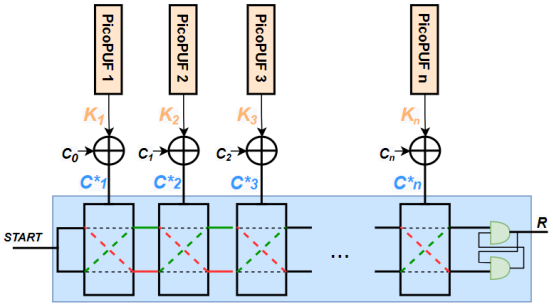

Fig. 3.9 The multi-PUF design based on a PicoPUF and APUF [12].

(RPPUF) composed of two configurable logic structures, as shown in Fig. 3.10. The RPPUF is a simple NAND-based SR latch with two flip-flop structures and two configurable logic circuits connected before the set-reset latch.

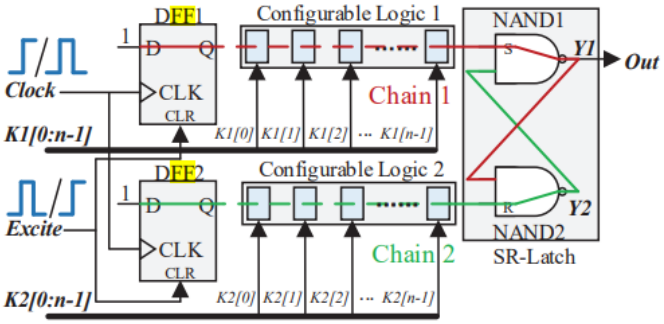

Fig. 3.10 The proposed RPPUF design with configurable logic [13].

Another multi-PUF implementation was proposed by **(author?)** [97] by combining the Ring-Oscillator PUF [16] and Arbiter PUF. The composed PUF is called a Composite PUF, and it is characterized by a larger challenge space and superior quality metrics for each of its components. However, this combined PUF is not secure against cryptanalysis, and modeling attacks [98].

(author?) proposed one of the most recently designed strong PUFs, called Interpose PUF (IPUF) [14], a combination of two XOR PUF. As shown in Fig. 3.11, *an (x, y)*-IPUF consists of two layers, the upper layer and the lower layer. The upper layer is a *x*-XOR APUF (x arbiter PUFs) with n challenge bits, whereas the lower one is a *y*-XOR APUF (y arbiter PUFs) with $n + 1$ challenge bits. The response r_x of the x-XOR APUF is interposed in the n^{th} challenge bit to form $n + 1$ challenge bits.

The experimental results showed that iPUF is not vulnerable to the reliability-based machine learning attack (CMA-ES) and the classical machine learning attack (Logistic Regression). But, the iPUF of 64-bit challenge length and size of 8 APUF in both layers is broken by the modeling attacks [103].

In order to improve APUF's security against machine learning attacks, **(author?)**

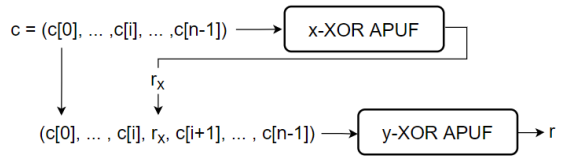

Fig. 3.11 The structure of the (x, y)-iPUF [14].

recently proposed a complex model of APUF, called the Racing APUF (R-APUF) [15]. R-APUF consists of two symmetric paths. However, instead of MUX, the path of R-APUF consists of sub-chains. Each sub-chain has a series of stages based on MUX. the sub-chain is ended by a route selector such as an AND gate or OR gate. R-APUF is characterized by the number of sub-chain in each path and the number of channels in each sub-chain. The structure depicted in Fig. 3.12 can be referred to as a 2-channel 2-stage R-APUF.

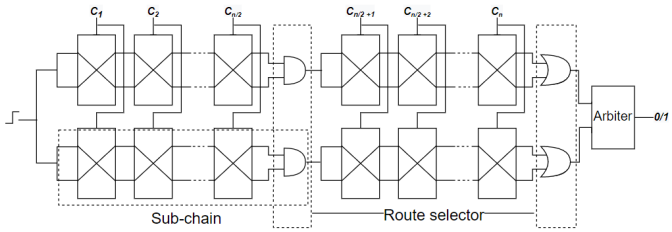

Fig. 3.12 2-channel 2-stage racing APUF [15].

- **Ring-Oscillator PUF:** Rather than the basic arbiter PUF and its derivatives, ring oscillator PUF (RO PUF) is another PUF design based on the delay difference of identical electrical paths initially proposed by **(author?)** [16].

  As represented in Fig. 3.13, a typical RO PUF consists of N identically laid-out delay loops, or ring oscillators (ROs), two multiplexers, two counters, and an arbiter. Theoretically, each RO oscillates at the same frequency, but due to manufacturing variations and environmental conditions, it oscillates at a slightly different frequency. To generate a one-bit response from these N ROs, a pair of ROs needs to be selected. This selection is determined by the input (challenges) applied to both MUX and a comparison of the frequency of the selected RO pair. The response bit is set to 1 or 0 depending on the comparison, 0 if the first oscillates faster than the second, and 1 if it is not. From N ring oscillators, RO-PUF can produce $log_2(N!)$ bits [16]. For example, 32 oscillators can produce 118 bits. Compared to APUFs, RO PUFs allow easier implementation for FPGAs and ASICs, easier evaluation of entropy, and higher reliability. Nevertheless, RO PUFs took longer, used more power, and needed more space to make the responses.

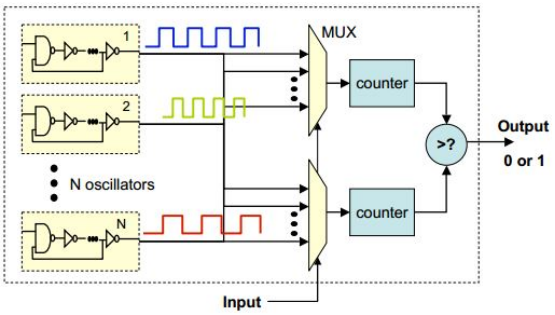

Fig. 3.13 Ring oscillator based PUF circuit [16].

Due to the low number of CRPs generated by RO PUF, it was classified as a weak PUF and is vulnerable to cryptographic analysis attacks. In [74] they showed that machine learning algorithms could model RO PUFs, and in [104] they used electromagnetic attacks to break the security of RO PUFs. Therefore, several variants of RO PUFs have been proposed.

To reduce the noise in RO PUF responses and increase the number of CRPs of the basic RO PUF, the first configurable ring oscillator PUF (CRO PUF) has been introduced in [17]. As shown in Fig. 3.14, a multiplexer has been added after each stage of the RO to check if the inverter will be selected as a member of the RO. According to the input selection bit, each MUX selects one output of the two inverters. So for RO with three stages, eight configurations are possible.

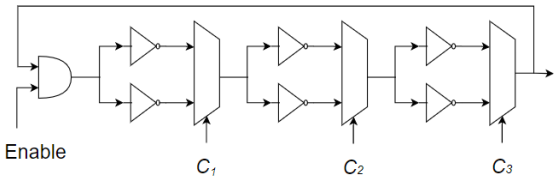

Fig. 3.14 Configurable RO [17].

Based on the same idea, **(author?)** [18] proposed another configuration of RO PUF, called configurable RO PUF or flexible RO PUF [18]. Fig. 3.15 shows that the selection of an inverter from the ring is chosen dependingn the input selection bit. If the bit is 0, the corresponding inverter is discarded, else it will be used in the ring. So, for a RO with three inverters, eight configurations are possible. CRO PUF is vulnerable to modeling attacks while it is characterized by a low number of CRPs as well as to machine learning attacks [105]. Fig. 3.15 depicts the architecture of one ring of CRO PUF.

(author?) [106] proposed the k-sum PUF that consists of k pairs of ring oscillators. To generate the one-bit response, k-sum PUF measures the difference between two delay terms, each produced by the sum of k ring oscillator values. To build these two terms for each k stage (Fig. 3.16), the challenge bit C_i defines

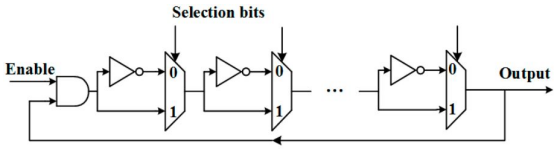

Fig. 3.15 Architecture of the configurable RO [18].

which RO is used to compute the bottom and top delay terms. However, K-sum PUF is vulnerable to machine learning attacks [107].

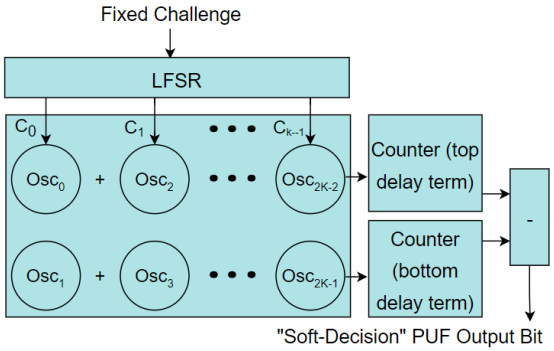

Fig. 3.16 K-sum PUF [19].

In [20], **(author?)** proposed the Transient Effect Ring Oscillator (TERO) PUFs as an alternative to RO PUFs with a similar structure, but it is constructed from TERO cells that have two states: stable and transient oscillating. As shown in Fig. 3.17, the basic structure of a TERO PUF is an RS flip flop, where the TERO cell is composed of two identical and symmetrical branches (Branch 1, Branch 2). Each branch is designed with an initialization stage and inverters whose exact number is used for both branches. The circuit starts oscillation for a short time by setting the init signal to one and depending on the mismatch in the delays between the two branches of the TERO cell caused by CMOS process variations. This behavior results in a finite number of oscillations of the TERO cell output that is considered as the TERO PUF response. Also, they showed that TERO PUF is not as susceptible to frequency injection and cloning attacks through electromagnetic analysis. But in [108], **(author?)** showed that using non-invasive electromagnetic measurements and tailored attack methodology could recover up to 25% of the TERO PUF response's bits without errors.

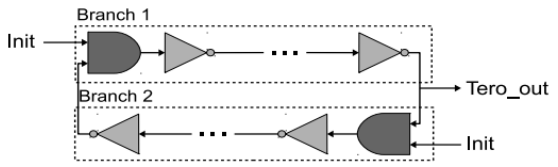

Fig. 3.17 Generic Structure of a TERO cell [20].

Recently, **(author?)** [21] proposed a new FPGA-compatible design named the Delay Difference PUF (DD-PUF), which requires a minimal area footprint and provides excellent reproducibility under temperature and supply voltage variations. Fig. 3.18 describes a single DD-PUF cell composed of two inverters (I_1 and I_2), interposed between two D-Latches (L_1 and L_2) forming two identified paths that can be identified ($P_1 = L_1$-I_1 and $P_2 = L_2$-I_2). The DD-PUF needs two control signals, START and RESET, connected to the enabling gate and used to clear the pins of the two latches. When the asynchronous RESET is set to 1, both latches' output pins are forced to 0. When the START signal is set to 1 for a period of time interval, an oscillatory state is produced within the DD-PUF cell. At this point, only the small delay difference between P_1 and P_2 determines the resulting stable bit (response).

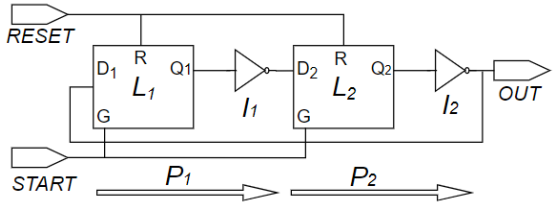

Fig. 3.18 The architecture of a single DD-PUF cell [21].

- **Glitch PUF:** It is the first FPGA-specific PUF [22] design proposed to reduce the ease of predicting the relationship between challenges and responses in delay PUFs. GPUF exploits glitch waveforms caused by variations in the delay between gates to generate the responses. Its architecture consists of three parts: 1) a combinational circuit for generating glitch waveforms, 2) a sampling circuit for Glitch, and 3) a response generator. First, the input value of the glitch generator is presented to a data register as a challenge. Then, the acquisition of the glitch waveforms. Finally, the conversion of the waveforms into response bits. Compared to other PUF designs, GPUF has good performance, and it is ranked among the most secure PUFs against modeling attacks. Fig. 3.19 represents the whole structure of Glitch PUF.

  As shown in Fig. 3.19, the circuit area of the discussed glitch PUF is large. Hence, **(author?)** [23] have proposed a simplified glitch PUF called the second glitch PUF. As shown in Fig. 3.20 the second PUF glitch is simplified in terms of eliminating certain circuit blocks. More precisely, the sampling circuit. In addition, the output of the glitch generator is connected directly to the toggle flip-flop converter (TFF). From the security side , no successful machine learning attack model against the two glitch PUF designs has been proposed.

- **Intellectual property PUF (IP-PUF):** To ensure the intellectual property (IP)

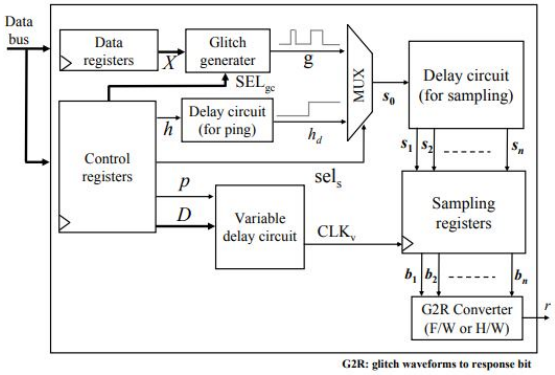

Fig. 3.19 Whole structure of Glitch PUF [22].

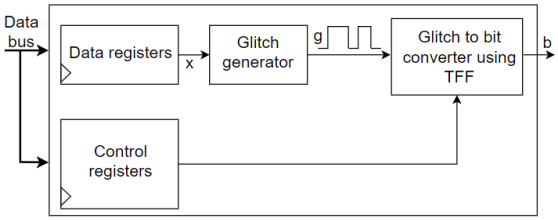

Fig. 3.20 Second glitch PUF [23].

of personal use, **(author?)** [109] proposed the use of a set of silicon circuits embedded on a personal computer (PC) as a PUF named Intellectual Property PUF (IP-PUF). Mainly, the authors used the intrinsic features found in silicon circuits to exploit mismatches in frequencies of oscillators of the CPU clock or the timer interrupt clock. Then, by exploiting the value of the time period needed to load instructions from the processor cache into the register memory that varies from one PC to another one.

- **Clock PUF:** The clock network routes a timing signal from the clock to various sections of the circuit design. It ensures synchronicity by respecting the time taken by the signal from the clock to reach any given area of the circuit. Otherwise, the issue of clock latency variation is known as clock skew. Based on these variations and skewing, **(author?)**[24] proposed the clock PUF (CLK-PUF) similar to an arbiter PUF since it uses MUXes to select two paths of the clock network and compares their delays using an arbiter to generate a response bit (Fig. 3.21). CLK-PUF has been broken by machine learning based attacks [14] and is vulnerable to non-invasive attacks [110].

3.1.1.2 Memory-based PUFs

The response generated by the memory-based PUFs depends on the initial state of the memory structures. At a power-up, the structures are set in an unstable state, and the response corresponds to the stable state of the structures caused by an external data

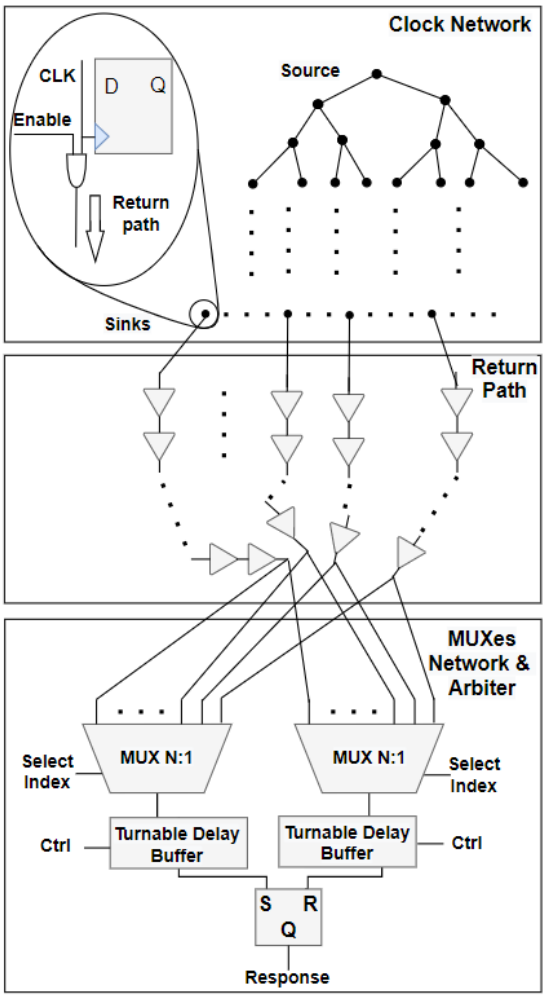

Fig. 3.21 The architecture of Clock PUF [24].

signal input [94]. This type of PUF family mainly includes:

- **SRAM PUF: (author?)** [111] proposed static random-access memory, or SRAM PUF, as the first intrinsic PUF construction based on the power-up state of an FPGA's SRAM memory. It does not need any modifications in the manufacturing process. It is based on the static-noise margin (SNM) that requires a memory cell to change its logical value. A SRAM cell is logically constructed as two cross-coupled inverters, hence leading to two stable states [62]. During the start-up, the initial value 0/1 of a SRAM cell is given randomly and independently by the SNM. This randomness is due to the manufacturing process of the SRAM cell. In order to generate the response, SRAM PUF uses a range of memory locations of an SRAM memory block as a challenge, and the responses are the start-up values of the whole SRAM cells that compose the challenge. In its first implementation, SRAM PUF was used in protocols for the IP protection problems implemented on FPGAs. Fig. 3.22 shows the design of the SRAM PUF cell with six transistors. In [112], authors showed that it is possible to clone SRAM PUF.

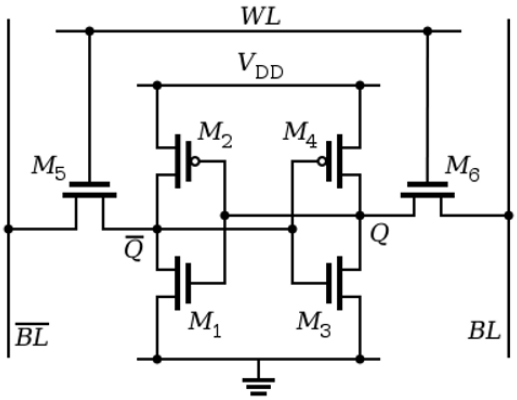

Fig. 3.22 SRAM cell with 6 transistors.

The start-up values of the SRAM cells are controlled by the IC manufacturer, which renders SRAM PUF useless for FPGAs [113]. To overcome this issue, many improved implementations of the SRAM PUF have been proposed, such as the Butterfly PUF [25], Flip-flop PUF [113], Latch PUF [114] and Buskeeper PUF [27].

SRAM PUFs [111] are used only on FPGAs that support initialized SRAM memory. In order to resolve this problem, **(author?)** proposed replacing the inverter with latches or flip-flops to build a cross-coupled circuit, and they called it Butterfly PUF [25]. As shown in Fig. 3.23 the structure of the BPUF cell consists of two latches, where each latch is a cross-coupled circuit, which represents a fundamental building block used in all types of storage elements in electronic circuits. This cross-coupled circuit has two different stable operating points, 0/1 and

an unstable operating point. An unstable state can be introduced, after which the circuit converges back to one of the two stable states. BPUF exploits this random assignment of a stable state from an unstable one to generate the secret key. This assignment can be comparable to the stat of the SRAM cell after power-up. After experimentation, they found that the proposed PUF can be used in IP protection and in cryptographic applications by generating a secret volatile key.

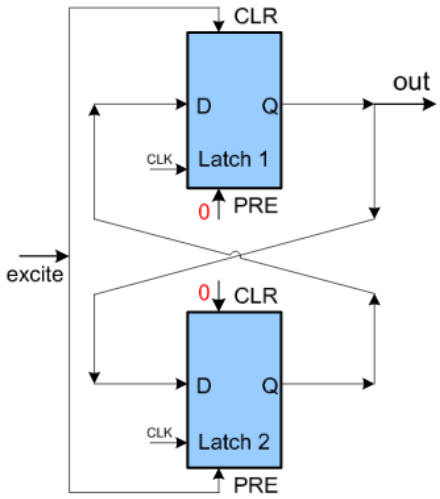

Fig. 3.23 Butterfly PUF cell [25].

(author?) used the power-up values of the flip-flops present on the FPGA as a PUF, named Flip-flop or D flip-flop PUFs [113], in the same way as an SRAM PUF. This is due to manufacturing variations. When the IC is powered up, the output state of each flip-flop has a random value; hence, it can be zero or one. The experimental results found that the amount of randomness present in the power-up values of the flip-flops is limited, so power-up bits cannot be used directly. So, to increase the quality of responses, post-processing is required [113]. The main advantage of this design is that it is easily spread over an IC and it is challenging to locate it, so it is robust against a reverse-engineering attack.

As we have seen, SRAM and Flip-flop PUF require being powered-up to generate the response bits. This means the cells of these two PUFs should be repowered whenever the responses are needed. Furthermore, Flip-Flop PUF requires some extra processing to extract uniform randomness.

Unlike SRAM and flip-flop PUFs, **(author?)** introduced the Latch, or SR-Latch PUF [114], which generates the response when its input is simultaneously enabled. The SR-Latch PUF consists of two cross-coupled NOR gates. Using the metastable value of these gates, LPUF can generate responses without an actual device power-up. As shown in Fig. 3.24, when the input is triggered with the rising edge, the SR-Latch starts oscillating and enters into a metastable state. After

a period of time, the SR-Latch stops oscillating and becomes stable. Due to the manufacturing variation, the state that the SR-Latch falls into is unknown, and it can be used as a response bit [26]. LPUF can be implemented on both ASIC and FPGA. But it is not appropriate for low-cost implementation of a PUF. Hence another approach is proposed to address this issue [26].

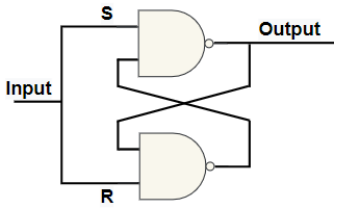

Fig. 3.24 Basic structure of SR-Latch cell [26].

In order to improve the D Flip-flop PUF, **(author?)** were the first to exploit the existing buskeeper cell as a viable alternative to the D-Flip-flop one. The big advantage over using a DFF cell for constructing a PUF is that the Buskeeper cell is minimal, and it does not require any additional circuits or processes to generate a reliable response bit [27]. As shown in Fig. 3.25, the Buskeeper or busholder PUF [27], consists of two inverters. The principle of BPUF is similar to all memory-based PUFs, where the initial patterns are read at the memory power-up. The authors' experiments prove that BPUFs have better reliability and uniqueness compared to DFF-PUFs [27].

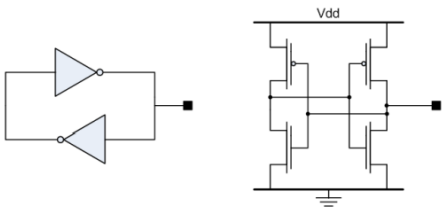

Fig. 3.25 Buskeeper cell structure [27].

- **Bistable Ring PUF:** SRAM, Butterfly, Flip-flop, and Buskeeper PUF possess an even smaller number of CRPs, which is proportional to their size. Hence, they can be used as so-called Weak PUFs. The Bistable Ring PUF [28] was the first strong memory-based PUF proposed by **(author?)** As shown in Fig. 3.27 BR-PUF consists of an even number of inverters connected to each other to build a ring. When the device is powered up, each inverter in the ring tries to force its output from an initial value of 0 to 1. For a BR-PUF of 6 inverters, the ring has two possible, stable states, 101010 or 010101. Hence, the output of the last inverter is the one-bit response generated according to which state the ring falls into, and this initial state corresponds to the response. In order to generate an

exponential number of CRPs, they proposed an architecture where the inverter count was duplicated to be used as a strong PUF. BR PUFs can be vulnerable to modeling attacks [115, 116].

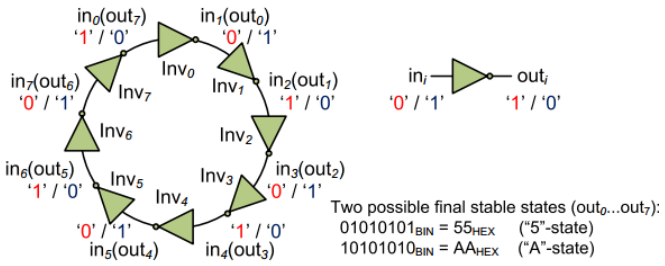

Fig. 3.26 Two possible stable states of an eight-stage bistable ring [28].

3.1.1.3 Analog electronic PUFs

The response generated by the analog electronic PUFs depends on the analog movements of the electronic components such as resistance and capacitance [68]. This type of PUF mainly includes:

- **ICID PUF:** Integrated Circuit IDentification (ICID) was proposed by **(author?)** [29]. It consists of some transistors with identical designs arranged in an addressable array. Each addressed transistor drives a resistive load due to the voltage thresholds, a random placement function of the doping atoms in the impurities of the silicon channels. The voltage on the load is measured and converted into a bit response where the challenge is the number of the transistor component. Fig. 3.27 shows a block diagram of the ICID PUF.

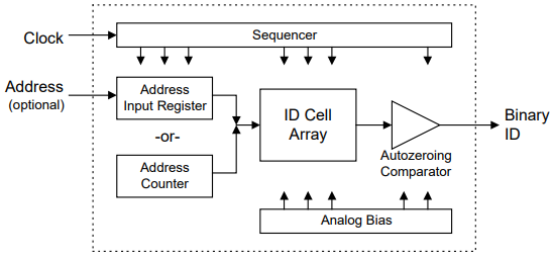

Fig. 3.27 Block diagram of ICID PUF [29].

- **Coating PUF:** Based on the idea "thou shalt not store secret keys in digital memory", **(author?)** introduced the first Coating PUF [30] using the randomness contained in the protective coating of an IC that is introduced during the manufacturing process. They drive the key, which could be used as the device's fingerprints. This is depicted in Fig. 3.28. The proposed experimental security evaluation says that the proposed PUF is safe from physical attacks.

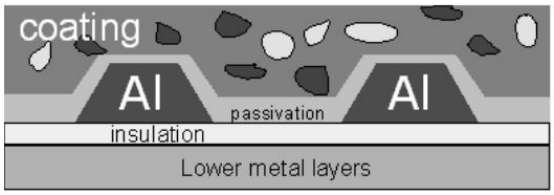

Fig. 3.28 Schematic cross-section of a Coating PUF IC [30].

- **Power grid PUF:** Since the voltage drops and the equivalent resistances are affected by random variations in the manufacturing process, **(author?)** have introduced a new PUF, called the Power Grid or Power Distribution PUF [117], which is based on the resistance variations in the electrical network of an IC. PG-PUF is susceptible to machine learning attacks [118]. Fig. 3.29 shows a circuit for the generation of the response using a power grid.

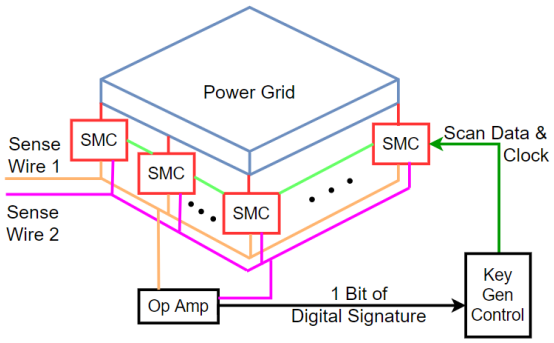

Fig. 3.29 Circuit for generation of the signature using power grid [31].

3.1.2 Silicon PUF Applications

PUFs have been used in a wide range of applications to secure devices depending on the PUF class (weak or strong) of the embedded chip within the device. This section surveys the existing application areas and use cases of Silicon PUFs that are illustrated in Fig. 3.30. Two applications were widely found: Secret key generation and authentication.

3.1.2.1 Authentication Protocols

One of the main objectives of any security system is to achieve robust authentication, which refers to verifying the device's identities and preventing malicious ones from accessing a trusted area or a network. However, numerous works have been proposed in the literature demonstrating various PUF-based authentication protocol schemes.Before surveying these works, we present a basic scheme for achieving authentication between a server and a device equipped with a PUF chip.

Fig. 3.33 depicts a conceptual PUF-based authentication process between a device

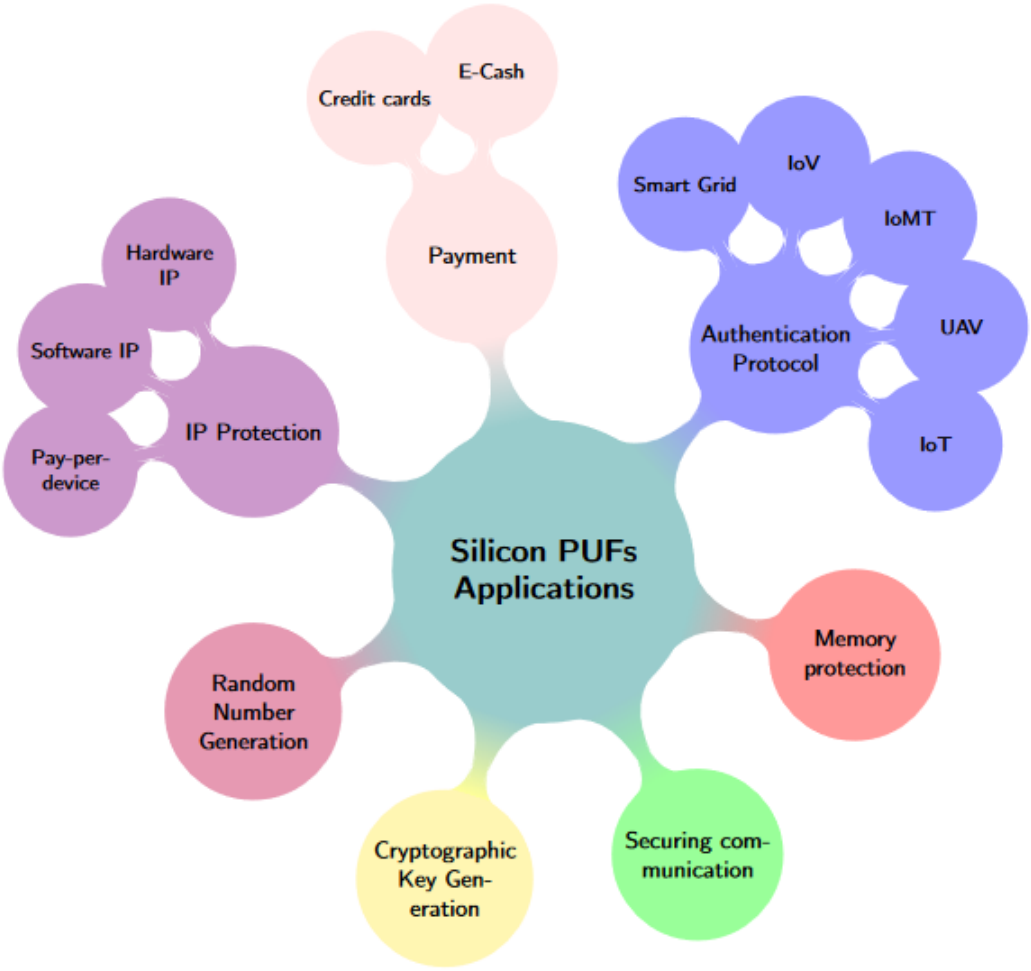

Fig. 3.30 Silicon PUFs Applications Areas and Use Cases [5].

equipped with a PUF and a trusted server. PUF-based authentication protocols can be accomplished in two distinct phases. Firstly, during the enrolment phase, the server has access to the IoT device to apply a set of random challenges and then stores their corresponding sets of responses that are extracted from the PUF circuit integrated with the IoT device. The second phase is verification, in which the device verifies the identity of the IoT device. Next, the server randomly selects from its CRP database a challenge that has never been used. Then, the IoT device generates its corresponding response and sends it back to the server. If the response from the server side matches the one that was stored for the challenge that was used, then the IoT device is real and can connect to the IoT network.

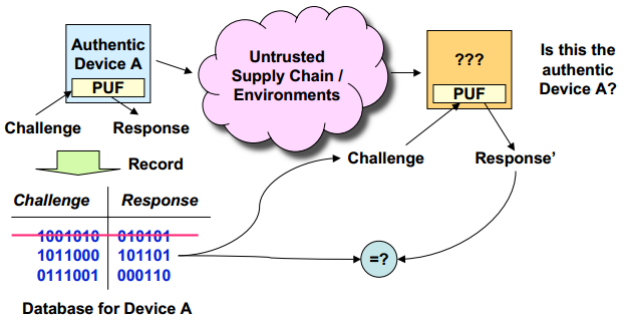

Fig. 3.31 A PUF-based Authentications Protocol Overview [32].

Over the last decade, a considerable amount of research has been conducted in the PUF-based authentication field. These protocols use a variety of silicon PUF types and different authentication mechanisms and aim to provide a lightweight and secure authentication scheme under various settings.

- **Internet of Things (IoT): (author?)** [33] proposed a lightweight PUF-based protocol that offers mutual authentication for IoT devices. Instead of storing the generated CRPs on the server, this scheme stores a so-called CRP soft model that can be obtained by performing a machine learning attack on the generated CRPs. This protocol does not ensure the reliability of communication, especially the error correction. **(author?)** [37] presented a PUF-based authentication protocol that does not offer mutual authentication, and many attacks were not considered in their scheme. However, they used a Convolutional Neural Networks (CNN) as a solution to eliminate the need for error correction mechanisms. Using elliptic curve cryptography (ECC) as a second security primitive, **(author?)** proposed a PUF-based authentication protocol [41] that does not consider noise elimination. **(author?)** [43] proposed a PUF-based authentication protocol that is vulnerable to physical attacks since the device stores an initial session secret key that will be used in the authentication phase. Rather than that, the proposed scheme does not use any noise elimination technique, making it impractical in a real appli-

cation. **(author?)** [44] proposed a mutual two-factor authentication mechanism between a device and a server, where the device is equipped with a strong and weak PUF. The first one is used for authentication and the second for encryption. This scheme does not present noise elimination, making it impractical in real applications and different environments. **(author?)** [119] presented a light-weight mutual PUF-based authentication protocol for IoT systems, including device-to-device or device-to-server communication. However, the proposed protocol does not consider error correction in the authentication steps.

- **Unmanned Aerial Vehicles (UAVs):** Nowadays, Unmanned Aerial Vehicles (UAVs) are becoming very popular due to the emergence of their areas of application: delivery, first-aid emergency, military, etc. Nevertheless, the communication between a UAV and its ground station (GS) is critical (sensitive data, weather, environmental changes, etc.) . In [39], **(author?)** proposed a mutual authentication protocol between a drone equipped with a PUF and its ground station without the support of error correction. Also, the authors do not show the details regarding the security analysis of the proposed protocol. **(author?)** [120] proposed UAV-GS and UAV-UAV PUF-based authentication mechanisms. The ground station plays an important role in the authentication phase, and it is also responsible for session key generation and delivery. The noise elimination process has not been considered, making both schemes impractical. Also, **(author?)** [121] presented mutual authentication in UAV swarm networks using PUFs. The proposed protocol uses a spanning tree protocol to identify the flow of authentication request messages in dynamic typologies and mobile UAVs.

- **Internet of Medical Things (IoMT):** For the safety of patients, PUFs have been used to secure the communication between devices, sensors and the health care monitoring system. **(author?)** [35] presented a PUF-based authentication scheme between the IoMT devices and the server, where the server is also equipped with its proper PUF. In addition, a secure database was used as a third party to store collected CRPs. However, the exchanged messages between the device and the server have not been subject to any encryption or camouflage techniques that facilitate easily launching modeling attacks. **(author?)** [122] proposed a lightweight and reliable authentication protocol for wireless medical sensor networks, that is composed of cutting-edge blockchain technology and a PUF. Also, **(author?)** [123] used a one-way cryptographic hash function and BS-PUF to ensure lightweight authentication between IoMT sensors and fog devices. **(author?)** [124] introduced a new lightweight anonymous authentication protocol for IoMT that is resilient against machine learning attacks on PUFs. To prevent various security weaknesses such as user anonymity, offline passwords, smart de-

vice theft, privileged insiders, and cloning attacks in WMSN, **(author?)** [125] proposed a three-factor-based mutual authentication scheme using PUFs.

- **Internet of Vehicles (IoV):** To guarantee the security and privacy of driving data in IoV, **(author?)** [126] introduced a secure authentication and key exchange protocol for IoV using two-factor security that combines PUFs with the user's password. The second factor is used if an advisory could hold the vehicle equipped with a PUF. Then, they added biometrics as a third factor to the same protocol [127]. In [128] vehicles and roadside units (RSU) use PUFs to authenticate themselves to the certificate authority. In this scheme, the authentication process depended on the reception of the silicon PUFs' unique fingerprint and the valid delivery certificate.

- **Smart Grid:** A smart grid (SG) can provide reliable, secure, economic, efficient, clean and high-quality electricity services. Smart meters are devices collecting data on smart grids, that can also receive instructions from the control center. However, the communication between smart meters and the control center confronts security and privacy challenges. Instead of storing a set of CRPs on the server, **(author?)** [34] proposed a PUF-based authentication protocol where only one pair of CRPs is stored on the server. The used pair is updated at the end of each successful authentication phase. This protocol is vulnerable to physical attacks since it stores secret information on the device's memory. To protect smart meters from physical attacks, **(author?)** [129] addressed the security and privacy problems in collecting metering data by proposing a lightweight privacy-preserving authenticated data collection scheme based on PUFs. In the case of fault or improper behaviour due to the high-tension power lines of the smart city, sensor nodes deployed on these lines send information to the control center to request in an emergency the recovery team. In [130], **(author?)** introduced an identity PUF-based lightweight authentication protocol for supply-line surveillance system between the sensor nodes and the control center.

3.1.2.2 Cryptographic Key Generation:

In any cryptographic primitive, it is recommended that the key must stay constant and can be reproduced several times. As silicon PUFs are a source of high randomness, their generated responses could be used as cryptographic keys in different security applications. However, the change in environmental conditions will cause noise in the output of the PUFs. This noise can cause one or more PUF output bits toggle, resulting in an incorrect and unusable key because it is not the same as the original key. Therefore, the response cannot be directly used as a cryptographic key. Hence, error correction must be used in order to tackle this issue [131]. Fuzzy Extractor (FE) and many coding

techniques for error correction are being employed in order to improve the reliability of PUFs' applicability [132].

Fuzzy Extractor (FE) [133] is designed for extracting nearly uniform random strings from noisy and non-uniform random data with high entropy. FE is built from a pair of algorithms to extract stable, reproducible information from the PUF responses; generation (*Gen*) and reproduction (*Rep*). Gen takes the initial response and outputs uniform random string data (refer to the cryptographic key) and non-secret data called public helper data. To reproduce the key from a noisy response, the reproduction algorithm, Rep, takes two inputs: the noisy response and the public helper data. The reproduction succeeds only if the initial and noisy responses are close enough. As shown in Fig. 3.32, given the same challenge c as input to the same PUF module PUF_i, in different temperatures $m_1 = 30K$ and $m_2 = 80k$, the PUF generates two different responses $R_i(c)$ and $R'_i(c)$. We consider the first response as the reference and the second one as noisy. We use the Gen procedure to generate the secret key k and the public helper data P. Then, for the reproduction of the same key, we use the Rep procedure, which takes the noisy response and P as input [134].

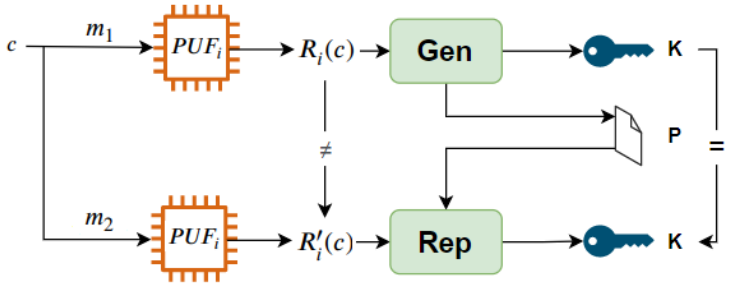

Fig. 3.32 Fuzzy Extractor.

3.1.2.3 Intellectual Property Protection

Electronic products suffer from many security challenges such as counterfeiting, cloning, reverse engineering, and the vicious addiction of components, making semiconductor companies suffer tremendous financial losses. Consequently, it is crucial to protect the intellectual property (IP) components of an IC design. **(author?)** [111] introduced a protocol for the hardware IP protection problem on FPGAs based on SRAM PUF. Also, **(author?)** [135] presented a PUF-based mechanism for firmware tempering protection to prevent the software and the hardware IP from being copied by third parties. To protect IPs from being copied, cloned, or used with unauthorized integration, **(author?)** [136] proposed PUF-based IP protection mechanism that restricts IP's execution only on specific FPGA devices, and enforces the pay-per-device licensing. **(author?)** [137] proposed a PUF-based pay-per-device scheme for protecting IPs from attacks based on CNN models.

3.1.2.4 Random Number Generation

The fourth application of PUFs is the generation of random numbers used in cryptography as an encryption key. Pseudo-Random Number Generators (PRNGs) were not truly random since the pattern repeated itself after a certain value. In fact, the Hardware Random Number Generators (HRNGs) are used to generate a true random without any initial condition [20]. By exploiting the randomness found in the inherent nature of the silicon PUFs, it could be used as a source of random number generation. **(author?)** [138] used the PUF response as an initial seed and **(author?)** [139] combined Chua circuits with PUFs. The former are a type of chaotic system that has the ability to produce different results from a fractional change in the initial conditions. PUFs are used as a random number generation mechanism to be used in cryptographic systems.

3.1.2.5 Payment

Electronic money (e-money or e-Cash) is the digital representation of physical banknotes where authentication, encryption, privacy, and anonymity play central roles. To not steal, predict, and/or clone tokens used by a device, **(author?)** [140] introduced an e-Cash based on PUFs called PUF-Cash, where the PUFs response is used in the authentication bit-strings, encryption keys, and e-Cash token generation. In [141], **(author?)** combined Leveraging TrustZone and SRAM PUFs technology to design the architecture of trusted mobile, which can be used in e-payment schemes while guaranteeing the anonymity of the users' identities to other entities, such as banks and merchants. **(author?)** [142] proposed a credit/debit PUF equipped with a weak PUF chip responsible for secure communication, data authentication and a private key stored by a customer.

3.1.2.6 Memory protection

In this field, the PUF's output is used to secure program execution by protecting the confidentiality and integrity of the memory instructions and the stored data against physical and software attacks [143].

3.1.2.7 Software licensing

Software licensing is a way to protect software from unauthorized modifications and from running on unauthorized platforms. To achieve this protection, many approaches were proposed such as the use of a hash function or checksum, where each block code has its own hash value to be checked in the next bloc by verifying the integrity of the first one. Other solutions include the use of the obfuscation method to protect software from reverse engineering and malicious modification. Meanwhile, software protection-based PUF has been proposed. The idea is to provide software with the possibility to

communicate with the PUF to perform some operations based on the generated keys where both static and dynamic PUF are used with more intention on the dynamic one [144]. **(author?)** [145] proposed a software licensing mechanism based on SRAM PUF. In this scheme, the user's PC is equipped with the SRAM PUF, giving it a unique identity. When the user needs to buy a needed software, a company will initiate a connection with the user's PC to have the SRAM PUF's outputs and make them available in the software as a license. The customer installs the software, and during installation, an authentication mechanism occurs between the software and the PC, where the embedded license is compared to the SRAM PUF's outputs. **(author?)** [146] combined self-checksumming code techniques with PUFs to establish hardware-assisted software protection. The self-checksumming code is used to check the program's integrity and protect it against tampering attacks. Then, PUFs guarantee the execution of the software instance only on the specific device (hardware) equipped with the right PUF.

3.1.2.8 Securing communication

As we presented before, PUF is widely used in the authentication protocol, especially when launching communication in many use cases. Another application of silicon PUF is communication, where the generated response will be used to guarantee secure communication by ensuring the confidentiality and integrity as well as non-repudiation of the exchanged messages.**(author?)** [36] proposed a PUF-based key-exchange protocol between IoT devices without the need for a trusted entity. After a successful authentication phase, both devices use PUF data to construct and exchange the session key to secure the communication. Also, **(author?)** [147] used Elliptic Curve Cryptography and PUFs to secure device-to-device communication. The registration centre is responsible for authentication and session key generation in this scheme. However, error corrections have not been considered in this mechanism.

3.1.3 Comparison

3.1.3.1 Architectures Comparison

Table 3.1 classifies the surveyed PUF schemes into their classes in terms of their strength (weak'W' or strong 'S'), performance (uniqueness and reliability), and resistance to different attacks. Based on this classification, we observe that.

- Arbiter PUF [6] is one of the most used PUF, and its improved architectures [95, 16, 75, 9, 7, 96, 11, 12, 97, 14, 15] achieve good performance in the two well-defined quantitative metrics: uniqueness (≈ 50) and reliability (≈ 100) (see Table 2.1 for more details). However, they do not perform well in other equally important metrics, especially security which is the most important metric that

Table 3.1 Comparison of Silicon PUFs Architectures.

Class	Scheme	W/S	Year	Uniqueness(%)	Reliability(%)	Attacks
Delay-based	APUF [6]	s	2004	23	99.76	[74, 99]
	FF APUF [95]	s	2004	38	90.16	[74, 99]
	XOR APUF [16]	s	2007	46.15	99.52	[74, 99]
	FFXOR PUF [9]	s	2020	≈ 50	89	-
	R-APUF [15]	s	2019	-	94.74	-
	LSPUF [7]	s	2008	46.16	92.32	[98]
	$m-n$ APUF [96]	s	2014	≈ 50	-	[10]
	MPUF [11]	s	2017	50	99.70	[101]
	rMPUF [11]	s	2017	50	99.55	[101]
	cMPUF [11]	s	2017	49.99	99.68	[101]
	MPUF [12]	s	2018	40.60	-	[99]
	CPUF [97]	s	2014	49.04	97.48	[98]
	IPUF [14]	s	2019	≈ 50	≈ 100	[103]
	RO-PUF [16]	w	2007	46.15	99.52	[74]
	CRO-PUF [17]	w	2011	47.31	99.14	-
	TERO PUFs [20]	w	2017	49.65	96.32	-
	DD-PUF [21]	w	2021	49.48	98.33	-
	RPPUF [13]	S	2021	45.80	99.23	-
	FR-PUF [18]	w	2014	46.88	-	[105]
	k-sum PUF [106]	s	2010	-	-	[107]
	G-PUF [22]	s	2010	41.50	>93.40	-
	SG-PUF [23]	s	2012	35	>93	-
	IP-PUF [109]	s	2011	-	-	-
	CLK-PUF [24]	s	2013	-	-	[14, 110]
Memory-based	SRAM PUF [111]	w	2007	49.97	>88	[112]
	B-PUF [25]	w	2008	≈ 50	>96	-
	FF-PUF [113]	w	2008	≈ 50	95	-
	L-PUF [114]	w	2008	50.55	96.96	-
	Buskeeper PUF [27]	s	2012	48.27	80.98	-
	BR-PUF[28]	s	2011	≈ 50	97.81	[115, 116]
Analog electronic	ICID-PUF [29]	w	2000	49	>95	-
	C-PUF [30]	w	2006	≈ 50	>88	-
	PG-PUF [117]	w	2009	-	-	[118]

determines its acceptability in real-life systems. Further, since a relationship between the challenge and the signal propagation time of the arbiter PUF can be represented as the linear model, exploiting this weakness, APUFs are vulnerable to many types of modeling attacks such as machine learning and deep learning attacks. However, strong silicon PUF is suitable for authentication by using many CRPs. Even though APUF is simple and easy to implement, its production process is precise, while the lines must be of the same length.

- Ring-Oscillator PUF [16, 17, 18, 106] is another widely used daily based PUF due to the simplicity of its design and ease of CRP extraction. However, the path between the oscillators and the counters should be exactly the same. As it is classified as a weak PUF, it is suitable for secret key generation, but is also vulnerable to modeling attacks. Compared with other daily-based PUF, RO PUFs are more considerable and consume more power, but provide higher reliability.

- Glitch PUF [22, 23] is predominant compared to other delay-based PUFs in terms of resistance against modeling attacks. It is suitable for secret key generation, but its design and glitch acquisition process are crucial.

- SRAM PUFs [111, 25, 113, 114, 27] are one of the most popular weak PUFs. Due to their simplicity and intrinsic categories, they do not require any extra hardware. However, as they have a restricted number of CRP, they are suitable for secret key generation and are widely used for identification. Compared with other PUFs, SRAM PUFs are sensitive to environmental conditions such as temperature and voltage. Therefore, error correction techniques are vital to moderate these impacts and provide reliable keys. SRAM PUFs are secure against modeling attacks, but are more susceptible to cloning attacks and invasive attacks in general.

- Bistable Ring PUF [28] is a strong memory-based PUF suitable for authentication. The BR PUF has a good uniqueness and reliability, and generally, it is reliable against aging, but it is also vulnerable to modeling attacks.

- ICID PUF or VT PUF [29] has limited IDs and fewer CRPs. Thus, it is suitable for secret key generation and identification. It is cheap and small in size, but vulnerable to cloning attacks, and It needs a particular design.

- Coating PUF [30] is suitable for secret key generation, identification, and for detecting physical tampering. It is small, fast, and cheap, but it needs a special design.

- Power Grid PUF [117] is suitable for secret key generation. It needs a special design, and it is vulnerable to cloning attacks.

We can resume that:

1. Delay-based PUFs are a class based on frequency variations or digital race conditions to generate PUF responses within integrated circuits (ICs) resulting from manufacturing variations. Several delay-based PUFs are made of arbiter PUF as a basic element. All delay-based PUFs are extrinsic PUFs, meaning they need specific extra hardware to be used in a silicon chip. The latter needs a precise process to generate a unique and reliable response. The number of the responses of several delay-based PUF is not limited, making them suitable for authentication. Whereas delay-based PUFs are not proficient in material resources and are subject to modeling attacks, this allows an attacker to build a mathematical clone of a PUF to estimate the PUF's responses.

2. Memory-based PUFs are based on the metastable state of memory cells and unpredictable start-up values. The generation of the response is limited by the number of memory cells. So, most of the memory-based PUFs are weak and have fewer CRP. However, they are suitable for identification and secret key generation. Memory-based PUFs are intrinsic PUFs (except BPUF) because their circuits are implanted within the design itself and do not require any additional hardware.

3. Analog electronic PUFs are a class of PUFs that exploit the analog measurement of an electric component to generate a response. Analog electronic PUFs are more suitable for integrated circuit identification and physical tampering. Generally, they are represented by power grid PUFs and coating PUFs. Analog electronic PUFs are vulnerable to cloning attacks.

3.1.3.2 Applications Comparison

Table 3.2 classifies and compares the surveyed contributions related to silicon PUF-based applications. We consider different criteria, including the area of application and the specification of use cases. We also show if the surveyed work relies only on silicon PUFs or uses other security primitives like hashing and cryptographic functions. Also, we consider the integration of noise cleaning and error correction. Further, PUF implementation is verified by checking if the proposed work indicates the architecture of the used PUFs. Based on this comparison, PUFs are used in many applications and fields, from cryptographic key generation to e-payment. Most of the proposed work in different application areas uses extra security primitives to achieve their objectives such as the hashing function and XORing operation. Also, sometimes PUFs are combined with elliptic curve cryptography or blockchains. From the reliability side, most of the discussed works do not consider error correction and noise elimination process in their

proposed scheme, making their solution impractical in any application and use case area since the PUFs reliability is considered as a principal metric that could gauge the efficiency of any proposed PUFs based scheme. Also, we observed that most of the discussed works do not indicate the architecture of the deployed PUF in their proposed scheme.

3.1.4 Summary

In this first section, we surveyed the state-of-the-art silicon PUF architectures and classified them into delay-based PUFs, memory-based PUFs, and analog electronic PUFs. Additionally, we listed the most existing implementations for each class and explained their operating processes with graphical representations. Furthermore, we have also surveyed, classified and compared the existing Silicon PUF applications and use cases. The next section is fundamental to our project since it will be dedicated to presenting the exciting related work, where a review of PUF-based authentication protocols is presented and compared.

Table 3.2 Comparison of the Applications and Use Cases of Silicon PUF.

Application	Use case	Works	Year	Only PUFs	Error	Type PUF
Authentication	IoT	[33]	2021	No	No	Delay-based PUF
		[37]	2021	Yes	No	DRAM PUF
		[41]	2020	No	No	-
		[43]	2018	No	No	-
		[44]	2020	No	No	APUF+SRAMPUF
		[119]	2017	No	No	-
	UAV	[39]	2020	No	No	-
		[120]	2020	No	No	-
		[121]	2021	No	No	-
	IoMT	[35]	2019	No	No	Arbiter PUF
		[122]	2021	No	yes	-
		[123]	2021	No	No	BS-PUF
		[124]	2021	-	-	-
		[125]	2021	No	Yes	-
	IoV	[126]	2019	No	Yes	-
		[127]	2021	No	Yes	-
		[128]	2021	No	No	-
	Smart grid	[34]	2020	-	-	-
		[129]	2021	No	Yes	-
		[130]	2021	No	No	-
IP Protection	Hardware IP	[111]	2007	-	Yes	SRAM PUF
	Software IP	[135]	2014	-	No	drPUFs
	Pay-per-device	[136]	2015	-	No	Delay-based PUF
		[137]	2018	-	-	-
Payment	e-Cash	[140]	2019	-	No	HELP [148]
		[141]	2016	No	yes	SRAM PUF
	Credit cards	[142]	2017	-	No	-
Licensing	Software	[144]	2019	No	No	-
		[145]	2018	No	Yes	SRAM PUF
		[146]	2015	No	Yes	SRAM PUF
Securing communication	IoT	[36]	2021	No	Yes	-
		[147]	2021	No	No	-
Memory	-	[143]	2007	No	No	RO PUF

3.2 IoT PUF-Based Authentication Protocols

Indeed, IoT applications have a wide range of advantages, including making our life easier, more intelligent, more convenient, and also personalized by altering the way we carry out our daily tasks. However, despite all of the benefits of IoT, these devices face several security issues, and hazards that must be continuously managed and maintained, including authentication, privacy, access control, and data collection and management [50]. Practically, wireless technology is the most used means of communication by IoT devices, and a secure network communication system requires a robust authentication protocol providing secure transmission sessions.

The authentication of IoT devices refers to verifying the device identity and preventing malicious devices from accessing the IoT network. However, this process is considered an essential requirement and the main security challenge in any IoT system. Unfortunately, any weakness in the authentication process will afford a compromised IoT device to get access to the IoT trusted network and to establish unauthorized communication, inject false data, get access to confidential data, and launch dangerous attacks. Hence, secure transmission protects data during the transmission process by guarantying integrity, confidentiality and non-repudiation of the data using different encryption schemes [149].

3.2.1 Authentication in IoT Devices

Mutual authentication must be secure to protect the privacy and confidentiality of data sent between a device and a server. Numerous authentication schemes have been developed for IoT devices. The three primary ones are as follows : (1) the encryption/decryption cryptography authentication technique, in which an encryption/decryption algorithm is used for authentication; (2) the localization and device environmental data authentication technique, in which the placement and surrounding information of the IoT device can be used to authenticate it; and (3) the PUF authentication technique, which uses the intrinsic characteristics of the IC in the IoT device [44]. In the subsections that follow, we discuss briefly each strategy.

3.2.1.1 Authentication via Encryption/Decryption

Either symmetric or asymmetric key cryptography can be used to authenticate a device in the IoT system. In symmetric key authentication, both the IoT device and the server share and securely store a secret key. Advanced Encryption Standard (AES) is the most prevalent algorithm used in symmetric key encryption [150]. In asymmetric key authentication, both the IoT device and the server employ a pair of public and private keys, where only the public key is shared. The private key is used to generate

signatures, whereas the public key is used to verify signatures. Both symmetric and asymmetric approaches need the safe storage of a key on the device. Numerous approaches for employing and securing the device's secret cryptographic key have been devised [151, 152]. There are credential-specific software solutions, for instance, that may be configured to manage the authentication credentials of a device. Other solutions utilize physical components such as Central Processing Units (CPUs) or integrated circuits (ICs) with memory storage to store the secret keys in their memory, which is always powered. Storing cryptographic keys on the IoT device is inherently vulnerable to certain types of attacks, such as firmware assaults, in which the attacker physically or remotely connects to the device and reveals the stored key.

3.2.1.2 Authentication via Localization and Device Environmental Data

It is possible to authenticate an IoT device by finding it or by having information about its neighboring devices or the characteristics of its communication channel. The server can collect information about the IoT device's location and its surrounding environment, such as arrival time and/or signal strength, examine the collected data, and authenticate the IoT device accordingly. Different technologies, such as the Global Positioning System (GPS) or a base radio station in a Wireless Local Area Network (WLAN), can be used to localize the device [153]. This technique can also be utilized with a smart phone. After the server registers the device's identity information, location information, and permission policies, the protocol executes the location-based authentication and authorisation procedure each time the device requests access to a resource or service.

3.2.1.3 Authentication via Physical Unclonable Functions

PUFs ensure secure authentication without storing the secret key on the device. An IoT device can be authenticated using its PUF component since PUFs generate unique information from the physical characteristics of an IC on demand. This uniqueness of the output is used as a unique identity of the IoT device and its authentication.

3.2.2 Basic PUF-Authentication protocol

As shown in Fig. 3.33, PUF-based authentication protocols can be accomplished through two distinct phases. Firstly, during the enrolment phase, the server has secure access to the IoT device, applies a set of challenges, and then stores their corresponding responses extracted from the PUF circuit integrated within the device. The second phase is verification, in which the server verifies the identity of the IoT thing. Then, the server randomly selects from its CRP database a challenge that has never been used. Then, the IoT device generates its corresponding response and sends it back to the server. If the received response from the server-side matches the stored one corresponding to the used

challenge, then the IoT thing is authenticated and can have access to the IoT network. Otherwise, the authentication fails.

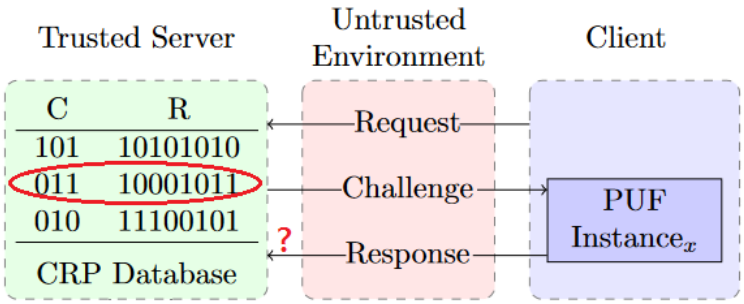

Fig. 3.33 A PUF-based Authentications Protocol Overview.

3.2.3 Requirements

Compared with the existing authentication protocols mechanisms, PUF-based authentication protocol have been proposed to ensure the device identity without storing any secret information on the device's memory. However, any PUF-based authentication protocol should respect the following requirements [154]:

- *Complete Specification:* A protocol should be specified in a complete description including textual description, graphical representation, and clearly detailing all authentication steps, especially exchanged messages.
- *Leakage Resilience:* A PUF-based protocol should be resilient against local memory information leakage. Ideally, it has to ensure a secure authentication mechanism without storing sensible and secret information on the device memory, which avoid attacks such as the physical ones.
- *Able to Handle Noisiness:* Noise elimination found in the PUF responses should be taken into account. This makes the possibility to use the PUF response as cryptographic keys in different environments.
- *Counteracting Strong PUF Modeling Attacks:* As showed before, strong PUF is used in authentication protocol. However, it is also vulnerable to machine learning attacks. So far, this attack should be always taken in the security analysis of the protocol, since it can be used to anticipate the response of a given challenge. This is achieved by building a soft model of the PUF.
- *Strong PUF Response Space Expansion:* All PUF-based authentication protocols use CRPs as the principal data. So to avoid brute-force and random guessing attacks, the used PUF design should to be able to generate a huge number of CRPs.

- *Low-Cost and Resource-Constrained:* Since PUF is considered as a lightweight security primitive, any PUF-based authentication protocol has to guarantee a low-cost and resource-constrained by minimizing the processing and storage operations.

- *Resistance against Protocol Attacks:* The PUF-based authentication protocol have to resist to a conventional protocol attacks, such as DOS attacks.

- *Identification Prior to Authentication:* PUFs provide a unique identity, so before use it as authentication mechanism, it guarantees always the devices identification. Unfortunately, identification prior authentication is more practical.

- *On the Mutual Authentication Order:* For more security, it is a good practice to check and verify the authenticity of the server first.

3.2.4 Attacks

In this section, we present a brief definition of the common existing attacks that can be launched upon a PUF-based authentication protocol [34, 155, 44].

- *Message Analysis Attack:* This kind of attack concerns the confidentiality of the exchanged message during or after a successful authentication phase. Meaning that an adversary tries to obtain the transmitted information between the communication entities.

- *Replay attack:* In this attack, the adversaries store the transmitted information of the valid authentication operation with the hope to utilize them in future authentication.

- *DoS attack:* The goal of adversaries in this attack is to exploit all the resources of the target device, and/or to temporarily or indefinitely disrupting services. It is accomplished by flooding the targeted device with superfluous authentication requests.

- *Physical attack:* In this kind of attack, an adversary may attempt to physically get access to a device and tries to obtain the secret information stored on its local memory.

- *MITM attack:* In this attack, an adversary tries to intercepts the exchanged information between different entities during their authentication process. It possibly also alternating the communications between them and makes them believe that they are directly communicating with each other.

- *Injection attack:* The goal of the adversary in this type of attack is to inject and change the transmitted messages between authenticated devices, or even create its malicious message and inject it during the communication steps.

- *Helper data manipulation:* In this attack, an adversary tries to manipulate and modify the helper data, stored on the device memory using physical attacks or manipulated during the transmission process using an injection attack. Any manipulation makes the fuzzy extractor operation impossible.

- *Modeling attack:* From the communication between the server and the device, an adversary tries to get CRPs of the used PUF. Then, it executes the machines learning techniques on the CRPs dataset and offers to the adversary the possibility of predicting the correct response related to a given challenge.

3.2.5 Existing Literature

In this section, we review the recently published IoT PUF-based authentication protocols (during the last five years, since 2016). For each contribution, we describe the steps of the developed protocol including its schema, and discuss its analysis techniques if exist. Also, we explain its strengths, and its security level and weaknesses. Further, we show each protocol's experimentation and applications.

Strong PUFs are vulnerable to modeling attacks, where adversaries collect the exchanged CRPs that are used in the authentication sessions and apply machine learning algorithms to produce a soft model of the PUF circuit. Also, using a hashing function to counter this attack by encrypting the exchanged information (responses) is expensive in terms of computation [33]. **(author?)** proposed a lightweight PUF-based protocol that offers mutual authentication for constrained devices on IoT networks (Fig. 3.34) [33].

Instead of storing the CRPs on the server, they save the CRPs trained soft model while except challenges are exchanged during the authentication process. In this case, it is considered as a 'pseudo' challenge since it is never used as direct input to the PUF circuit. By using a predefined challenge transformation functions, pseudo challenges are dynamically transformed for being fed to the PUF circuit. Thus, the authentication process run in three steps. First, each IoT device and server generate an (n/2)-bit pseudo challenge and exchange them (exchanging C11 and C12). Secondly, the device uses the PUF to generate the response of the transformed challenge, as well as for the server, but by using a soft computing PUF model. Finally, based on the generated response, the device generates two random variables and sends them to the server. If they satisfy the generated response from the soft model, so the device is authenticated. In fact, this protocol develops and uses extensive computational functions with respect to the motivation of this work that targets lightweight protocols. The trade-off between

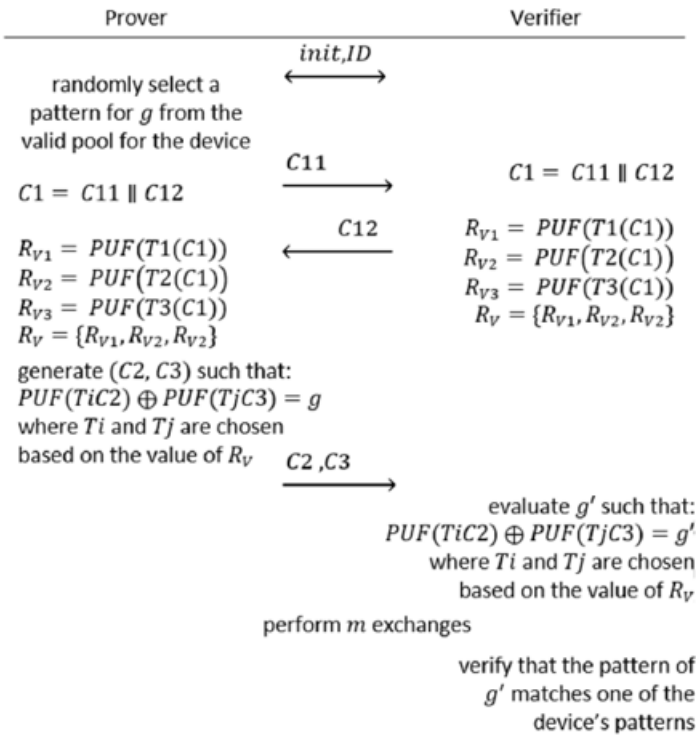

Fig. 3.34 Pseudo challenges based PUF authentication protocol [33].

energy consumption and computational functions should be token into account especially in this category of protocols. Moreover, the protocol does not ensure the devices anonymity and either the communication reliability especially error correction.

Following the same idea of using PUF without storing many CRPs on the server, Kaveh *et al.* [34] proposed a two-way authentication protocol (Fig. 3.35) by relying on the PUFs of smart meters (device) and their neighborhood gateways (server) in smartgrids.

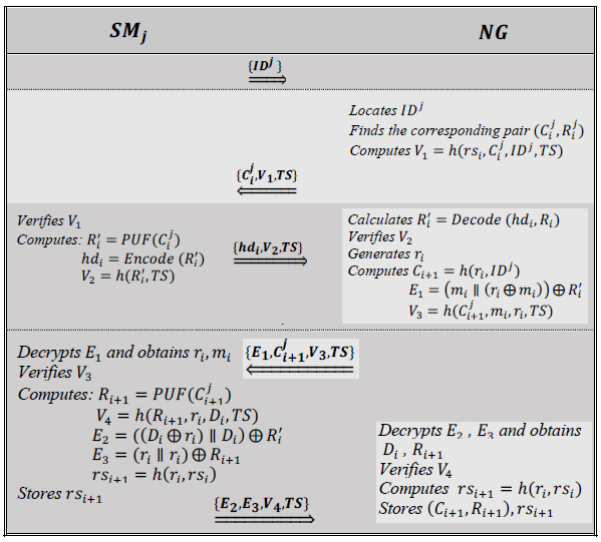

Fig. 3.35 Two-way Authentication Protocol [34].

The developed authentication mechanism goes through two phases: offline and online. In the former, the first CRP is generated by the smart meter and stored on the neighborhood gateway. Thus, for its first authentication, the initial stored CRP is used. Then, each successful authentication phase is ended by the generation and storage of a new CRP. Unfortunately, the protocol is considered as incomplete since the initial response is needed to decrypt some received information from the server. In addition, the response is stored on the device, rather than other variables that are classified as secret. This protocol do not satisfy the PUF principles that accomplish security primitives without storing keys. This protocol could be improved by resuming some unused variables, and enhancing its reliability and assuring the anonymity.

(author?) [35] presented a PUF-based authentication scheme for the Internet of Medical Things (IoMT) where both the server and the IoMT are equipped with a PUF (Fig. 3.36). Instead of storing the CRPs on the server, they are only stored in a third entity (secure database).

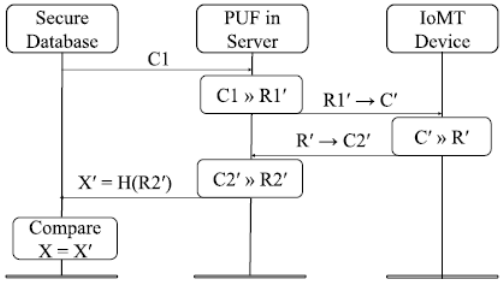

Fig. 3.36 PUF-based authentication scheme for IoMT [35].

The authentication process is performed in three steps. First, after the authentication request, the database selects a challenge and sends it to the server, which uses its proper PUF to generate the response. Then, the response is passed to the IoMT device as a challenge, after which the device generates a response to be sent back to the server as a new challenge. Finally, the server generates a corresponding response of the received challenge and sends it to the database to compare it with the stored one and decide the authenticity of the IoMT device. The exchanged challenge and the response between the server and the IoMT device has not been subject to any encryption or camouflage techniques which facilitates easily launching modeling attacks. Also, no security analysis has been presented, rather than, error correction has not been token into consideration. Thus, the authentication could never succeed as the database compares the hash of the new noisy response with the initial stored one.

Compared to database-driven or server-based IoT authentication approaches, **(author?)** presented a Peer-to-Peer (P2P) IoT connection [36]. They proposed a lightweight PUF-based mutual authentication and a key-exchange protocol between IoT devices without the need of third entities (trusted server or secure database) (Fig. 3.37).

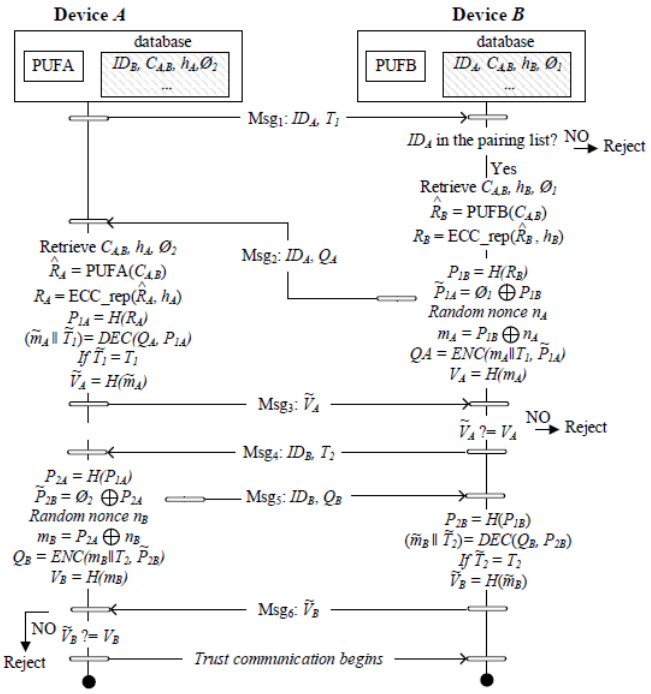

Fig. 3.37 IoT P2P PUF-based Protocol [36].

First, both IoT devices (A,B) go through the enrolment phase with a trusted server. From the CRPs of both devices, the server generates useful information to be stored on the memory of both devices that will be used later in the authentication phase. Let's take an example for the device A (the shared challenge $C_{A,B}$, Identity of B ID_B, the helper data of A h_A, and a secret variable Φ that is the result of xoring the hash of both responses of A and B when applying the same challenge $C_{A,B}$). In the authentication phase, when A wants to communicate with B, first B checks the existence of ID_A in its local database. When found, to decrypt Φ, its PUF is used to generate the response of the stored challenge $C_{A,B}$. This phase is to retrieve the response of A and use it to encrypt a message M to be communicated to A. The latter uses its PUF to generate the response of the stored challenge $C_{A,B}$ with the stored helper data h_A to eliminate the noise on this response and use the result to decrypt the received encryption information M. Then, the device A sends the hash of *M* to B and compares it with *M*. Finally, both devices exchange the session key that is used to secure the communication between A and B. In fact, the protocol was designed for low-cost and resource-limited devices, but awkwardly it needs to store four variables for each device that wants to establish a connection with it. If A wants to communicate with N devices, it must store N $\times$ 4 variables in its local memory. Some of this information is classified as private, which make them vulnerable to physical attack. Practically, this protocol is not easy to be deployed because if a new device C wants to communicate with a device A, which is already in service. First, the device A and C have to pass through the enrolment phase to receive the needed information from the trusted server, that help them to communicate

in a secure way.

As evidence, by applying several times the same challenge as input on a PUF, the generated response has to remain unchanged. Inadequately, due to the aging and the environment variations, the response could be noisy and differs from the original one with some error bits. To overcome this issue, error correction techniques such as fuzzy extractors have to be applied on the noisy output of the PUF. However, this solutions cause additional costs and cannot be efficiently implemented in a constrained IoT device [51, 37]. **(author?)** [37] presented a device authentication protocol without using error-correction codes by relying on a deep CNN as an alternative solution (Fig. 3.38).

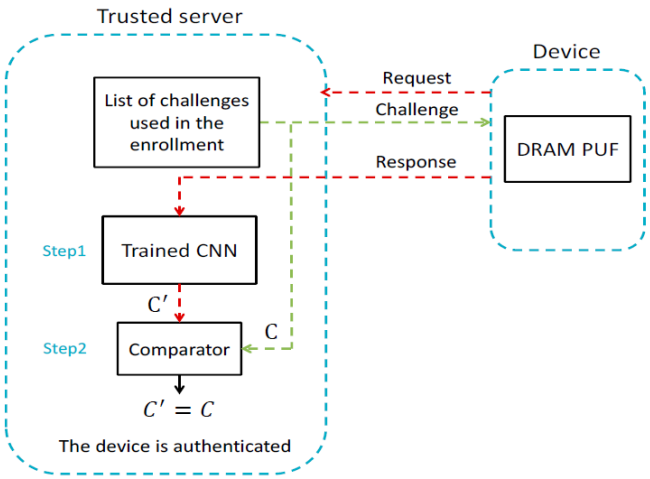

Fig. 3.38 CNN-PUF based Authentication Protocol [37].

During the enrollment phase of a device, its CRP dataset is created then the CNN is used to extract the shared features and the failure patterns of the generated responses of the dataset. In the authentication phase, the server sends one of the stored challenge to the device, and the latter generates its associated response to be sent back to the server. Afterward, the response is received as a raw of bits and identified by the CNN structured classes during the enrollment phase. If the class label in which the response was categorized matches the used challenge, the device will be authenticated. Therefore, they are several disadvantages related to this scheme. First, it does not offer mutual authentication, and does not support the anonymity. In addition, many attacks are not considered such as modeling, man-in-the-middle and DoS attacks.

Taking the scenario, when a user wants to access and manage an IoT device in the network. First, he/she needs to authenticate on the target network. In order to protect the integrity and the authenticity of the confidential data (password) in IoT systems, Zhenyu et al. [38] proposed an identity PUF-authentication protocol for IoT devices (Fig. 3.39) using the PUF outputs and the user password as a two-factor authentication. The proposed protocol is composed from three main entities: the user, the IoT device that should be equipped with a PUF, and the trusted server with a secure database. To

generate a response, the two-factor authentication uses the user password to create the challenge of the PUF. First, it will be combined with the user password and send it later to the device as a a challenge. With respect to the proposed scheme, no security analysis process has been developed. In addition, the user is implicated in all steps of the authentication by receiving and transmitting data from the device to the server. Ideally, the user could be interrogated only when the credential information are asked. Further, the other steps should be done directly between the server and the device without user interactions. Furthermore, the protocol does not take into consideration the constrained IoT, especially the device executes many mathematical operations which is a kind of contradiction with the PUF principles (using only CRPs).

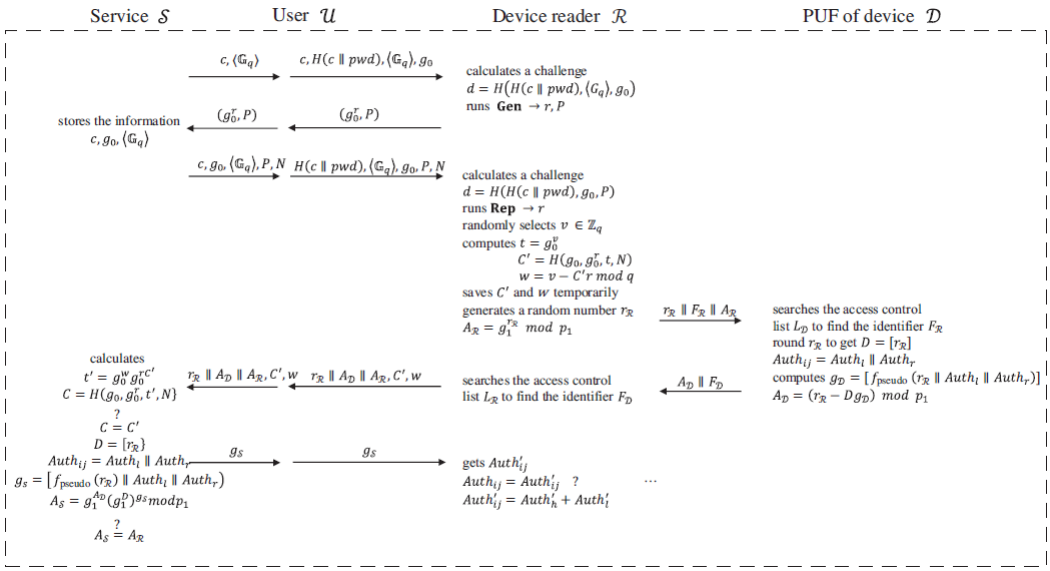

Fig. 3.39 Identity PUF-authentication Protocol [38].

The intrinsic resource constraints of aerial vehicles (UAVs) and the specific nature of wireless communication network make the traditional cryptographic techniques inefficient to secure communication, data storage, and privacy in UAVs application. Cong and Yucheng [39] proposed a lightweight mutual authentication protocol between aerial vehicles and the ground stations while each UAV has its proper PUF (Fig. 3.40). By taking a drone as an UAV example, the authentication protocol is executed as follows.

First, the drone x generates a nonce N_x and sends it to the ground station with its identity, ID_x.Then, the ground station checks the existence of ID_x in its database. When found, it generates a random number PRF_GS, and creates a new message M_x, $(ID_x|N_x|PRF_GS)$. By applying the Duffing map system and randomly selecting a CRP (C_x, R_x), it encrypts M_x and sends it to the drone with the used challenge C_x in addition to the message authentication code (MAC) to guarantee the integrity of M_x and C_x. When received, the drone generates the response of the received challenge and used it to decrypt M_x and checks the validity of the MAC value. Then, it generates

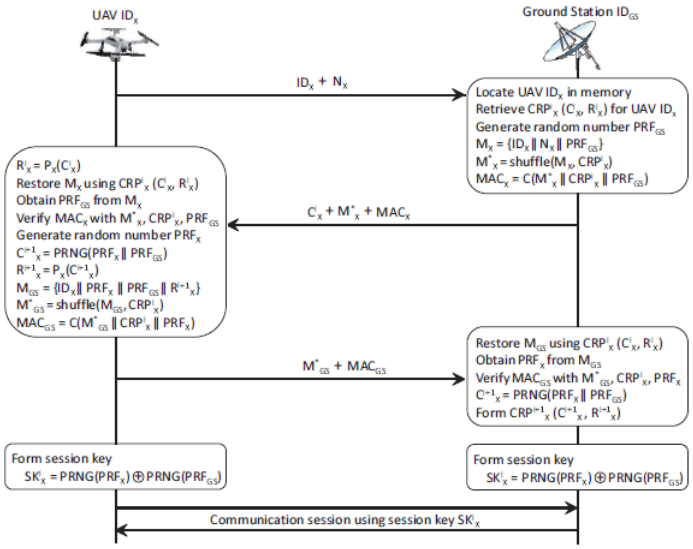

Fig. 3.40 UAV-PUF based Protocol [39].

a random number PRF_x and combines it with PRF_GS to compute a new challenge C_{i+1}, then generates a new response of this challenge. The drone generates a new encrypted message M_gs (ID_x—N_x— PRF_x—PRF_GS—R_{i+1}) using (C_i, R_i). Finally, the ground station decrypts the received message M_gs, and recuperates PRF_x with the new response R_{i+1}, and computes its corresponding challenge C_{i+1}. Also, the new CRP(R_{i+1}, C_{i+1}) is stored for a future authentication process. At this end, both the drone and the ground station can compute the session key by Xoring PRNG(PRF_x) and PRNG(PRF_GS) values. Unfortunately this protocol is unpractical, since the error correction is not considered in an UAV environment known by its dramatic changes.In practice, this situation makes the decryption operation impossible. Further, the author does not show the details regarding Duffing map system and the security analysis of the protocol.

(author?) [40] proposed a mutual authentication between IoT devices and the server. They rely on PUFs as the main security component in addition to hashing functions and XORing operations (Fig. 3.41).

First during the registration phase, contrarily to the previously discussed protocols, the device generates itself the challenge, the corresponding response, and some random time variables to be sent to the server for a secure storage. Then, the IoT device stores some private and sensitive information. Later in the authentication phase, the server computes a set of hashing and xoring operations of the stored variables with some nonce while the result is sent back to the IoT device. The latter first checks the validity of the received information to prove the authenticity of the server, then it generates the secret key and the helper data using the fuzzy extractor. It uses this result with other stored variables and a new random one to prove its identity. Finally, the server and the IoT device compute the secret session key. This protocol is characterized by eliminating

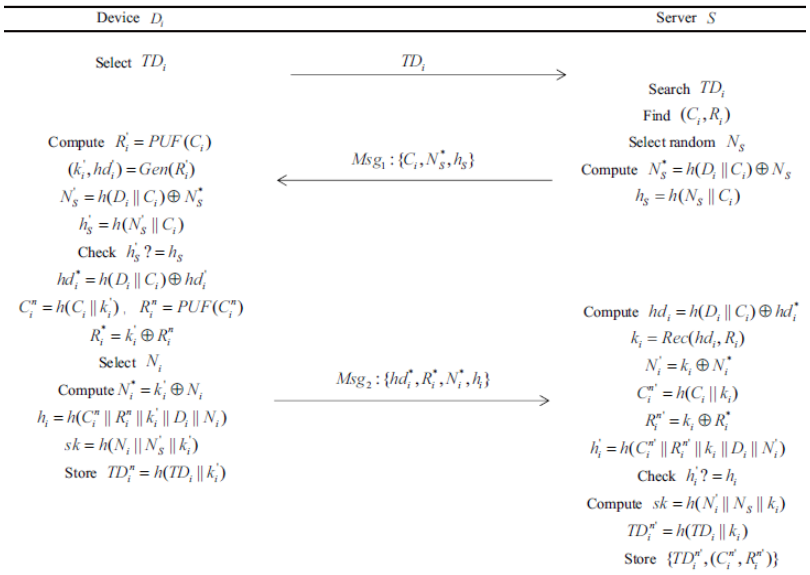

Fig. 3.41 Lightweight Authentication Protocol [40].

noise while running the generation algorithm on the device side and the reproduction on the server side. It could be interesting to show the security analysis on a real case scenario.

Since physical attacks is one of the major concerns for IoT device security, **(author?)** proposed a PUF-based authentication protocol (Fig. 3.42) called RapidAuth [41] that resists to physical attacks. Rather than using only PUFs as a security primitive by IoT device, elliptic curve cryptography (ECC) is also integrated. Before the authentication phase, the server is assumed to have the CRP of the IoT device. The server uses the stored response and ECC to encrypt some generated random variables. Then, the result of the encryption, the challenge and the MAC message are sent to the IoT device. To decrypt and check the integrity of the received information, the device generates the corresponding response of the selected challenge. Thus, after the verification of the authenticity of the server, the IoT device generates random variables and computes some data using the response and ECC, then send the result to the server within the MAC message. Finally, the server checks the identity of the IoT device using the stored response. Both IoT device and the server compute a session secret key by Xoring two generated random variables. As showed on all steps of the protocol, it does not take into consideration the noise elimination that plays a main role in the encryption and decryption steps of the proposed PUF authentication protocol. In addition, it does not guarantee the anonymity of the IoT devices.

To minimize the risk of the authentication key exposure and the load of authentication server in the IoT network. **(author?)** [42] proposed a PUF-based IoT device authentication protocol (Fig. 3.43). They follow the same strategy presented in [39] where the device stores only one CRP pair. For the authentication, the device and the

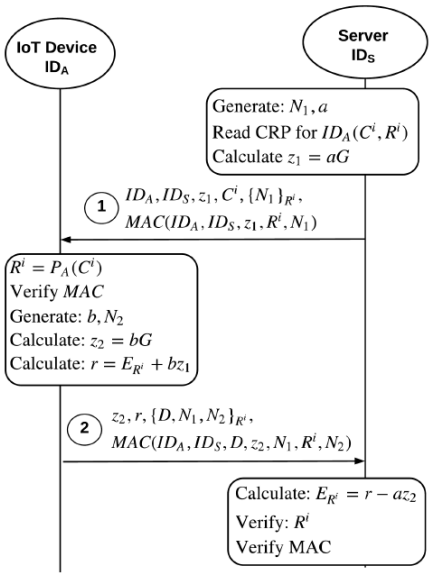

Fig. 3.42 RapidAuth Protocol [41].

server go through an offline and secure phase, where a single CRP is generated and stored in the server. When the IoT device wants to authenticate, the server challenges it. Then, the IoT device generates the response of the received challenge, and a secret key (symmetric key) by combining the response and the challenge. Also for a future authentication, the IoT device generates a new CRP, by selecting a new random challenge and generates its response. Finally, the IoT Device encrypts the first generated response and the new pair by the generated symmetric key,then sends back the result to the server. From the server side, it computes the symmetric key using the stored CRP, and decrypts the received data. It verifies the authenticity of the device by comparing the stored response and the received one. Also, it stores the new CRP for a future authentication and deletes the previously used one. Using the response as cryptographic key without correcting errors makes the application of the proposed symmetric encryption impossible since the key is generated from the responses. Also, the authors have not shown how they prove the correctness and the security of the developed protocol.

To ensure a secure communication among smart IoT devices, **(author?)**[43] proposed a PUF-based authentication protocol, called PAS (PUF Authentication Scheme, Fig. 3.44). Initially, the device and a gateway generate a session key that will be used in the authentication phase. Both of them store the session secret key $K_u - G$, the device credentials, the user's fingerprint FPu, and the device's Serial Number SNu. In the authentication phase, first the device prepares a cipher text E $K_u - G$, (SN_u, FP_u, $N1_u$, ID_u, hash (SN_u —— FP_u —— $N1_u$)), where $N1_u$ is a random nonce to be transmitted to the gateway with the ID_u. Then, the gateway decrypts the received cipher text using the stored session key, and verifying the validation and the existence of FP_u in its repository. If found, the gateway generates a challenge C_G and sends it to

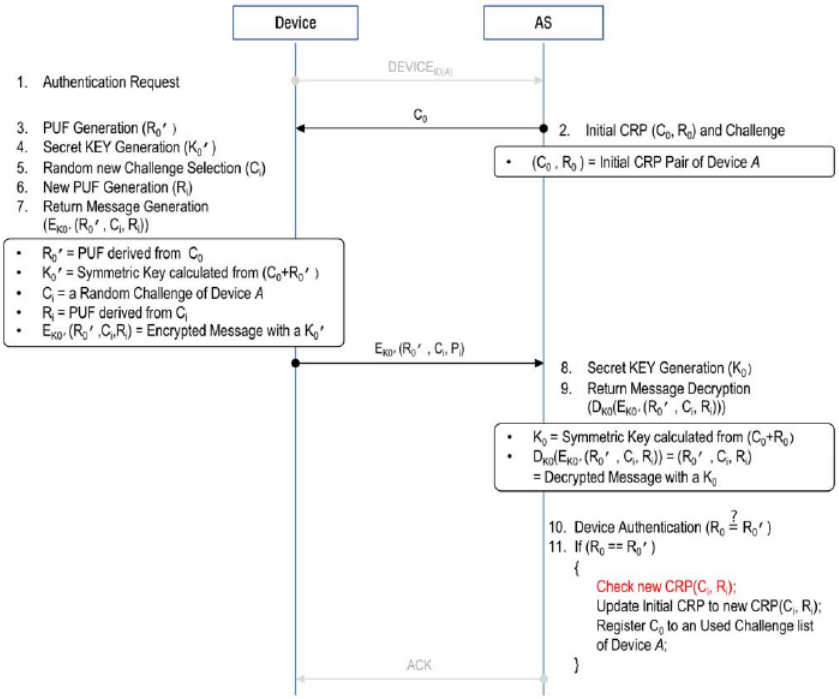

Fig. 3.43 Symmetric Key PUF based Protocol [42].

the device through a new cipher text using the same used session key. As a next step, the device decrypts the cipher text and generates the response R_u of the received challenge. Then, it generates a new secret key $Sku - G$ = h($N1_u$) xor h(FP_u) xor h(R_u). Finally, the device prepares a new cipher text using $Sku - G$, and transmits the result to the gateway. The latter decrypts the received cipher text, and checks the authenticity of the smart device by verifying the content of the decrypted text such as the response R_u. Unfortunately, the proposed protocol does not use any error correction and noise elimination technique, which make the protocol impractical. Further, the device stores the initial session key which allows physical attacks.

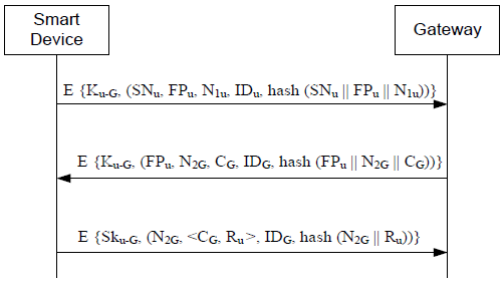

Fig. 3.44 PUF Authentication Scheme [43]

To ensure a secure communication between IoT devices and a trusted server through a public network, and also to avoid cloning and side-channel attacks, **(author?)** [44] proposed a lightweight mutual two-factor authentication mechanism between a device and a server (Fig. 3.45). The proposed scheme uses the strong PUF for the authen-

tication process whereas the weak PUF is to generate cryptographic key. First, the device generates a secret key $SRAM_k$ using the SRAM PUF, a timestamp TS_1, computes HMAC($SRAM_k$, TS_1) and sends all the result to the server with the device's identity ID_d. To check the integrity of the received data, the server uses the stored $SRAM_k$ that corresponds to the received ID_d and calculate HMAC($SRAM_k$, TS_1). Then, the server generates a timestamp TS_2, selects randomly a challenge C, and calculates HMAC($SRAM_k$, C $\|TS_2$) message. Afterward, it sends it to the IoT device with C and TS_2. Later on when the IoT device receives the server message, it checks the integrity of the received challenge C by calculating HMAC($SRAM_k$, C $\|$ TS_2), and generates the corresponding response R, the timestamp TS_3, and calculates HMAC(RAM_k R $\|T_S3$) message to be send to the server. Thus, the server calculates HMAC($SRAM_k$, R $\|TS_3$) and compares it with the received one to check the authenticity of the device. This scheme does not present the noise elimination which make it impractical in real applications and different environments.

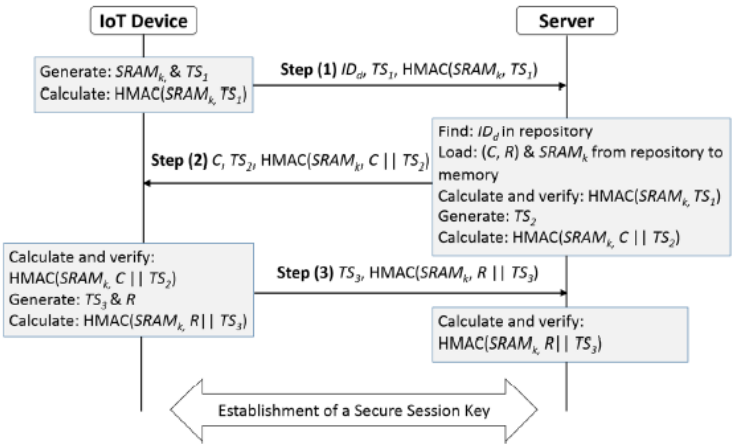

Fig. 3.45 Lightweight mutual two-factor authentication mechanism [44]

3.2.6 Comparison and Discussion

In this section, we run a comparison between the previously discussed protocols in Section 3.2.5. First, we compare the protocols in terms of their performance, strengths and portability. Then, we check how much a protocol is satisfying the known authentication protocols requirements. Finally, we look how the protocols are mitigated or designed against the existing attacks.

3.2.6.1 Performance

In the first comparison, Table 3.3 classifies the surveyed PUF-based authentication protocols that are proposed during the last five years. Indeed, for our comparison we took into consideration many criteria including the area of application. We show also that

if the protocol relies on less processing operations like Boolean logic operands (and, or, XoR, etc), or computes more complex ones, like hashing and cryptographic functions. In addition, the authentication as a main objective of PUF protocol, we look if the proposed protocols support the different authentication mechanism: mutual, device to device, device to server, works on an open-secure environments, etc. Also, we consider the integration of noise cleaning and error correction as a strong point of a protocol since it makes it applicable in different area. Further, the correctness and the soundness of the proposed protocol are verified by checking if a verification tool, an experimental process, or a simulation based approach has been ran. In Table 3.3, ✓ means the protocol has this criteria and ✗ when it does not.

Work	Year	Application	XOR operation	Hash algorithm	Fuzzy Extractor	Extra cryptographic	D2D authentication	session key generation	Practicality	Formal security analysis	Informal security analysis	Simulation
(author?) [33]	2021	IoT	✓	✗	✗	✗	✗	✗	✓	✗	✓	✓
(author?) [34]	2020	Smart Grid	✓	✓	✓	✗	✗	✗	✗	✓	✓	✓
(author?) [35]	2019	IoMT	✗	✓	✗	✗	✗	✗	✓	✗	✗	✓
(author?) [36]	2021	IoT	✓	✓	✓	✗	✓	✓	✗	✓	✗	✓
(author?) [37]	2021	IoT	✗	✗	✗	✗	✗	✗	✓	✗	✗	✓
(author?) [38]	2019	IoT	✗	✓	✓	✓	✗	✗	✓	✗	✗	✓
(author?) [39]	2020	UAV	✓	✗	✗	✓	✗	✓	✗	✗	✗	✓
(author?) [40]	2021	IoT	✓	✓	✓	✗	✗	✓	✓	✓	✗	✗
(author?) [41]	2020	IoT	✗	✗	✗	✓	✗	✗	✓	✓	✗	✗
(author?) [42]	2019	IoT	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
(author?) [43]	2018	IoT	✓	✓	✗	✗	✗	✓	✗	✗	✗	✗
(author?) [44]	2020	IoT	✗	✓	✗	✓	✗	✓	✓	✓	✗	✓

Table 3.3 Summary description of the proposed PUF-Based authentication Protocols.

3.2.6.2 Requirements Satisfiability

In a second time, Table 3.4 checks if the surveyed protocols respect and satisfy the main requirements already defined in Section 3.2.3. In Table 3.4, ✓ means the protocol satisfies the presented requirement, otherwise ✗ is mentioned.

3.2.6.3 Security

From the security perspective, Table 5.2 presents a comparison of the surveyed protocol by checking their capability to avoid the *nine* attacks defined in Section 3.2.4. In addition, it certifies if the indicated protocol respects the following security policies:

1. *Anonymity Device/Server*: It is essential for archiving secure authentication. It allows to reveal the identity of the IoT device/server in the network, and it avoids traceability of the device/server and its exchanged messages.

2. *Integrity*: It ensures that the authentication messages are not tampered during the authentication process.

3. *Confidentiality*: It protects the exchanged messages between two entities by ensuring that only the authorized device can view and use the content of an exchanged message.

Work	Complete Specification	Leakage Resilience	Able to Handle Noisiness	Strong PUF Response Space Expansion	Low-Cost and Resource-Constrained	Identification Prior to Authentication	On the Mutual Authentication Order
(author?) [33]	✓	✗	✗	✓	✗	✓	✗
(author?) [34]	✗	✗	✓	✓	✗	✓	✓
(author?) [35]	✗	✓	✗	✓	✓	✓	✗
(author?) [36]	✓	✗	✓	✓	✗	✓	✓
(author?) [37]	✗	✓	✗	✓	✓	✓	✗
(author?) [38]	✓	✗	✓	✓	✗	✓	✗
(author?) [39]	✓	✓	✗	✓	✓	✓	✓
(author?) [40]	✓	✗	✓	✓	✓	✓	✓
(author?) [41]	✓	✓	✗	✓	✓	✓	✓
(author?) [42]	✓	✗	✗	✓	✓	✓	✗
(author?) [43]	✓	✓	✗	✓	✓	✓	✓
(author?) [44]	✓	✗	✗	✓	✓	✓	✓

Table 3.4 PUF-based Authentication Protocol Requirements Satisfiability.

4. *Non-repudiation*: It ensures that no party of the authenticated devices can deny that it sent or received a message.

In the following comparison, ✓ means that the protocol is not secure against the indicated attack. But when it comes to the security policies, ✓ means that this protocol guarantees the corresponding policy. However, ✗ means the inverse of the previous statements regarding the attacks and the policies. Also, **?** means that the authors of the cited contribution did not show any purpose regarding the selected criteria.

Work	Replay Attack	Message Analysis Attack	physical attack	DoS Attack	Modeling attacks	MITM attacks	Cloning attack	Injection attack	Helper data manipulation	Anonymity Device	Anonymity server	Integrity	confidentiality	Non-repudiation
(author?) [33]	✗	✓	✗	✗	✗	✗	✗	✗	?	✗	✗	✗	✗	✗
(author?) [34]	✗	✗	✓	✗	✗	✗	✗	✗	✓	✓	✗	✓	✓	✗
(author?) [35]	✓	✓	✗	✓	✓	✓	✗	✓	**?**	✗	✗	✗	✗	✗
(author?) [36]	✗	✗	✓	✗	✗	✗	✗	✗	✓	✗	✗	✓	✓	✓
(author?) [37]	✓	✓	✗	✓	✓	✓	✗	**?**	**?**	✗	✗	✗	✗	✗
(author?) [38]	✓	✓	✓	✗	✗	✗	✗	✓	✓	✓	**?**	✗	✗	✗
(author?) [39]	✗	✗	✗	✗	✗	✗	✗	✗	**?**	✗	**?**	✓	✓	✗
(author?) [40]	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	**?**	✓	✓	✗
(author?) [41]	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓
(author?) [42]	✗	✗	✗	✓	✗	✗	✗	✗	**?**	✗	**?**	✗	✓	✗
(author?) [43]	✗	✗	✓	✗	✗	✗	✗	✗	**?**	✗	✗	✓	✓	✓
(author?) [44]	✗	✗	✗	✗	✗	✗	✗	✗	**?**	✗	**?**	✓	✓	✗

Table 3.5 Security of the proposed PUF-Based authentication Protocols.

The discussed IoT PUF-Based authentication protocols are compared in terms of practicality and performance, requirement satisfiability, and their resistance to the defined attacks. Based on the results showed in the comparison section, we found that.

- Most of the studied IoT protocols are designed for generic IoT systems without specifying their field of application, except three are about: IoMT, Smart Grid, and UAV. However, when a protocol is dedicated to a specif application, it will consider the requirements dedicated to that field and application. Indeed, many era of applications are affected by this case, especially critical systems and those known by their standards and norms, example: automotive, aviation, healthcare systems and military applications. From a technical point of view, most of the protocols use Hash algorithm and XOR operation to obfuscate messages, secure

the protocol, and guarantee the security requirements. When both techniques were not used, the protocol is hardened by a cryptographic algorithm such as Elliptic Curve. Further, CRPs are affected by the environment condition, thus fuzzy extractor is trivial to generate a stable and unique response. Hence, we found that the half of the proposed protocols guarantee noise elimination using fuzzy extractor techniques, which play an important role in the practicality of a protocol. In terms of authentication, only one protocol proposed a peer-to-peer authentication between IoT devices, while most of them do not generate session keys at the end of the authentication process. Concerning the verification and security analysis, most of works do not apply formal and informal analysis technique where some of them run simulation. We mention that ProVerif [156] is the most used tool for the security verification of the proposed protocols [36]. Also, BAN logic [1] has been proposed to specify the requirements and to prove the security of the protocols [40, 36]. For error correction, no prior definition behind the used techniques dedicated to *Generation* and *Reproduction* procedures of the deployed fuzzy extractor.

- From the learnt lessons regarding the studied requirements and their presented satisfiability in Table 3.4, we can say that all the surveyed protocols satisfy the strong PUF Response Space Expansion (*fourth requirement*) as well as the identification Prior to Authentication (*sixth requirement*) since they rely more on strong PUF and consider the uniqueness of the response, respectively. Also, we observe that most of protocols give a complete specification, except three [34, 37, 35]. In addition, for the leakage resilience, some protocols store secret and private information on the device memory [155, 38, 36, 43]. Unfortunately, many of them do not take into consideration noise elimination process [33, 35, 37, 39, 41, 42, 43, 44], low-cost and resource constraint of the IoT devices [33, 155, 36, 38]. Finally, some protocols do not offer mutual authentication [33, 35, 37, 38, 42].

- From a security perspective and by basing on the discussed PUF-based authentication protocols, it is clear that none of them is vulnerable to cloning attack due to the unclonability property of the used PUF. For the replay, message analysis, MITM, and injection attacks are present and can be successful within the protocols (like [35, 37]) that exchange the authentication messages in a clear text without adopting the appropriate solutions especially the hashing operations, cryptographic systems, and the use of timestamps variable. Physical attacks have been succeeded only with the protocols that store private and secret information on the device local memory [155, 36, 38, 43]. Dos attack is present only with the

[1] https://www.cl.cam.ac.uk/ rja14/Papers/SEv2-c03.pdf

scheme that does not verify the server trust and not use timestamp variables. Further, only the protocols that exchange CRPs without obfuscation or encryption are vulnerable to modeling attacks. For the helper data manipulation, only the protocols that use fuzzy extractor are vulnerable to this attack, especially when the helper data is stored in the device or transmitted over the communication channel in a plain text. Concerning the security policies, the anonymity properties are not present in the studied protocols. Except two [155, 38], by using the hashing value of the device identity instead of its ID only. In addition, the integrity of the exchanged data during the authentication process is guaranteed using, hash, MAC, and HMAC algorithms. Also, the confidentiality is ensured by deploying the cryptographic techniques. The non-repudiation is guaranteed only by three schemes: [36, 41, 43] .

3.2.7 Summary

In this section, we first give a short survey of the main existing authentication techniques used with IoT systems, including cryptography-based authentication, localization-based authentication, and PUF-based authentication. After that, we give a detailed survey of IoT PUF-based authentication protocols, where we give a description summary of each reviewed protocol, and further, we give its graphical representation. Finally, we compared the related works presented in terms of performance, requirements satisfiability, and security.

3.3 Conclusion

In this chapter, we provide a list of the relevant literature studies. We begin by discussing the innovative Silicon PUF architectures, followed by an overview of PUF applications. Then, we compared and debated the applications and architectures presented. In the second section of this chapter, we examine the current PUF-based authentication mechanisms for the Internet of Things. First, we assessed their security concerns, including the authentication methods in use. Then, we examined recent additions to authentication systems that leverage the randomness of IoT devices (IoT-based PUFs). Finally, we compared the assessed IoT PUF-based authentication protocols on the basis of their performance, strengths, portability, satisfaction of requirements, and resistance to existing vulnerabilities. This section is essential for identifying the constraints of the existing work, which will be necessary for the development of a robust authentication mechanism. We present our suggested IoT PUF-based authentication protocols in the following chapter, which constitutes the primary body of this thesis.

CHAPTER 4

LIGHTWEIGHT IoT PUF-BASED AUTHENTICATION PROTOCOLS

Nowadays, more constrained devices become connected, building an extensive Internet of Things (IoT) network but suffering from many security issues. In particular, authentication has become a severe research challenge for IoT systems. Furthermore, confidentiality, integrity, and availability are considered the core underpinnings of information security in general. Unfortunately, it is challenging to deploy conventional authentication protocols for IoT devices in practice for two main reasons. First, IoT devices are limited in memory capacity, processing power, and energy resources. Second, these protocols store secret keys on the IoT devices' volatile memory, making them vulnerable to physical attacks. Luckily, Physical Unclonable Functions (PUF) have emerged as a promising low-cost security primitive. A PUF eliminates the need to store secret keys in device memory, making them a potential alternative to deploying more secure and low-cost authentication protocols for IoT systems. Thing-to-Thing (T2T), or direct connection between IoT devices, represents a promising technique to enable things to communicate directly without the interaction of a trusted third party. In this chapter, our proposed authentication protocols were presented. We have designed three novel lightweight Mutual Authentication and Key Exchange Protocols (MAKEP) for IoT devices using PUFs. The first scheme ensures secure communication for Thing-to-Server (T2S) and is called T2S-MAKEP. The second, T2T-MAKEP, allows two IoT devices to communicate with each other via an embedded strong PUF circuit, and the third one is LT2S-MAKEP, for Lightweight T2S-MAKEP, which is suitable for resource-constrained IoT devices. All of the proposed protocols, T2T-MAKEP, T2S-MAKEP, and LT2S-MAKEP, allow robust authentication without storing any information on the device's memory and simultaneously establishing the session key exchange. The main focuses of this chapter is:

1. Presenting the system and security models as well as the requirements related to the proposed protocols.

2. Showing how generating, producing, and correcting the stream keys.

3. Introducing T2S-MAKEP for mutual authentication and key exchange protocol between a thing and a server.

4. Introducing T2T-MAKEP for direct mutual authentication and key exchange protocol between two things.

5. Introducing LT2S-MAKEP for for lightweight mutual authentication and key exchange protocol between a thing and a server.

4.1 Architecture Settings

This section details the system model, security model, the used notation requirements, and the assumptions used in the design of the proposed protocols.

4.1.1 System Model

As shown in Fig. 4.1, we consider the same system model of IoT as in [157, 36, 119]. The network model mainly contains two entities: Things and the server. Things could include various sensor devices, and the server is responsible for managing things and storing security parameters. Things can communicate with the server, but they can communicate directly with other things through the internet network. Although the considered system model is simple, the proposed protocol can be applied to various complex IoT network models. In this paper, we assume that:

- Each IoT thing is equipped with an embedded PUF circuit.

- Any physical tampering with the PUF will irreversibly modify the slight physical variations in the integrated circuit, which in turn changes the PUF challenge-response behaviour, or even destroys the PUF circuit.

- The IoT things have limited storage capacity and cannot protect any stored secrets in their local storage memory.

- The server is considered the trusted party with no limitation of resources that can be found only in a secure area.

- The communication between the IoT thing microcontroller and its PUF component cannot be accessed only through a secure channel [119].

- The proposed solution targets single-hop networks.

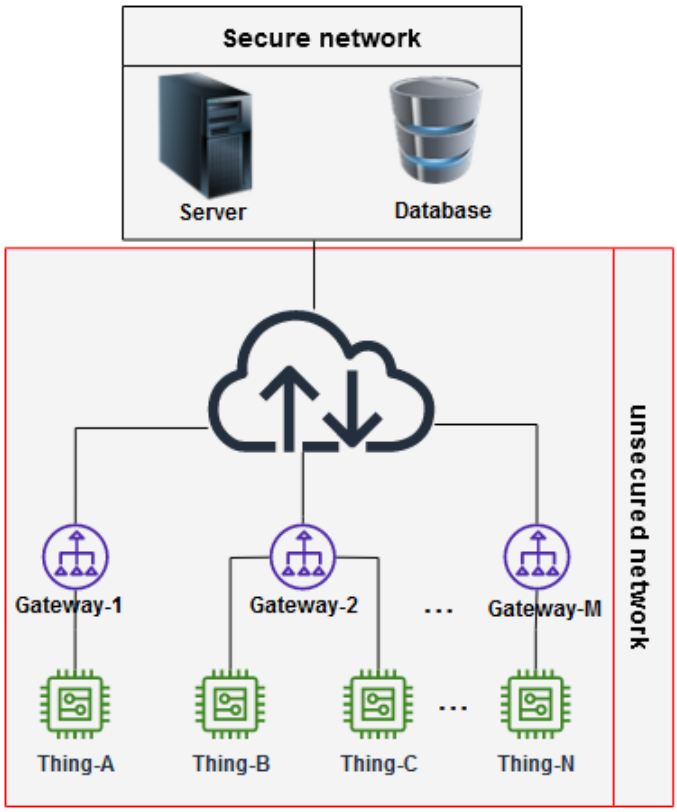

Fig. 4.1 System model [32].

4.1.2 Security Model

In the mentioned network model, we consider that the IoT things are untrustworthy devices, and they could be installed in a public space, which allows an attacker to capture things and obtain the crucial secrets from their memory. In addition, each thing communicates with the other through the Internet using wireless technology, where an attacker can read, manipulate, forge, reply, delay, and delete messages during the communication between things or between things and the server. Also, it is assumed that an adversary can disrupt the network using a denial of service attack. Besides, the attacker cannot have access to the server's database.

4.1.3 Requirements

This section contains the design requirements for three proposed protocols. Functional requirements, security requirements, and performance requirements make up these requirements.

4.1.3.1 Functional Requirements

T2S-MAKEP, T2T-MAKEP, and LT2S-MAKEP are designed to satisfy the following functional requirements.

- **Achieve identification of IoT devices:** The trusted server or the Internet of Things (IoT) device (T2T-MAKEP) must be able to identify each new object that requests authentication. This identification should rely solely on the answer generated by the PUF utilized by the new entity.
- **Achieve authentication of IoT devices:** Each thing should be authenticated securely by the server before accessing the trusted network.
- **Achieve authentication of the server:** The thing communicating with the server should authenticate the server.
- **Achieve data authenticity during the authentication execution process:** This is necessary to ensure the origin authentication and integrity protection of data sent to or generated by the communicating entities. In other words, the exchanged messages during authentication must maintain their integrity.

4.1.3.2 Security Requirements

This section specifies a set of security requirements for the proposed designs.

- **Entity Authentications:** Entity authentication is to ensure that an entity is the one that it claims to be. This is to ensure that only trustworthy objects can connect to the trusted network. This should cover mutual authentication between a device and a trusted server. Or among two objects.
- **Authenticity of Protocol Messages:** Origin authentication and integrity protection should also be applied to all protocol messages (requests and responses) that help to achieve the identification and authentication process.
- **Confidentiality of Protocol Messages:** Entity identification and authentication data carried in protocol messages should be kept confidential.
- **Confidentiality of the stored data:** The confidentiality of the stored data on the server used in the entity identification and authentication process should be kept secret.

4.1.3.3 Performance Requirements

In addition to functional and security requirements, the design must also meet performance requirements in order to be as effective as possible. The following criteria are examined in terms of performance:

- **The communication overhead:** The communication overhead introduced in each proposed protocol should be as low as possible. This means that both the

number of authentication messages and the length of each message should be as low as possible.

- **The computational overhead:** The computational overhead incurred in accomplishing the functions of each proposed protocol should be as low as possible. This means that the computational costs of any algorithms and/or cryptographic primitives chosen for the implementation of the these functions should be taken into account.

- **Storage Data :** The stored data on the IoT device and the server should be minimised as possible.

4.1.4 Notations

Table 4.1 defines the notations used by both proposed protocols: T2S-MAKEP and T2T-MAKEP.

Symbols	Definitions
ID_A	The identity of a IoT device *A*
Reg_{req}	Registration request
$Auth_{req}$	Authentication request
$C_{A,i}$	The i^{th} challenge of the device *A*
h(.)	One-way hash function
PUF_A	The PUF of a device *A*
$R_{A,i}$	A response of the challenge $C_{A,i}$
$R'_{A,i}$	A noisy response of $C_{A,i}$
Gen(.)	Generation procedure of Fuzzy Extractor
Rep(.,.)	Reproduction procedure of Fuzzy Extractor
$K_{A,i}$	Extracted key from $R_{A,i}$
$P_{A,i}$	Helper data of $R_{A,i}$
TS	Timestamps
$\|\|$	Concatenation symbol

Table 4.1 Authentication protocol's symbols.

In addition to those notation listed in Table 4.1, Table 4.2 defines some of new notations used by LT2S-MAKEP.

4.2 T2S-MAKEP Scheme

In this section, we provide a detailed description of the proposed mutual authentication protocol between an IoT device, equipped with a PUF, and the server. A physical unclonable function, a hashing function, and a fuzzy extractor are employed in our

Symbol	Definition
ID_d	Device Identity i
C	Challenge
$APUF$	Arbiter PUF of device ID_d
$SRAM.PUF$	SRAM PUF of device ID_d
R_{APUF}	A response of C Generated by $APUF$
R_{SRAM}	A response of C Generated by $SRAM$ PUF
R'_{APUF}	A noisy response of C Generated by $APUF$
R'_{SRAM}	A noisy response of C Generated by $SRAM$ PUF
K_{APUF}	Extracted key from R_{APUF}
K_{SRAM}	Extracted key from R_{SRAM}
P_{APUF}	Helper data of R_{APUF}
P_{SRAM}	Helper data of R_{SRAM}

Table 4.2 LT2S-MAKEP's symbols.

scheme. The protocol consists of three modules; 1) the enrollment phase, 2) the authentication phase, and 3) session key establishment. Fig. 4.2 shows T2S-MAKEP system model.

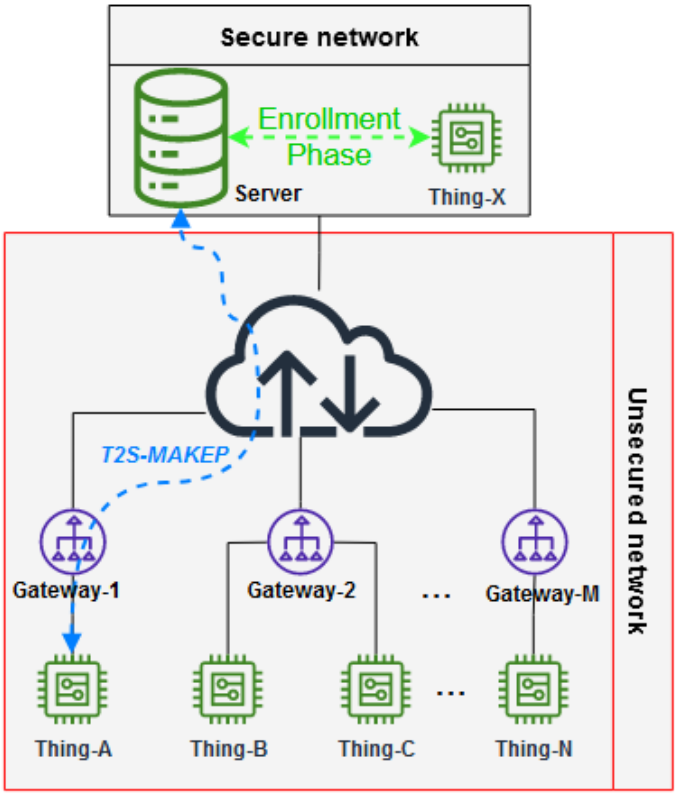

Fig. 4.2 T2S-MAKEP system model.

4.2.1 Enrollment Phase

In the most existing PUF-based authentication protocols that store a considerable number of CRPs in the authentication process the server challenges the IoT device with one randomly selected pair, and at the end, it will be deleted from the server database. In our proposed mechanism, the server stores only one pair. It minimizes the security threat due to the confidentiality of the stored data and the resources of the server database [42]. In this phase, when a new IoT device needs to be added as a member of the trusted network, it goes first with the server through the enrollment phase. This phase is executed in a trusted and secure environment. Fig. 4.3 describes the main steps of this phase to be performed by the *IoT-Device-A* and the $Trusted - server$ are as follows.

- *IoT-Device-A* sends its identity Id_A in plain text to the $Trusted - server$ with a registration request Reg_{req}.
- $Trusted-server$ generates randomly a challenge $C_{A,i}$, and sent it to *IoT-Device-A* within ID_A.
- The *IoT-Device-A* inputs this challenge into its PUF component to output the corresponding response $R_{A,i}$=$PUF_A(C_{A,i})$. Then, $IoT - Device_A$ sends to the server $ID_A, C_{A,i}, R_{A,i}$.
- Using the generation procedure of fuzzy extractor Gen, the $Trusted-server$ extracts the secret key $K_{A,i}$ and the public helper data $P_{A,i}$, ($\mathrm{K}_{A,i}, P_{A,i}$)=Gen($R_{A,i}$). Then, the server computes the hash of the device identity and the secret key and stores h(ID_A), $C_{A,i}, h(K_{A,i}), P_{A,i}$ on its local secure database.
- In the end, $Trusted - server$ informs *IoT-Device-A* about the end of the registration process.

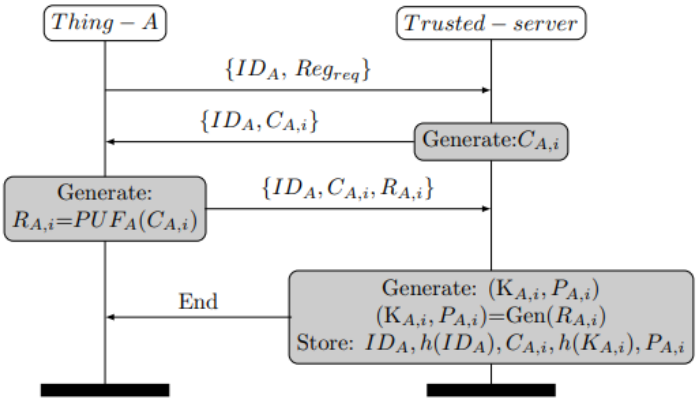

Fig. 4.3 Enrollment Phase.

After a successful registration, the Internet of Things device may be deployed and activated. Since preserved credentials ($C_{A,i}, h(K_{A,i}), P_{A,i}$) are automatically created at

the conclusion of every successful authentication operation, the device will never seek for the server's permission to register.

4.2.2 Authentication Phase

Our proposed mutual authentication scheme is based on deploying a silicon PUF, especially a strong PUF, in the IoT device. The idea behind using the PUF is that the IoT device uses the challenge-response pair of the PUF as a fingerprint and uses it to prove its identity with the server. The device and the server achieve mutual authentication since only those who know about the generated secret key for a given challenge in the enrollment phase and stored on the trusted server. In this proposed protocol, the server stores only one pair of CRPs to avoid attacks on the server. One of the most vital points of our protocol is that the IoT device does not store any secret or public information, which avoids physical attacks.

In the authentication phase to check the device's identity, PUF-based authentication protocols mainly compare the stored response in the enrollment phase to the new generated one to the same given challenge. Unfortunately, generating the same response for the same challenge in different environments and conditions such as voltage and temperature makes the response noisy or hazarded compared to the original one. This step is processed differently in our case, so to generate the secret key and store it safely on the server for the authentication process, the error correction technique has been adopted to eliminate the noise and ensure the comparison operation. More precisely, the proposed protocol takes into consideration the noise elimination process using the fuzzy extractor.

As shown in Fig. 4.4 the authentication process between the IoT device (*IoT-Device-A*) and the server (trusted-server) is running as follows.

1. Firstly, in **step (1)**, when the *IoT-Device-A* wants to communicate with the server, it generates a timestamp TS_1 and calculates a hash value of its identity (ID_A), $h(ID_A, TS_1)$. Then, it sends the hash of its identifier $h(ID_A)$, the authentication request $Auth_{req}$ and $h(ID_A, TS_1)$ message to the server.

2. In **Step (2)**, upon receiving the message from the IoT device, the server executes the following operations:

   - The server checks the existence of the received $h(ID_A)$ in its database.
   - If the finding fails, the server rejects the authentication request. Otherwise, the server verifies the received message integrity by calculating $h(ID_A, TS_1)$ message and matches both the calculated hash messages within the received one.

- If the matching fails, the server rejects the authentication request. Otherwise, the server makes sure that the message has not been corrupted or tampered during the transmission phase. To calculate the message m_2: $h(ID_A, C_{A,i}, P_{A,i}, K_{A,i}, TS_2)$, the trusted server retrieves $C_{A,i}, P_{A,i}, h(K_{A,i})$ associated with the ID_A from its database and generates a timestamp TS_2.
- Finally, the server sends to the IoT device *A*: the stored challenge $C_{A,i}$, the helper data corresponding to this challenge $P_{A,i}$, TS_2, and the message $h(ID_A, C_{A,i}, P_{A,i}, K_{A,i}, TS_2)$.

3. In **Step (3)**, once the IoT device receives the server response, the following steps are executed.
   - The IoT device uses its PUF to generate the response of the received challenge: $R'_{A,i} = PUF_A(C_{A,i})$.
   - By default, the generated response is considered noisy. Then, the device reproduces the secret key $K_{A,i} = Rep(R'_{A,i}, P_{A,i})$ from the noisy response $R'_{A,i}$ using the reproduction process of the fuzzy extractor.
   - To verify the integrity of the received message from the server, *IoT-Device-A* calculates $h(h(ID_A), C_{A,i}, P_{A,i}, h(K_{A,i}), TS_2)$, and compares the received and the calculated hash messages. A successful matching provides the first factor for authenticating the server within the *IoT-Device-A*. This matching is ensured since the server is the only entity in the network that knows the secret key $K_{A,i}$ generated from the response $R_{A,i}$ of $C_{A,i}$.
   - If the comparison fails, the connection is rejected by the IoT device. Otherwise, the IoT device generates a timestamp TS_3, computes a new challenge $C_{A,i+1} = h(C_{A,i}||K_{A,i})$, and generates the response corresponding to the new computed challenge $R_{A,i+1} = PUF_A(C_{A,i+1})$.
   - Finally, the device calculates $h(ID_A, TS_3, k_{A,i}, R_{A,i+1})$ message, encrypts the new response with the reproduced secret key $(R_{A,i+1})_{h(K_{A,i})}$, and sends back the calculated message with TS_3 and the encrypted data.

4. Upon receiving the message from the IoT device, the server executes the following operations:
   - Decrypts $(R_{A,i+1})_{h(K_{A,i})}$ using the stored secret key $h(K_{A,i})$.
   - Verifies the integrity of the received data by calculating $h(ID_A, TS_3, k_{A,i}, R_{A,i+1})$.
   - Compares the received and the calculated hash messages. If the matching fails, the server terminates the connection. Else, the server validates the authenticity of the IoT device as a successful matching of these two hash

messages. Consequently, the IoT is authenticated and the server communicates with the right IoT device.

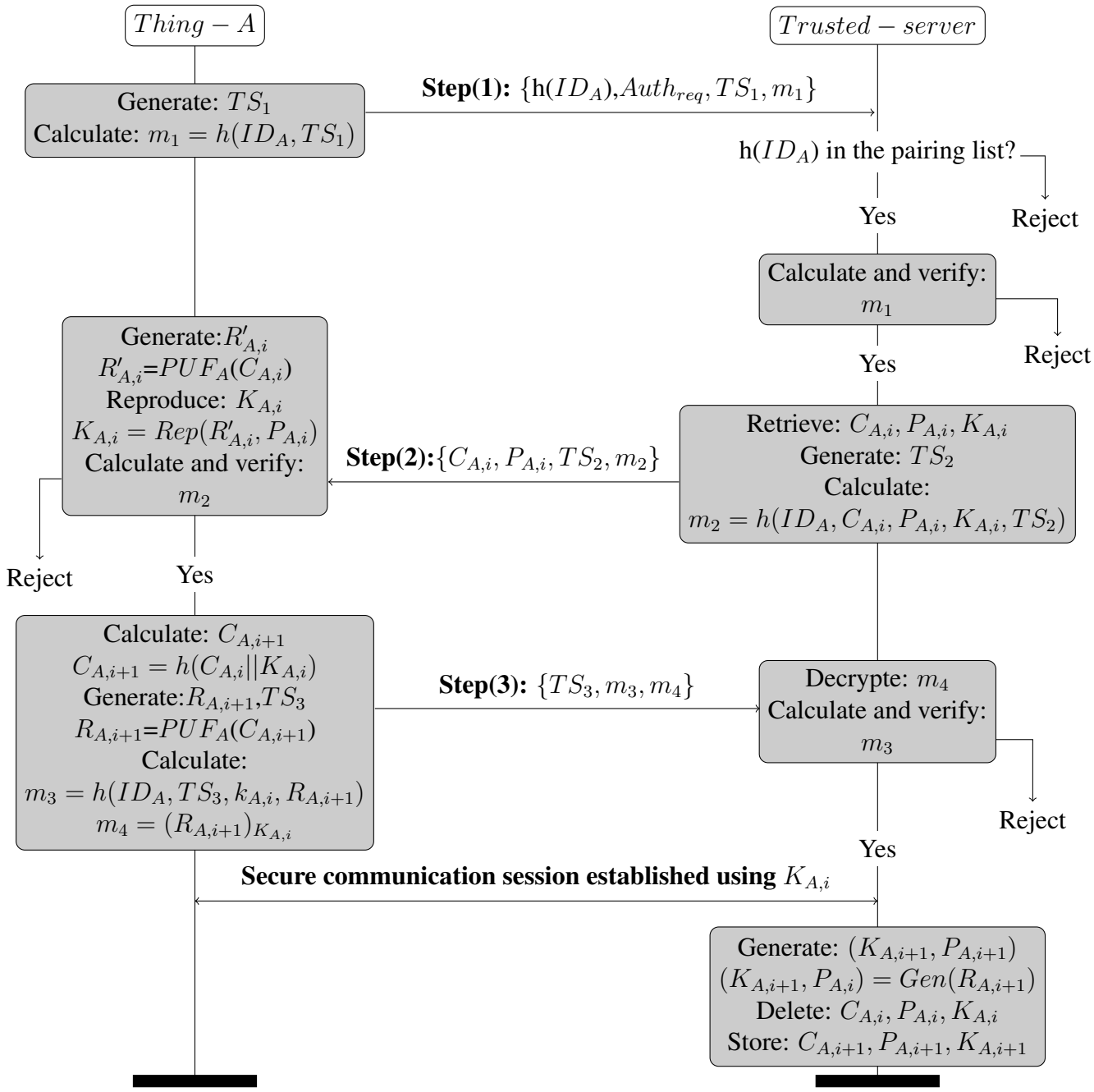

Fig. 4.4 T2S-MAKEP Protocol.

4.2.3 Session key establishment

After authenticating the IoT device in the trusted server, both communicating entities can use the secret and exchanged key $h(K_{A,i})$ as a session key to secure the communication during the current session. $h(K_{A,i})$ is securely generated by the device and

stored by the server. Also, it has never been used and communicated in plain-text. At the end of the communication, the server generates a new secret key and its corresponding helper data from the new generated response $R_{A,i+1}$, using the generation procedure of the fuzzy extractor $(K_{A,i+1}, P_{A,i}) = Gen(R_{A,i+1})$. Then, it calculates the new challenge $C_{A,i+1} = h(C_{A,i}||K_{A,i})$. Finally, the server updates the used information $C_{A,i}, P_{A,i}, h(K_{A,i})$ by the new one $C_{A,i+1}, P_{A,i+1}, h(K_{A,i+1})$ for a next authentication of the device ID_A.

4.3 T2T-MAKEP Scheme

In this section, we detail the proposed mutual authentication and key exchange protocol between two things (Thing-A and Thing-B). By following T2T-MAKEP rules, things could authenticate and communicate each to others without a real participation of the trusted server. As the first schema, T2T-MAKEP consists of three modules; 1) the enrollment phase, 2) the authentication phase, and 3) session key establishment. Fig. 4.5 shows T2T-MAKEP system model.

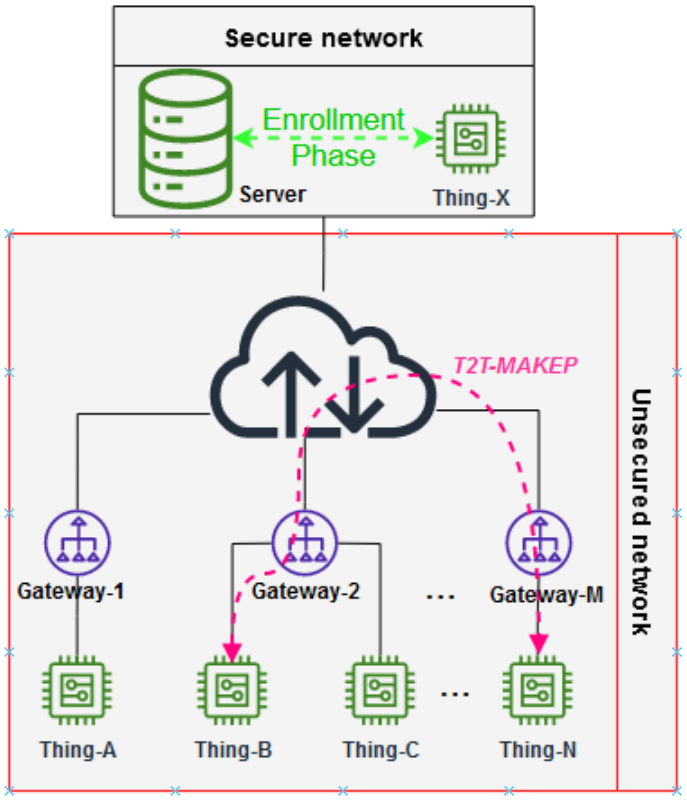

Fig. 4.5 T2T-MAKEP system model.

4.3.1 Enrollment Phase

This phase is identical to the one presented in T2S-MAKEP.

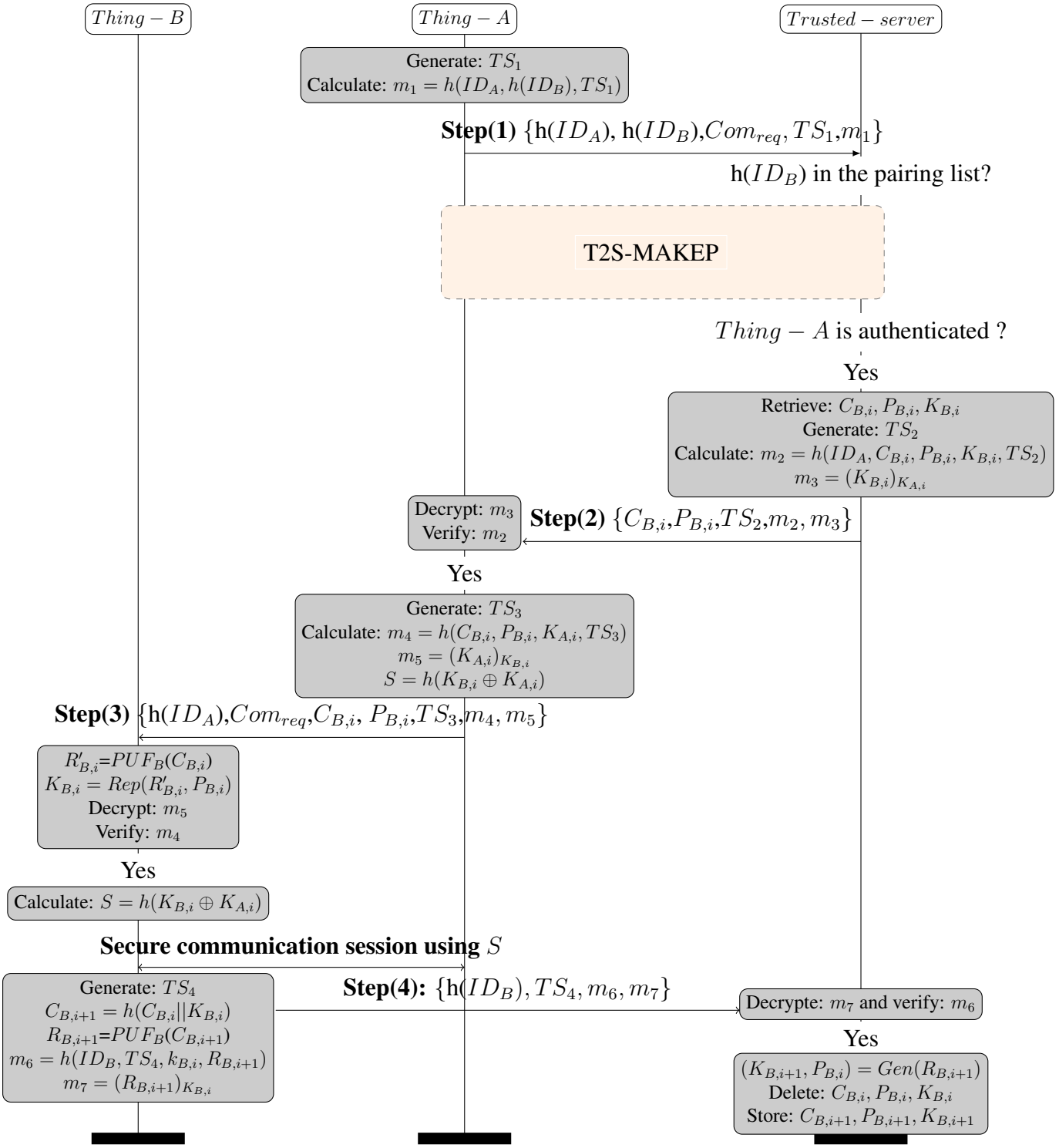

Fig. 4.6 T2T-MAKEP Protocol.

4.3.2 Authentication Phase

As shown in Figure 4.6, T2S-MAKEP is used as an essential step at the first stage of T2T-MAKEP executions that are detailed as follows. When an IoT end-device $Thing - A$ wants to communicate with another end-device $Thing - B$, first in **Step (1)**, $Thing - A$ generates a timestamp TS_1 and calculates a message m_1, where $m_1 = h(ID_A, h(ID_B), TS_1)$ to ensure the integrity of the T2T communication request message. Then, the IoT device sends the hash of its identifier $h(ID_A)$, the hash of $Thing - B$ identifier $h(ID_B)$, the authentication request Com_{req}, and the message m to the server.

In **Step (2)**, upon receiving the message from $Thing_A$, the server checks the validity of $Thing_B$ identity by checking $h(ID_B)$ in its database. If the finding fails, the server rejects the authentication request. Otherwise, the server follows steps of Section 4.2.2 to launch T2S-MAKEP process and verify the authenticity of the IoT device lunched the communication request ($Thing_A$).

At the end of the T2S-MAKEP phase, if the authentication fails, the server rejects the authentication request. Else, $Thing - A$ is authenticated and the message of the communication request was not corrupted or tampered during the transmission phase. Then, the server retrieves the correspondent stored data of $Thing - B$: ID_B, $C_{B,i}$, $P_{B,i}$, $K_{B,i}$ and generates a timestamp TS_2.

After that, the server calculates both messages m_2 and m_3, where m_2 is used to guaranty the integrity of the transmitted data. In addition, $m_3 = (K_{B,i})_{K_{A,i}}$ is a result of the cryptographic operation on the secret key $K_{B,i}$ using the secure session key $K_{A,i}$ established at the end of T2S-MAKEP phase. Finally, the server sends $C_{B,i}$, $P_{B,i}$,TS_2,m_2 and m_3 to the device that lunched the communication request $Thing - A$.

In **Step (3)**, once $Thing - A$ receives the server response, it decrypts m_3 by using the T2S-MAKEP session key $K_{A,i}$. Then, it verifies the integrity of the received response by calculating m_2 using the decrypted message $K_{B,i}$ and the non-encrypted received data ($C_{B,i}$, $P_{B,i}$, and TS_2). If the verification fails and the connection does not rejected rejected, $Thing - A$ generates a timestamp TS_3 and calculates $m_4 = h(C_{B,i}, P_{B,i}, K_{B,i}, TS_3)$, $m_5 = (K_{A,i})_{K_{B,i}}$, and $S = h(K_{B,i} \oplus \mathrm{K}_{A,i})$ which is the new session key used to secure the communication between the two things by Xoring the secret keys of both devices. Finally, $Thing - A$ sends the hashed identity of $Thing - b$ ID_B), the challenge $C_{B,i}$, the helper data $P_{B,i}$, a timestamp TS_2, the control of the integrity m_4 , and $Thing - A$ encrypted secret key m_5.

Afterwards,$Thing - B$ uses its PUF component and generates the response of the received challenge $R'_{B,i} = PUF_B(C_{b,i})$, which is considered as a noisy response. Then, it reproduces the original secret key $K_{B,i}$ from the noisy response through the fuzzy extractor reproduction process $K_{B,i} = Rep(R'_{B,i}, P_{B,i})$. It uses this key to decrypt the

received encrypted message m_5, then the result $K_{A,i}$ is used to calculate and verify the integrity of m_4. In the communication request is granted, the IoT end-device $Thing-B$ makes sure that the message was not tampered during the transmission phase. Also since only the server has a secret key $K_{B,i}$, the device requesting the connection is trusted with the same trusted network and server.

4.3.3 Session key establishment

Finally, $Thing - B$ calculates the new session key $S = h(K_{B,i} \oplus \mathrm{K}_{A,i})$ used to secure the communication with $Thing - A$. At this step, both IoT end-devices could communicate securely using the secret session key *S*.

Finally, in **Step (4)** $Thing - B$ and the server execute the needed operations to update the used information $C_{B,i}, P_{B,i}, K_{B,i}$ by the new generated one $C_{B,i+1}, P_{B,i+1}, K_{B,i+1}$

4.4 LT2S-MAKEP Scheme

In this section, we present our third contribution named LT2S-MAKEP for lightweight thing-2-server mutual authentication and key exchange protocol between a thing and a server, in this scheme, we exploit two silicon PUFs built-in with a given IoT device, one is a weak PUF (SRAM) and another is a strong PUF (Arbiter PUF). SRAM PUF is used to generate a secret cryptographic key used to encrypt the exchanged messages during the authentication phase, and the APUF is used during the authentication process. Utilizing the weak PUF with APUF in the same IoT device avoids the need of using another cryptographic primitive to protect the response during transmission phase. Thus, the device uses the response generated by APUF to prove its identity in the network. On the other side, this scheme differs from T2S-MAKEP and T2T-MAKEP, since it achieve a a secure authentication using PUF without a need to noise elimination process such as fuzzy extractor, this due to the advanced research that showed that weak PUF is more reliable and has zero bit-error-rate when regenerate the same response for a given same challenge [158, 159]. Further, the authenticated is based on the comparison of the stored APUF response $R\_APUF_{A,i}$ with the new generated one $R\_APUF'_{A,i}$ regarding if they are close each to other. This comparison principal is inspired from the Fuzzy extractor technique (see fuzzy extractor and secure sketch definition.A.2.3). Fig. 4.7 shows the network model of the LT2S-MAKEP proposed scheme, where each IoT device is assumed to communicate with the trusted server through a public and unsecured network like the Internet. Each IoT device is considered as a resource limited (memory and processing) and is equipped with two PUFs.

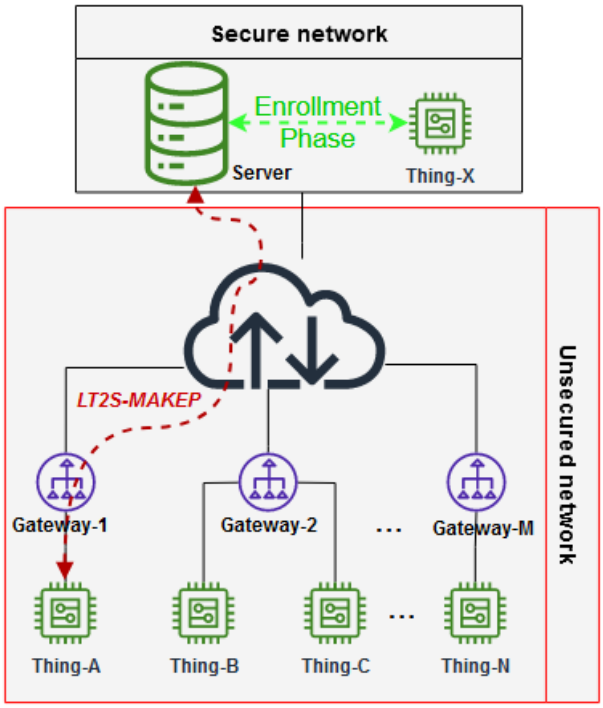

Fig. 4.7 LT2S-MAKEP system model.

4.4.1 Enrollment phase

Fallowing the same principal, in this proposed scheme the server stores only one pair of challenge and response. When a new device, equipped with an APUF and SRAM PUF, needs to be added as a member of the trusted network, the device and the server go first through the Enrollment phase. This phase is executed in a trusted and secure environment. Fig. 4.8 describes the main steps of this phase to be performed by the $Thing - A$ and the $Trusted - server$ are as follows.

- $Thing - A$ sends its identity Id_A in plain text to the $Trusted - server$ with a registration request Reg_{req}.
- $Trusted - server$ generates randomly a challenge $C_{A,i}$, and sent it to $Thing - A$ within ID_A.
- The $Thing - A$ inputs this challenge into its both PUFs APUF and SRAM PUF to output the corresponding responses, respectively R_SRAM_A=$SRAM\_PUF_A(C_{A,i})$ and $R\_APUF_{A,i}$=$APUF(C_{A,i})$. Then, $Thing - A$ sends the generated responses to the server R_SRAM_A and $R\_APUF_{A,i}$ with its ID_A.
- The $Trusted - server$ calculates the secret key K_SRAM_A by computing the hash of R_SRAM_A, and stores $ID_A, C_{A,i}, K\_SRAM_A, R\_APUF_{A,i}$ on its local secure database.

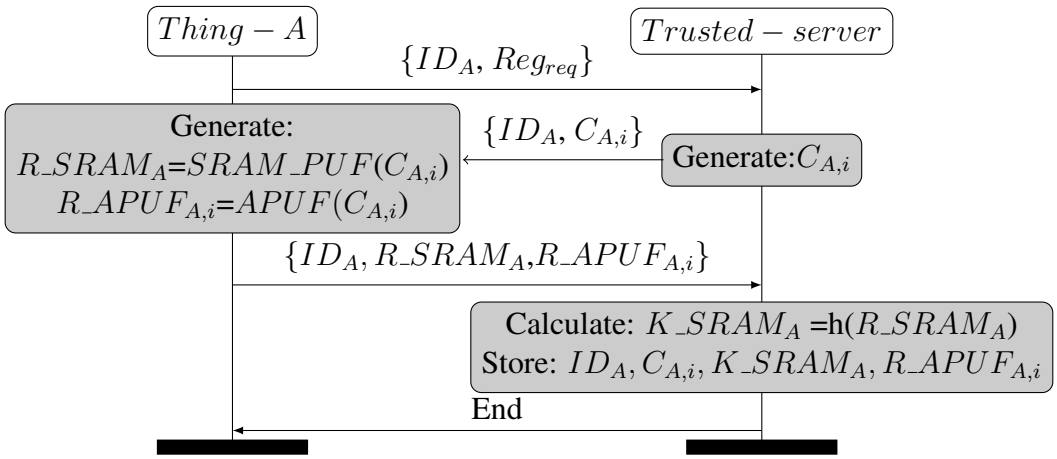

Fig. 4.8 LT2T-MAKEP Enrollment Phase.

- In the end, $Trusted-server$ informs $Thing-A$ about the end of the registration process.

4.4.2 Authentication phase

Our proposed scheme is a two-factor authentication by using two silicon PUFs built within the IoT device, SRAM PUF and APUF. The IoT device does not store any information, besides the server store only one CRP pair of each used PUF. In our scheme, we use PUF only as security primitive, where the exchanged responses secure and encrypt the communication by using the secret key extracted from the weak PUF, and for the integrity of the transmitted information, a one-way hash function is used in each transmitted message. Also, to ensure the freshness of messages and to prevent replay attacks, we send with each message a timestamp value. Fig. 4.9 shows the detailed steps of our proposed protocol described below.

- The device first sends the authentication request $Auth_{req}$, the device identity ID_A and a timestamp Ts_1.

- Upon receiving the message from the IoT device by the server, it directly verifies the identity of the device ID_A in his storage system and the integrity of the received message m_1.

- If it fails to find it, the connection is rejected. Otherwise, the $Trusted - server$ loads from its memory the stored information among the received ID_A that are: the SRAM secret key K_SRAM_A and the stored response of APUF $R\_APUF_{A,i}$. Then generates a timestamp Ts_2. After that, it calculates the messages $m_2 = C_{A,i}, Ts_2, h(ID_A, C_{A,i}, K\_SRAM_A, TS_2)$.

- Then, the server sends to the device its identifier (ID_d), the challenge $C_{A,i}$, and the calculated message m_2.

- Once received by the device, first, it verifies the integrity of m_2, if the verification process fails, the connection is rejected. Otherwise, it generates the response proper to the challenge using SRAM PUF and APUF, (R_SRAM_A=$SRAM\_PUF(C_{A,i})$) and $R\_APUF'_{A,i}$=$APUF(C_{A,i})$, respectively. Then, calculate the secret key K_SRAM_A =h(R_SRAM_A). After that, it generates a new timestamp Ts_3, and calculates two new messages $m_3 = h(ID_A, TS_3, R\_APUF'_{A,i})$
$m_4 = (R\_APUF'_{A,i})_{K\_SRAM_A}$. Finally, the device sends to the server TS_3, m_3, and m_4.

- When received by the server, it decrypts m_3 and verify the validity of m_4. If the verification fails, the server terminates the connection. Else, the server compares the received APUF response $R'\_APUF_{A,i}$ with the sored one $R\_APUF_{A,i}$, if both response are close each to other, the server validates the authenticity of the IoT device as a successful matching of these two response messages. Consequently, the IoT is authenticated and the server communicates with the right IoT device. Else, the connection is rejected.

- At this step, both the server and the device calculate the secret session key $K_{A,i}$ and they can communicate in a secure way.

- As the server stores only one CRP, the proposed protocol need an update phase at the end of each successful authentication. for this, $Thing - A$ generates a timestamp TS_4, computes a new challenge $C_{A,i+1} = h(C_{A,i}||K_{A,i})$, and generates the response corresponding to the new challenge $R\_APUF_{A,i+1}$=$APUF(C_{A,i+1})$. Then, the device calculates $m_5 = h(ID_A, TS_4, R\_APUF_{A,i+1})$ message, encrypts the new response with the secret key $m_6 = (R\_APUF_{A,i+1})_{K_{A,i}}$, and sends back the calculated message with TS_3 and the encrypted data.

- At the end, the server decryptes m_6 and verifies m_5, if the verification pass, the server, the server updates the used information $C_{A,i}, R\_APUF_{A,i}$ by the new one $C_{A,i+1}, R\_APUF_{A,i+1}$ for a future authentication of the device ID_A.

4.5 Conclusion

In this chapter, we have proposed three schemes of IoT PUF-based authentication protocol, a thing-to-thing mutual authentication and key exchange protocol for IoT devices

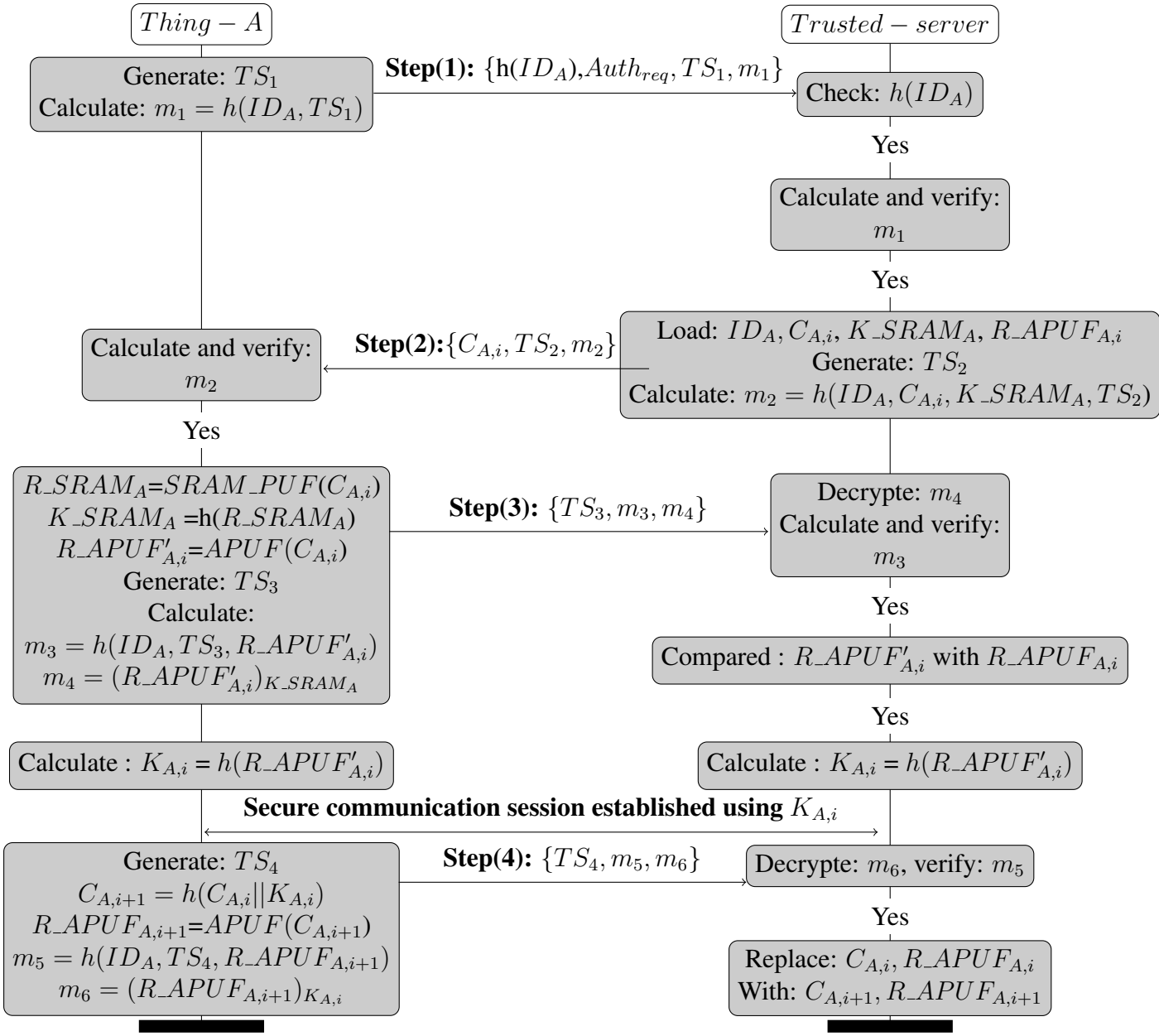

Fig. 4.9 LT2S-MAKEP Protocol.

(T2T-MAKEP), where a thing wants to communicate with another. First, it has to authenticate with trusted_server using the T2S-MAKEP scheme. In practicality, the proposed schemes guarantee error correction and noise elimination using a fuzzy extractor solution. Further, T2T-MAKEP does not need an update phase without storing any secret or non-secret information on the local memory of the IoT end-device. Also, at the end of each successful authentication phase, our schemes generate a session key and exchange it securely between the communicating entities. Further, we have developed a Lightweight low-cost protocol LT2S-MAKEP that relies on two PUFs: the SRAM PUF and the Arbiter PUF. The first is used for encryption, whereas the second is used for authentication. In the next chapter, we verify the robustness of the proposed protocols using formal and informal security analysis techniques.

CHAPTER 5

VERIFICATION AND ANALYSIS

Before implementation, we first check the correctness, the soundness and the robustness of our proposed authentication protocols. This chapter provides formal and informal analysis of the proposed authentication protocols' security. To give additional evidence for the security of the protocols, we modeled and formalized the developed schemes using the Verifpal verification tool. In addition, the evaluation of the performance of the proposed protocols is examined in terms of computational overhead, communication overhead, and storage restrictions. Principally, the main concerns of this chapter is:

1. Presenting the existing attacks against IoT PUF-based authentication protocols.
2. Argumenting the use of Verifpal verification tool.
3. Presenting and detailing the formal and informal security analysis of T2S-MAKEP, T2T-MAKEP, and LT2S-MAKEP schemes.
4. Presenting and discussing the performance analysis of the proposed protocols.
5. Comparing the proposed protocols within the existing ones in terms of security.

5.1 Attack Scenarios

A brief definition about the most known attacks [34, 44] that an advisory can launch on an IoT PUF-based authentication protocol is presented as follows:

- **Message Analysis Attack:** This kind of attack concerns the confidentiality of the exchanged message during or after a successful authentication phase. Meaning that an adversary tries to obtain the transmitted information between the communication entities.
- **Replay attack:** In this attack, the adversaries store the transmitted information of the valid authentication operation with the hope of utilizing it in future authentication.

- **DoS attack:** The goal of adversaries in this attack is to exploit all the resources of the target device, and/or to temporarily or indefinitely disrupt services. It is accomplished by flooding the targeted device with superfluous authentication requests.
- **Physical attack:** In this kind of attack, an adversary may attempt to physically get access to a device and try to obtain the secret information stored on its local memory.
- **MITM and Injection attack:** In this attack, an adversary tries to intercepts the exchanged information between different entities during their authentication process. It possibly also alternating the communications between them and makes them believe that they are directly communicating with each other.
- **Modeling attack:** From the communication between the server and the device, an adversary tries to get CRPs of the used PUF. Then, it executes the machines learning techniques on the CRPs dataset and offers to the adversary the possibility of predicting the correct response related to a given challenge.
- **Helper data manipulation:** This attack targets to make the fuzzy extractor operations impossible by modifying the helper data transmitted or stored on the device memory.
- **Cloning attack:** Our proposed protocol resists cloning attacks since it exploits PUFs for authentication, and the main feature of PUF is unclonability as its name says.
- **Camouflage attacks:** In this attack, an adversary inserts into the network a counterfeit IoT device to operate as a normal device or attacks an authorized device in order to obtain, process, send or redirect and manipulate packets, or be passive and just analyze the traffic [160] .
- **Anonymity Device:** In this attack, the intruder targets to reveal the identity of the IoT device and its traceability in the network.

5.2 Verifpal

In order to analyze our proposed protocols formally, we have selected Verifpal, a modern tool dedicated to the formal verification of the security of cryptographic protocols. It is inspired by some older formal verification tools such as ProVerif, and it covers the most well-known model, Dolev-Yao, which is a modelling technique for verifying cryptographic protocols [161]. Verifpal is a the symbolic verification tool, Verifpal. Compared to the most relevant protocol verification tools based on symbolic models, mainly

Tamarin and ProVerif, Verifpal extends them to cover more properties to construct correctly and verify efficiently cryptographic communication protocols, especially the following.

- **Modeling** looks if graphical or pre-defined behaviours can be instantiated and repeated.
- **Cryptography** shows if cryptographic syntax is supported like standard hashing, encryption and decryption functions.
- **Unbound** is to not limit the number of communication sessions.
- **Equational** presents if the user defined functions are supported, in addition to classical communication calculus properties, like: associative and commutative properties.
- **Mutability** shows if the tool support the verification of protocols with global mutable state.
- **Equivalence** checks if the properties related to equivalence checking are enabled, like: trace equivalence, bisimilarity, etc.
- **Implementation** shows how possible is generating an executable code.

The comparison presented in Table 5.1 extending the results presented in [162] shows that Verifpal [45] is a good candidate to specify correctly and verify efficiently cryptographic protocols.

Tool	Modeling	Cryptography	Unbound	Equational	Mutablity	Equivalence	Implementation
CPSA [163]	○	○	●	○	●	○	○
DEEPSPEC [164]	○	◐	○	◐	●	●	○
F7 [165]	○	◐	●	◐	●	○	●
Maude-NPA [166]	○	◐	●	●	○	●	○
ProVerif [167]	◐	◐	●	◐	○	●	○
Scyther [168]	◐	○	●	○	○	○	○
Tamarin [169]	◐	◐	●	●	●	●	○
Verifpal [45]	●	●	●	◐	●	◐	◐

Table 5.1 Comparison between Verifpal with the main symbolic security analysis tools [45].

5.3 T2S-MAKEP analysis

In this section, we check the robustness of the T2S-MAKEP protocol against the IoT known attacks, and in the second, we use Verifpal [45] as an automatic security verification tool, which is used to analyze the proposed protocol formally. Finally, its performance evaluation is given.

5.3.1 Informal security analysis

A defence assessment of each presented attack scenario is given in the following points.

- *Message Analysis Attack:* In our scheme, the exchanged data during the authentication steps is hashed by using a one-way hash function or encrypted using a secret key $h(k_{A,i})$. Also, the hashed values are not vulnerable to this attack, and the secret key $h(k_{A,i})$ is known only by the IoT device and the trusted server. Furthermore, the secret key changes after each session. Hence, this attack is not practical for our scheme.

- *Replay attack:* In our scheme, both the IoT devices and the trusted server include timestamps in exchanged messages, which helps prevent this kind of attack. Besides that, the used information (challenge, response, helper data and secret key) changes in each authentication phase, and the session key also changes for each session. Hence, the attack will not be possible.

- *DoS attack:* In our proposed protocol, each entity verifies the received packets immediately by computing a hash value of the received message. If the integrity of the message is compromised, the attempted communication is banned. Further, except for the authentication request where the verification is based on the presence of the $h(ID_A)$ on the server's database, the random guessing of the hash value is not applicable as each hashed value contains the secret key $h(k_{A,i})$ which is known only by the server and the device equipped with the right PUF. Further, the server is installed in a secure environment, such as a data center, which can avoid and resist this kind of attack using software and hardware solutions. As a result, the proposed protocol has shown good resistance against DoS attacks.

- *Physical attack:* As mentioned before, our scheme is based on PUFs, which is a perfect solution against this kind of attack, where the keys are generated and reproduced through performing PUF operations, so there is no need to store the secret key. Further, in our scheme, the IoT device does not store any information in its local memory. Hence, the IoT device does not reveal any secrets even if an adversary has physical access, making the proposed protocol secure against physical attacks.

- *MITM and Injection attack:* In our authentication mechanism, the IoT device and the server do not exchange sensitive data (such as $h(k_{A,I})$, or the new response $R_{A,i+1}$) in plaintext over the unsecured communication channel. Also, an adversary has no access to these secrets to calculate the right hashed value for passing the verification step. So, the proposed protocol is secure against MITM attacks.

- *Modeling attack:* In our mutual authentication scheme, we do not exchange the PUF response in plaintext, so if an adversary could capture the exchanged messages between the IoT device and the server, she/he could not run a machine learning attack since she/he possesses only the challenge that is exchanged in plaintext. Further, if an attacker has access to the trusted server, she/he can capture only one pair of CRPs. Therefore, launching a modelling attack on our proposed protocol is not applicable.

- *Helper data manipulation:* In our scheme it is securely generated by the server and transmitted to the IoT device. For each exchanged helper data, its hashed message with some other information are transmitted to help in verifying the integrity of the helper data on the IoT side. In addition, the manipulation can be executed only during the communication phase, where the helper data is transmitted to the IoT device. Consequently, a helper data manipulation attack is not feasible in our mutual authentication scheme.

- *Cloning attack:* Our proposed protocol resists cloning attacks since it exploits PUFs for authentication, and the main feature of PUF is unclonability as its name says. It has been shown that each PUF output is a unique value and it is hard to clone a PUF, due to the fact that any modification to the physical characteristics of the PUF circuit makes the cloned version a new instance of the PUF which results in generating different responses from the original PUF circuit.

- *Camouflage attacks:* In our scheme, before adding the device to the network, it goes first through the enrollment phase with the server in a trusted environment, where the initial authentication credentials $(C_{A,i}, h(K_{A,i}), P_{A,i})$ were generated using embedded PUF component, and stored on the server. Further, the credentials are unique for each IoT device since they are extracted from the silicon PUF, which is a good solution for anti-counterfeiting [170]. So, even if an attacker inserts a forgery device equipped with a PUF, a camouflage attack cannot succeed as it does not figure out the server's database.

- *Anonymity Device:* In our scheme, the identity of the device is encrypted using a one-way hashing function that makes it difficult to recover the device identity from the hashing results.

5.3.2 Formal security analysis

To model and check how much our developed protocols are secure using Verifpal, we must first define whether our schemes will be analyzed under a passive or active attacker. A passive attacker can observe the communication messages over the network and cannot inject or modify these messages. Contrarily, an active attacker can observe, modify, and inject new messages. In our case, we chose the active one who can alter the authentication messages and inject its data. Secondly, we have to define the different principles (devices and the trusted server) taking part in our communication system, including the active attacker. Then, we describe the authentication messages being communicated between the model's parts across the network. Finally, we query Verifpal the following queries about the integrity, privacy, and freshness of the messages exchanged and the authentication of the participants. The Verifpal code that was utilized to analyze T2S-MAKEP is presented in A.2.3.

- **Integrity:** In each step, after receiving the message by the device or the server, we check the integrity of the received packet by computing its hash function. To check this property, we use `ASSERT` a Verifpal's predefined primitive. `msg_1` checking integrity using Verifpal is illustrated as follows.

```
ASSERT(HASH(ID_a, TS_1),msg_1)?
```

  - **Reachability and Secrecy:** The first objective of T2S-MAKEP protocol is to guarantee the reachability and secrecy of the shared secrets between the IoT end-device and the trusted server. Using Verifpal, we show that the considered secrets id_a, h_{ka1} and r_{a2} are secretly shared between $Thing - A$ and the server (i.e. these secrets cannot be obtained, deleted or altered by an active attacker). To test this property, the following variables are declared.

$$\textit{knows private } id_a$$
$$\textit{knows private } h_{ka1}$$
$$\textit{knows private } r_{a2}$$

    $knows\ private$ is a Verifpal's predefined primitive used to declare private information. The first variable is declared private to guarantee the anonymity of Thing-A identity. The second is the secret and the session key. The last one is the new generated response used to generate the new secret key that will be used for future authentication. At the end of the model, we ask Verifpal for the next confidentiality query of the privates used information. The confidentiality query is the most basic of all Verifpal queries, which are used to test if an attacker could obtain the private information communicated between $Thing - A$ and the server during the authentication phase.

Those queries are expressed as follows to check the confidentiality of id_a, $h\_k\_a\_1$, and r_a_2.

confidentiality? id_a

confidentiality? $h\_k\_a\_1$

confidentiality? r_a_2

– **Mutual Authentication:** The second objective of the proposed T2S-MAKEP scheme is the mutual authentication between the IoT device and the server, which adds more resistance to the protocol against man-in-the-middle and replay attacks. To test this security property under Verifpal, we use the following two authentication queries:

authentication? Trusted_server $\longrightarrow$ *Ting_a: msg1*

authentication? Thing_a $\longrightarrow$ *Trusted_server: msg2*

The first query checks if $Thing_a$ can authenticate the $Trusted\_server$ based on the received message $msg1$ in step (2) of T2S-MAKEP scheme. This operation succeeds because $msg1$ results from a one-way hash function, which contains the secret key h_{ka1} known only by the server and is communicated in the setup phase. So, this secret information guarantees that $msg1$ comes from an authentic entity, the $Trusted\_server$. Also, in the second authentication query, the server authenticates $Thing_a$ based on the value of the secret key that is reproduced using FE on the noisy generated response. This step succeeds because only $Thing\_a$ could use its PUF to regenerate the same stored key corresponding to the stored one on the server. Both queries guarantee mutual Authentication.

– **Freshness:** In addition to the above-specified security properties, freshness of the transmitted messages is also a pillar requirement. Freshness query is useful for detecting replay attacks, where an attacker could manipulate one message to make it seems valid in two different contexts. To assert this property, the following Verifpal queries are declared, where each tested message contains a timestamp value that guarantees the freshness of data.

freshness? msg1

freshness? msg2

freshness? msg3

The verification results of the formal security analysis of the proposed protocol are shown by the execution of the Verifpal code given in appendix A.2.3. Fig. 5.1 shows that all the quires were passed successfully. This means that it is infeasible for active

attackers to decrypt messages without secret keys and either to get the secret session key used between the IoT device and the trusted server. Further, an intruder cannot launch MITM and injection attacks due to the freshness of the timestamp value. Finally, both the IoT device and the trusted server can authenticate each other based on the secret key generated using PUF. Consequently, the mutual authentication protocol using PUF is secure and guaranteed.

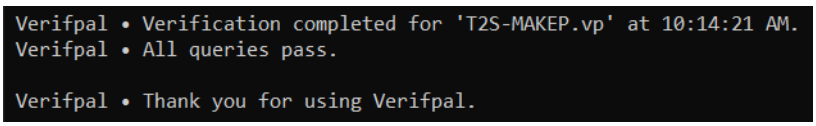

Fig. 5.1 T2S-MAKEP's Verifpal outputs.

5.4 T2T-MAKEP analysis

5.4.1 Informal security analysis

A defence assessment of each presented attack scenario is given in the following points.

- *Resisting to message analysis attack.* In this schemes, secret keys, session keys, and responses are confidential and cannot be accessible by an attacker despite the fact the possibility of intercepting the authentication messages. This is achieved by encryption and hashing the transmitted messages and not storing them locally.

- *Resisting to replay attack:* By considering an attacker was able to intercept old authentication messages and want to replay them, she/he cannot forge the current transmitted message since a valid timestamp is assigned at each transmitted message. Further, except for the identity of IoT end devices, all parameters are updated after each new session. Consequently, replay attacks are prevented efficiently.

- *Resisting to denial of service attacks:* Only two types of entities could be found in both proposed protocols, a server and the IoT end-device. However, the DOS attack is infeasible on the server side due to its high computation power [34]. So, this attack is considered only on the ToT end-device side. The attack can be done in **step (2)** and **step (3)** for the T2T-MAKEP scheme. In each step, after receiving the message by the device, it checks the integrity of the received packet by computing its one hash function. Further, the probability that the random guessing of the hashing value to pass the verification process is negligible due to the confidentiality of the secret key included in each hashed value. As a result, the Dos attack is not practical in our scheme.

- *Resisting physical attack:* In this scheme, IoT end-devices, do not store any secret or sensitive information in their local memory. Further, one of the assumptions of the target system model is that the communication between the PUF circuit and the IoT device's micro-controller is considered secure. Thus, making our proposed protocols secure against physical attacks, even if An attacker captures ioT end-devices.

- *MITM and Injection attack:* All the authentication messages do not include any secret or sensitive information in plaintext in our proposed protocols. The only exchanged data in plaintext during authentication are the thing's identity, timestamps, challenges and public helper data. Also, any secret is hashed or encrypted using a one-way hashing function or a generated secret key known only by the authentication entities. A MITM attack will not benefit from any captured data in such settings. Thus, the proposed protocol is secure against this type of attack.

- *Modeling attack:* This type of attack is based on capturing a set of CRPs. In our mechanism, only the challenge was communicated from the server to the IoT end-devices in plaintext. So, even if an attacker could intercept the authentication's messages, she/he will see only the challenge, which will be useless for launching a modelling attack. This mechanism makes this attack not applicable in this scheme.

- *Helper data manipulation:* In our mechanism, the helper data is securely stored on the server and transmitted to the IoT end-devices in plaintext to reproduce secret keys. So, this parameter can be manipulated only during its transmission. Further, each helper data is transmitted with a hashed message of helper data and other additional information. The hashed value guarantees that the public helper data was not manipulated during the transmission. So, this attack is not feasible with T2T-MAKEP.

5.4.2 Formal security analysis

This section presents the security analysis for T2T-MAKEP. This analysis follows the same verification and security properties specification as the same analysis paradigm of T2S-MAKEP. We do not repeat the T2S-MAKEP authentication step between $Thing - A$ and the trusted server to avoid repetitions. Thus, we consider that $Thing - A$ is successfully authenticated, and our analysis started from Step (2) of T2T-MAKEP scheme. The Verifpal code that was utilized to analyze T2T-MAKEP is presented in A.2.3.

As shown in Figure 5.3, in addition to the IoT end-device identity id_a, h_{ka1} and h_{kb1} that are secret keys of $Thing_a$ and $Thing_b$ respectively; SAB and SBA are equals calculated session keys between both devices ($SAB = SBA$). After the declaration of

these device, we check their confidentiality using the confidentiality query. Further, by using the authentication query, we verify the mutual authentication property between the participants in our scheme.

```
  queries[
   authentication? Trusted_server -> Ting_a: m2
   authentication? Ting_a -> Ting_b: m4
   authentication? Trusted_server -> Ting_a: m3[]
   authentication? Ting_a -> Ting_b: m5[]
   confidentiality? id_a
   confidentiality? s_a
   confidentiality? s_b
   confidentiality? h_k_b_1
   confidentiality? h_k_a_1
   freshness? m2
   freshness? m4
   equivalence? s_a,s_b
]
```

Fig. 5.2 T2T-MAKEP's Verifpal queries code.

Based on the exchanged m_2 and m_4, and the encrypted ones m_3 and m_5, we check the freshness of m_2 and m_4 using the freshness query. Then, we verify if both IoT end-devices could calculate the same secret session key S, using the equivalence query between the generated keys (SAB) and (SBA) by $Thing_a$ by $Thing_b$, respectively.

Figure 5.3 shows that T2T-MAKEP has successfully achieved mutual authentication between the IoT end-devices. Further, the confidentiality of secret information and the freshness of the messages are guaranteed. In addition, the generation of the same and unique secret session key generation is satisfied by both devices.

```
Verifpal • Verification completed for 'T2T-MAKEP.vp' at 02:17:08 PM.
Verifpal • All queries pass.

Verifpal • Thank you for using Verifpal.
```

Fig. 5.3 T2T-MAKEP's Verifpal outputs.

5.5 LT2S-MAKEP analysis

5.5.1 Informal security analysis

- *Modeling Attacks:* The protocol is strong against machine learning attacks especially it ensures mutual authentication between devices and the provers/server. An attacker cannot impersonate the server either the device. Even an attacker try to generate challenge, it cannot get any information regarding responses. Additionally, the protocol uses two classes of CRPs from a weak and strong PUFs. Get any information from one does not infer piece of information even if the helper data is public.

- *Spoofing Attack:* Since the protocol is strong against modelling attack, an intruder cannot impersonate a device or decrypt an intercepted session.

- *MITM Attack:* An attacker impersonating the device and tries to initiates an authentication attempt cannot succeed since the server verifies the authenticity of the device through m_4. Similarly, if it wants to impersonate the server, a device verifies the identity using m_2 thanks to ***SRAM***.

- *DoS Attack:* A successful DoS attack can be only if a soft model is constructed and this is not possible since the responses are encrypted. In addition, if does the identified not found in the data base then the attacker is banned.

5.5.2 Formal security analysis

The full Verifpal code that was utilized to analyze LT2S-MAKEP is presented in A.2.3.

- **Confidentiality:** the first objective of our protocol is guaranteeing the confidentiality of the shared and communicated secrets between the IoT device and the server. In our scheme K_{SRAM}, $R_{SRAM}, R_{APUF}, K_{APUF}, R'_{APUF}$ are considered as secret information for the attacker, but are known by the IoT device and the server. Using Verifpal, we show that the active attacker could not have access to the considered secrets information. To test this property, we use $knows\ private$ a Verifpal's predefined primitive, and the secret information is declared as follows:

  knows private K_SRAM, R_SRAM, R_APUF, K_APUF, R'_APUF

  After that, we ask Verifpal the next confidentiality query of the privates declared information. This query test if an attacker could obtain the private information communicated between the IoT device and the server during the authentication phase. One example of these queries is expressed as follows to check the confidentiality of ***K_SRAM***.

  confidentiality? K_SRAM

- **Mutual Authentication:** The second objective of the proposed protocol is to ensure mutual authentication between the IoT device and the server. After describing the transmitted messages during the authentication phase, we use the authentication query of Verifpal using the following two authentication queries:

  authentication? Server $\longrightarrow$ Device: msg_2

  authentication? Device $\longrightarrow$ server: msg_4

The first query checks if the device can authenticate the server, upon receiving the message msg_2, and the second asserts if the server authenticates the device through msg_2. Both queries guarantee the mutual Authentication. The first one attests that the secret information K_SRAM is known only by the server, so the IoT device can authenticate the server based on msg_2 and in the second msg_4 contains R_APUF which is can be generated only by the device using the $APUF$.

- **Freshness:** In addition to the above correspondence security proofs, freshness of the transmitted messages is also evaluated using Verifpal. Using the freshness query helps to detect replay attacks, where an attacker could manipulate one message, and send it at different time. Using timestamp variable avoids this type of attacks, an example of testing the freshness of msg_2 using freshness query is as follows.

freshness? msg_2

By executing the Verifpal code, the results of the formal security analysis of the proposed protocol are displayed. Fig. 5.4 shows that all questions were satisfactorily answered. This means that active attackers cannot decrypt messages without secret keys and cannot obtain the secret session key used between the IoT device and the trustworthy server. In addition, an intruder cannot conduct MITM and injection attacks because the timestamp value is current. Lastly, both the IoT device and the trusted server are able to authenticate each other using the PUF-generated secret key. Therefore, the PUF-based mutual authentication technique is secure and guaranteed.

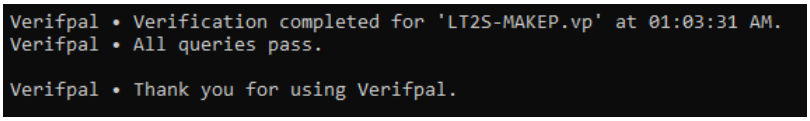

Fig. 5.4 LT2S-MAKEP's Verifpal outputs.

5.6 Comparison

With respect to the existing protocols, Table 5.2 compares our developed protocols to the recently deployed ones by considering their capabilities to avoid the above discussed attacks in Section 5.3.1. In the following comparison, ✓ means that the protocol is not secure against the indicated attack. ✗ means the inverse of the previous statements regarding the attacks. Also, **?** means that the authors of the cited contribution did not express any purpose regarding the selected attack.

Work	Replay Attack	Message Analysis Attack	Physical attack	DoS Attack	Modeling attacks	MITM attacks	Cloning attack	Injection attack	Helper data manipulation	Anonymity Device	Anonymity server
(author?) [33]	✗	✓	✗	✗	✗	✗	✗	✗	?	✗	✗
(author?) [34]	✗	✗	✓	✗	✗	✗	✗	✗	✓	✓	✗
(author?) [35]	✓	✓	✗	✓	✓	✓	✗	✓	?	✗	✗
(author?) [36]	✗	✗	✓	✗	✗	✗	✗	✗	✓	✗	✗
(author?) [37]	✓	✓	✗	✓	✓	✓	✗	?	?	✗	✗
(author?) [38]	✓	✓	✓	✗	✗	✗	✗	✓	✓	✓	**?**
(author?) [39]	✗	✗	✗	✗	✗	✗	✗	✗	?	✗	?
(author?) [40]	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	?
(author?) [41]	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
(author?) [42]	✗	✗	✗	✓	✗	✗	✗	✗	?	✗	?
(author?) [43]	✗	✗	✓	✗	✗	✗	✗	✗	?	✗	✗
(author?) [44]	✗	✗	✗	✗	✗	✗	✗	✗	?	✗	?
(author?) [171]	✗	✗	✗	✗	✗	✗	✗	?	?	✓	✗
(author?) [172]	✗	✗	✗	✓	✗	✗	✗	✗	✓	?	✗
T2S-MAKEP	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
T2T-MAKEP	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
LT2S-MAKEP	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗

Table 5.2 Security of the proposed PUF-Based authentication Protocols.

T2S-MAKEP, T2T-MAKEP, and LT2S-MAKEP are robust against the attacks presented in Section 5.1. This strength comes from using PUF without storing any information (secret or not) in the local memory of the IoT end devices. Rather than, no sensitive information is transmitted in clear during the authentication process of both proposed protocols.

In the second step, we run a comparison between our proposed schemes and the existing work reviewed in Section 3.2.5. Our comparison summarised in Table 5.3 is based on the following characteristics. ✓ means that the protocol considers the characteristics. However, ✗ means the inverse of the previous statements.

1. *T2T and T2S schemes* to indicate if the considered architecture supports one of both authentication schemes.

2. *PUF-based authentication* shows if the referred protocol uses PUF only as a security primitive to achieve authentication successfully or it needs other computational primitives like the elliptic curve. We do not consider one-way functions,

hashing and XORing as complex security primitives.

3. *Formal/informal security analysis* this criteria specify the type of the analysis. Formal relies on sound mathematics approaches to show the correctness of the proposed protocol, whereas informal refers only to a textual description.

4. *Mutual authentication* shows if the protocol supports mutual authentication between all authenticated entities.

5. *Session key generation* indicates if the referred protocol generates a session key to secure the communication after each successful authentication.

6. *Practicality* checks if the protocol is portable, scalable, and robust.

7. *Error correction* indicates if errors and noise in responses are considered.

8. *Thing storage* shows if the IoT device does not store any secret or non-secret information that helps to accomplish the authentication process.

9. *Confidentiality* indicates if the protocol's secret information is kept confidential during the authentication phase.

10. *Freshness* checks if the exchanged messages are received with respect to precise variables like timestamp, which help not to use an old message in a new authentication.

We found that most of the studied IoT protocols are designed for the T2S authentication protocol scheme, except two [36, 173] that support T2T and only one ([119]) covers both T2S and T2T. From a technical point of view, most of the protocols use one-way and hashing functions to ensure the confidentiality of secret information. Instead of using these two functions, two contributions [38, 41] use another cryptographic algorithm such as Elliptic Curve. In terms of correctness validation and performance evaluation, most of the reviewed protocols do not apply formal and informal security analysis techniques. Regarding the authentication, some of them do not offer mutual authentication [33, 37, 35, 38, 42], nor generate a session key for securing the communication after each successful authentication operation [33, 37, 35, 38, 41, 42]. Further, most of the studied schemes do not consider error correction, which plays a vital role in the practicality of a protocol. Also, for the leakage resilience, we observe that some of the protocols store sensitive information on the local memory of the IoT device. In addition, most of the reviewed protocols ensure the confidentiality of the transmitted secret information but less guarantee the freshness of the transmitted messages.

Table 5.3 Comparison's summary between T2T-MAKEP and the reviewed PUF-based authentication protocols

Work	T2S scheme	T2T scheme	PUF-based authentication	Formal security analysis	Informal security analysis	Mutual authentication	Session key generation	Practicality	Error correction	Thing storage	confidentiality	Freshness
(author?) [33]	✓	✗	✓	✗	✓	✗	✗	✗	✗	✓	✓	✗
(author?) [34]	✓	✗	✓	✗	✓	✓	✓	✗	✓	✗	✓	✓
(author?) [35]	✓	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗
(author?) [36]	✗	✓	✓	✓	✗	✓	✓	✗	✓	✗	✓	✓
(author?) [37]	✓	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗
(author?) [38]	✓	✗	✗	✗	✗	✗	✗	✓	✓	✗	✓	✗
(author?) [39]	✓	✗	✓	✗	✗	✓	✓	✗	✗	✓	✓	✗
(author?) [40]	✓	✗	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓
(author?) [41]	✓	✗	✗	✓	✗	✓	✗	✗	✗	✓	✓	✓
(author?) [42]	✓	✗	✓	✗	✗	✗	✗	✗	✗	✓	✓	✗
(author?) [43]	✓	✗	✓	✗	✗	✓	✓	✗	✗	✗	✓	✗
(author?) [44]	✓	✗	✓	✓	✓	✓	✓	✗	✗	✓	✓	✓
(author?) [173]	✗	✓	✓	✓	✗	✓	✓	✗	✓	✗	✓	✓
(author?) [119]	✓	✓	✓	✓	✓	✓	✓	✗	✗	✓	✓	✓
T2T-MAKEP	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
T2S-MAKEP	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
LT2S-MAKEP	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓

5.7 Conclusion

In this chapter, the security analysis of the proposed authentication protocols is given. For each proposed scheme, we gave both informal and formal security analysis as well as evaluations of how well it would work. In the informal analysis, we study and verify the robustness of each proposed protocol against the existing IoT PUF-based authentication ones. Then, Verifpal, the formal verification for cryptographic protocols, is used to prove formally the correctness and show the security of the proposed protocols.

Finally, we compared our proposed schemes with the works presented in section 3.2.5. In the next chapter, the implementation of the proposed schemes within the experimental results will be presented.

CHAPTER 6

IMPLEMENTATION, EXPERIMENTATION AND SIMULATION

In this chapter, we shall to show the effectiveness of the developed protocols. To implement our proposed IoT PUF-based authentication protocols, we first design then simulate the arbiter PUF that will be embedded within the IoT circuit of a given device. Before bringing the constructed PUF into operation, it is required to evaluate the generated CRP for each simulated APUF instance in order to determine its performance. In addition, it is essential to implement error correction and noise elimination methods to assure its capacity to produce steady and dependable output. At this point, the selected and evaluated APUF can be integrated into the IoT device and utilized by it. Finlay, we study and evaluate the performance of the proposed protocols regarding a given scenarios.

The main focus of this chapter is:

1. Implementation and evaluation of a 16-arbiter PUF.
2. Presentation and implementation of the noise elimination process and PUF response reproduction.
3. Presentation the performance evaluation on the real scenario using Unmanned Aerial Vehicles.

6.1 Arbiter PUF Design

Currently, we lack access to hardware components such as ASIC and FPGA fabrication, which are often used to propel PUF. We employ simulation as an alternative to gaining experience, simulating an arbiter PUF with the PyPuf simulator. In this part, we will first introduce the PyPuf simulator, followed by the methods required to simulate an arbiter PUF. Then, we generate the necessary CRP sets. In Finlay, we use PUF metrics to evaluate the performance of the simulated APUF.

6.1.1 PyPuf Simulation

PyPuf [174] is a a toolbox for simulation, testing, and attacking Physically Unclonable Functions. PyPuf is a Python-based Physical Unclonable Function, as suggested by

its name, which is derived from the programming language Python and the acronym for Physical Unclonable Function. PyPuf is open-source software distributed under the GNU GPLv3 and is accessible to everybody via [1]. PyPuf includes three modules. Simulation, learning, and experimenting are examples. Using the experimentation model, reproducible experiments can be conducted. The learning model is used to evaluate PyPUF's resistance against modification attacks. This is achieved by employing machine learning-based attacks. In our instance, however, we utilize only the first model, which is the simulation, which allows us to simulate a variety of PUF topologies.

In terms of topologies, PyPuf focuses on some PUF constructions introduced in section 3.1.1.1, such as the APUF, and the XOR PUF. These PUF constructions are simulated using a broad class called *LTF_array* that creates arrays of Linear Threshold Functions (LTFs). In the book Analysis of Boolean Functions [175] Ryan O'Donnell goes into great detail about what LTFs are, but in this context it is enough to know that LTFs are useful to simulate the above mentioned PUF constructions. Thus far, this report has mostly used the 0, 1 bit notation, indicating false and true, respectively. This bit notation excels at describing relative voltages, which is, of course, very common in electronics. This notation is supported in PyPuf, but that appears to be the case mostly for compatibility reasons. The more common bit notation in PyPuf is the 1, 1 bit notation, indicating False and True, respectively. Specifically, note that 1 corresponds to False in this notation. This seems to be the notation that O'Donnell prefers using as well. A plausible reason for using this notation is that notation choice influences required mathematical operations to perform certain calculations.

6.1.2 Experience Setup

In our experience, the arbiter PUF construction has been chosen from all the constructions offered by PyPuf since it is the basis of all other existing strong PUF designs. We simulate a 16-APUF, which means an arbiter PUF with 16 delay stages (16 bits as inputs) and one bit as output. Each stage is composed of two 2-to-1 multiplexers as shown in Figure 6.1. A rising pulse at the input propagates through two nominally identical delay paths. The paths for the input pulse are controlled by the switching elements, which are set by the bits of the challenge. For c = 0, the paths go straight through, while for c = 1, they are crossed. Because of manufacturing variations, however, there is a delay difference between the paths. An arbiter at the end generates a response of '0' or '1' depending on the difference in arrival times.

Unfortunately, the output of the simulated 16-APUF could not be used directly in our proposed authentication protocol because it generates only one bit as a response, either zero 0 or one 1, which is impractical in the authentication process because a third

[1] https://github.com/nils-wisiol/pypuf

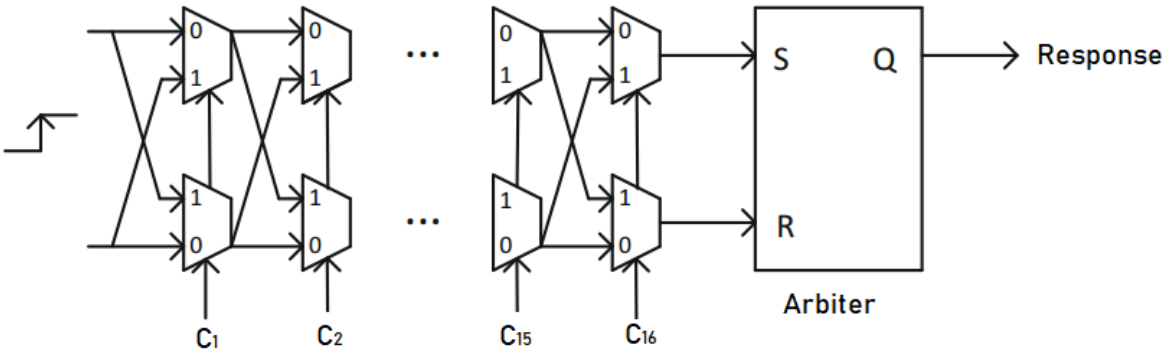

Fig. 6.1 A generic structure of an arbiter PUF with 16 stages.

party (attacker) can easily break the authentication operation with this configuration. To avoid this scenario, we have learned to generate a 12-bit response. For this, several adjustments must be made to the existing PyPuf. By simulating 12 instances of the 16-APUF in parallel and by concatenating the output of each instance with other, it is possible to get replies of 12 bits (Figure 6.2).

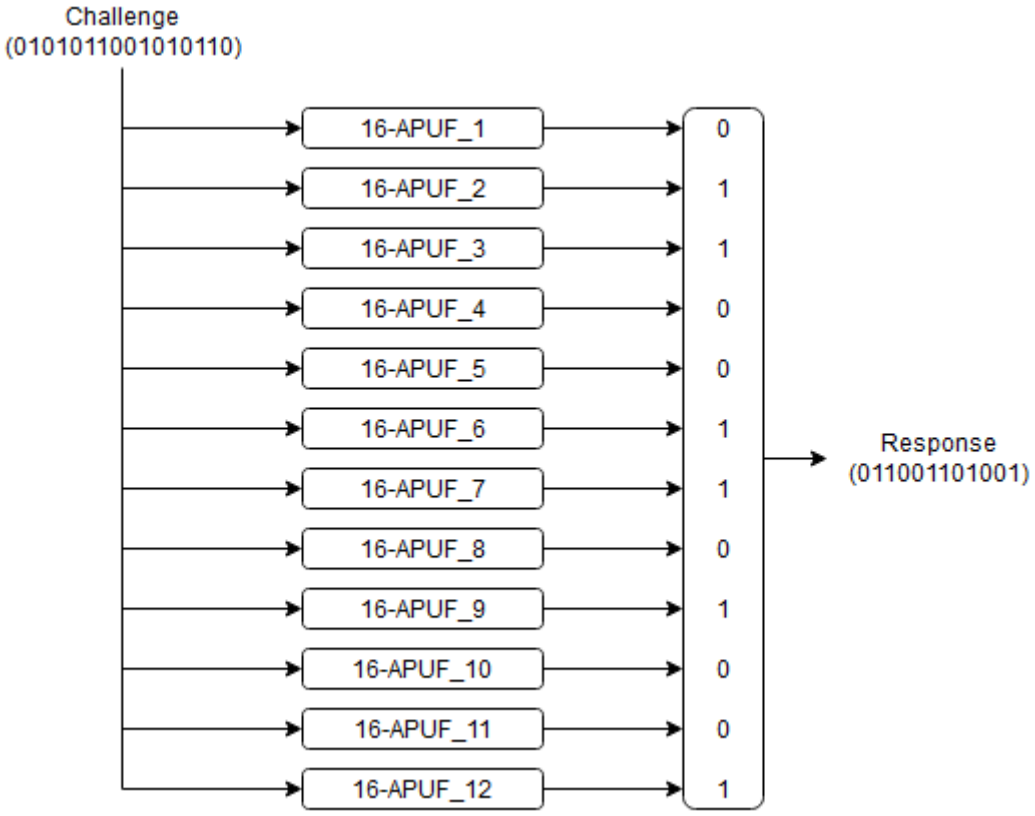

Fig. 6.2 12 bit response generation using 16-APUF.

6.1.3 CRPs Generation

After adding the needed module to the existing PyPuf, we can easily simulate a PUF component that could generate a 12 bit response. According to the given scenario, we have three IoT devices in the described network, so we need to simulate 3*12*16-Apuf, which means 36 16-APUF instances. Each instance generates a one-bit response from a 10000 challenge of 16 bit length. This means we have 10000 response bits generated from each 16-APUF instance. To test the effect of noise on the simulated PUF, the same challenge is subjected to the $12 - st$ 16-APUF with the noise parameter of PyPuf (noisiness).

6.1.4 16-APUF Evaluation

Before we use the simulated 16-APUF in our proposed authentication protocols, we evaluate its performance first. By checking the presented metrics in section 2.2.2, we evaluate their uniqueness, steadiness, randomness, correctness, bit aliasing, uniformity, and diffuseness. To evaluate reliability, we obtained 12-bit length responses at noise value of 0, 0.1 and 0.3 and compared them to see if they were identical. A reference response was obtained at noisiness = 0.Following these metrics, table 6.1 and Figure 6.3 show the results of this evaluation.

Metrics (%)	**Result**	**Ideal value (%)**
Uniqueness	48.07	50
Randomness	99.8	100
Bit Aliasing	49.93	50
Uniformity	49.93	100
Reliability	97.96	100

Table 6.1 Evaluation results.

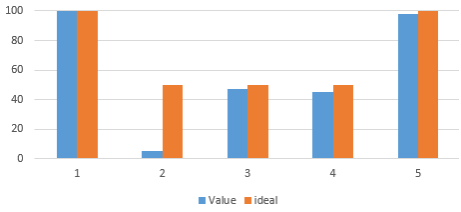

Fig. 6.3 16-APUF Evaluation results.

As illustrated in Table 6.1, it was clear that the four metrics uniformity, uniqueness, bit-aliasing and the randomness of the generated CRPs data are closer to the ideal values. It is well observed that both values of uniformity and bit-aliasing are equivalent which confirm the existing results in the literature. Also, the reliability of the used APUF is 97.96%, where the ideal value is 100%. This means that all 30 APUF instances generate a near stable response at different noise parameter 0, 0.1, and 0.3. Using the same challenge as an input, this simulated 16-APUF outputs a noisy response that has some error bits, compared to the reference response (initial response), so fuzzy extractor operation must be executed to eliminate this noise and generate a stable and reproducible response.

6.2 Noise Elimination and Response Reproduction

To show the impracticability of using the PUF response directly in the authentication operation directly without noise elimination process, let us consider the task of biometric data authentication. As a user, Alice uses her biometric data (e.g., iris) as a unique identity w when she wants to gain access to her account. A trusted server stores some information $y = f(w)$ about the password. When Alice enters w, the server lets Alice if only if $f(w) = y$. In this naive authentication scheme, we accept that Alice could use w itself as her identity for the verification process. Hence, two issues can be found with this authentication process. First, when Alice reused her biometric information w' at a latter authentication step, the new generated identity w' will differ from the initial one. Unfortunately, it is close to it, but not equal to the initial value w, due to some noise on w'. In fact, she will not be able to recover her original identity by using a standard encryption scheme. Second, w is not uniform, and it cannot be used as a cryptographic key in standard encryption schemes.

6.2.1 Noise elimination process

Similarly, one of the biggest challenges with PUF is to guarantee the stability of the key in each regeneration phase, which is caused by the environmental variables. This since the outputs of PUF are affected by the environment conditions, and To achieve the stability (zero error rate) of the PUF response, it usually needs to be post-processed. The latter is generally composed of two phases: *enrollment* and *reconstruction* (see Figure 6.4).

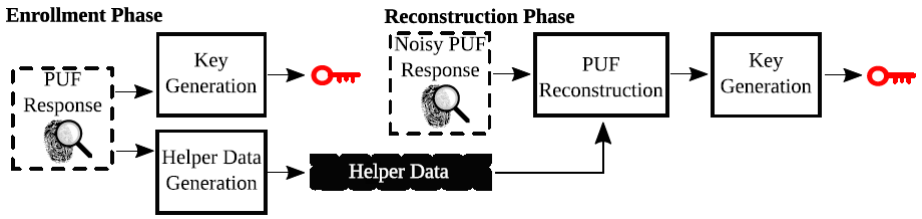

Fig. 6.4 Enrollment and reconstruction phases.

During the enrollment phase, the unique identifier (or the key) is generated from the initial PUF response. At the same time, the public helper data is also generated. Clearly, the key has to be kept secret as it is used in cryptographic applications. On the other hand, if the helper data does not reveal information about the secret, it can be classified as public and stored in non-secure non-volatile memory. At the reconstruction phase, the stored public helper data is used to recover the enrolled key and to correct the bit errors that occurred in the actual PUF response (noisy response). This can be achieved since the helper data content describing the secret key allows a reliable key reconstruction, even under the variations of the environmental conditions.

Several approaches have been used to correct an error in the PUF response and generate a unique reproducible key such as Fuzzy Commitment, fuzzy vault, Code-offset fuzzy extractor, Syndrome construction, Parity construction, and Systematic low leakage coding. [176].

Since the reconstruction procedure of the initial PUF response from the noisy one depends mainly on the distance between both responses. From the three metrics Set difference, Edit and Hamming metric, we focus on the latter one, since it is used in the generation of PUF-based keys. Further, the main building block of fuzzy extractor construction is the secure sketch (See annexe A.2.3). In other words, in the PUF-based key generation process, a secure sketch is used to correct errors in the noisy PUF output using the Helper Data. Thus, the use of error-correcting codes (ECC) is one of the solutions. Many possibilities exist to realize a Secure Sketch based on error-correcting codes like syndrome construction, code-offset construction, index-based syndrome coding, complementary index-based syndrome coding, and differential sequence coding.

As our experience is based on the binary string response, that allows the use of the Hamming distance metric, syndrome construction is the prevalent secure sketch in a PUF-based key generation. During Syndrome construction , compute $s = w.H^T$, where H^T is the transposed parity-check-matrix of the used linear error correction code C. For recovery, compute $s' = w'.H^T$. Determine $e = locate(s' \oplus s)$ by using the error-location algorithm $locate()$ of the code C. Then, $w^* = Rec(w', s) = w' \oplus e$ with $w^* = w$ if $d(w', w) \leq t$. The syndrome construction does not require extra input random *codeword* c to extract the helper data s.

6.2.2 Response reproduction process

In this section, we present the needed steps used by the Fuzzy extractor to accomplish PUF-based response regeneration operation using secure sketches and strong randomness extractors. Here, we use syndrome construction as secure sketch construction and hash function as strong randomness extractors function. From our generated data, we applied the reconstruction phase only on the 16-APUF where the noise parameter is added, this to show the regeneration process.

Let's consider the initial and reference PUF response from what the key is extracted the first time, $w = 100010010111$ with 12 digits, and for the error correction code lets refer to the used binary linear block code $(6, 3, 1)$ with the flowing transposed parity-check-matrix H^T (Matrix A.2.1).

In the enrollment phase, the secret string key and a public helper data s are extracted from the initial PUF response w, using a hash function for the first one as a strong randomness extractor (ext) and the sketching procedure (SS) for the second one. First, let us compute the helper data s where:$s = w.H^T$, with $(6, 3, 1)$. The initial response

needs to be segmented in message blocks of length $(n = 6)$ which results in four messages $\{m_0 = 100010, m_1 = 010111\}$. To compute s, we calculate s_i correspondent to each message m_i, where $s_i = m_i.H^T$. The syndrome s_0 of the first message m_0 is calculated as follow.

$$s_0 = m_0.H^T = (100010).\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} = 100 + 011 = 111$$

Hence, $s_0 = 111$ and by following the same steps we found $s_1 = 010$, and finally the helper data will be $s = (111, 010)$.

Second, for the secret key generation from the response w, lets use a universal hash function $sha1$ as a strong randomness extractor (ext), $ext(w) = sha1(w)$.

$$sha1(w) = sha1(100010010111) = fc85f381d98096c60b763230399e4e4c40bfa693$$

So, we have the secret string *key=fc85f381d98096c60b763230399e4e4c40bfa693*, and the public helper data is $s = (111, 010)$.

In the reconstruction phase, we use the stored helper data s to recover the initial response w from a noisy one w', and calculate the secret key key from the recovered response w. Lets consider a noisy response $w' = 110010010101$. First, we compute $s' = w'.H^T$. So, $s'_0 = m'_0.H^T = 110010.H^T = 101$, $s'_1 = 010101.H^T = 001$, and we consider $s' = (101, 001)$. Then, we evaluate $s'_0 \oplus s_0 = 101 \oplus 111 = 010$ to compute $e_0 = locate(010)$. $locate(010)$ uses Table A.1 to find the corresponding error pattern e_0 for the syndrome 010, which is $e_0 = 010000$. Then, $m^*_0 = m'_0 \oplus e_0 = 110010 \oplus 010000 = 100010 = m_0$.

By repeating the same steps to calculate m^*_0, $s'_1 \oplus s_1 = 001 \oplus 010 = 011$, and $e_1 = locate(011) = 000010$, then $m^*_1 = m'_1 \oplus e_1 = 010101 \oplus 000010 = 010111 = m_1$.

To reproduce and recover the secret information key, we concatenate m^*_0 and m^*_1 that gives w^* and by using $sha1$ we will have:
$sha1(w^*) = sha1(100010010111) = fc85f381d98096c60b763230399e4e4c40bfa693$
Finally, we can show that using the public helper data $s = (111, 010)$, the secret string *key=fc85f381d98096c60b76 3230399e4e4c40bfa693* has been recovered from a noisy response w'=110010010101.

6.3 Experiments and Performance

The implementation of a 16-APUF and the good results of the evaluation operation of the generated CRPs as well as fuzzy extractor guarantee the generation of a sable key from the used 16-APUF. The simulated 16-APUF can be used to test and implement the proposed protocols. In this section, we present a scenario in which the proposed protocols are compared to other works.

6.3.1 Scenario

The network model is described in Fig. 6.5, which consists of two participants: UAVs and a ground station. Each UAV is equipped with an integrated circuit consisting of an APUF. Any adversary that attempts to probe or alter the circuit of a captured UAV will irreversibly modify the slight physical variations in the integrated circuit, which in turn changes the PUF challenge-response mapping, or even destroys the PUF. Each UAV has limited resources due to the stringent constraints imposed by its size, weight, and power limitations. The ground station is a trusted party and has no limitations on resources.

During the enrolment phase, the ground station obtains an initial CRP from each UAV in a trusted and secure network. Once the UAV exchanges the initial CRP with the ground station, it can function independently. Thus, the ground station stores the CRP for each UAV, while the UAV does not store any secret information. Due to the long distance between the UAV and the operator and the lack of physical protection, the UAV can be easily captured by an adversary. The open nature of wireless communication can also enable the adversary to overhear, duplicate, corrupt, or alter the data. The goal of the adversary is to establish an authentication with the ground station without being detected to cause more financial and strategic damage.

6.3.2 T2S-MAKEP

In this section, the performance evaluation of the proposed protocol is analyzed in terms of computational complexity, communication overhead, and storage constraints. We compare T2S-MAKEP scheme to **(author?)** [172] and **(author?)** [34] since they consider our protocol's features like mutual authentication and error correction.

6.3.2.1 Computational Complexity

As illustrated in Table 6.2, the computational complexity consists of the primary operations and their frequency involved in completing the authentication process. In the proposed schemes, the main used operations are hashing, encryption/decryption, and PUF operations. Table 6.2 shows the time required by the IoT device and the trusted server to

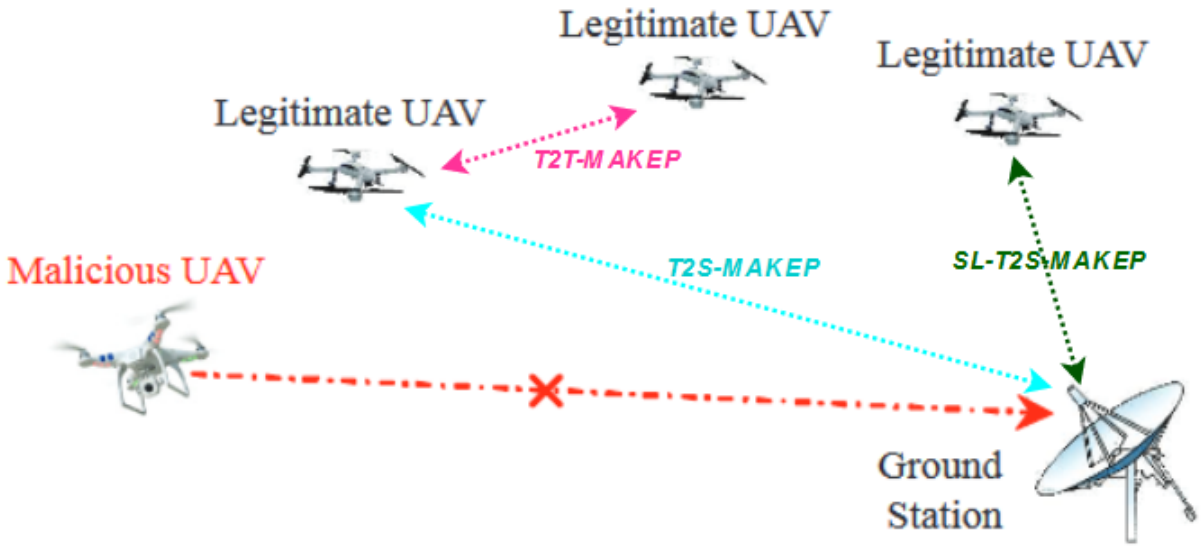

Fig. 6.5 UAV scenario network model

achieve the authentication steps of our proposed mechanism, **(author?)** scheme [172] and **(author?)** scheme [34] . The hashing time is denoted by N_H and N_{PUF} is the time for a PUF to produce a response, N_{ENC} is the time needed for encryption/decryption operations, and N_{FUZZ} represents the time of fuzzy extractor operations. We measure these values by counting their number of occurrences from T2S-MAKEP operations.

Works	IoT Device	Server
(author?)	5 N_H + 1 N_{FUZZ} +1 N_{PUF} +3 N_{ENC}	6 N_H+1 N_{FUZZ} + 3 N_{ENC}
(author?)	5 N_H + 1 N_{FUZZ} +2 N_{PUF}+3 N_{ENC}	5 N_H+1 N_{FUZZ}+3 N_{ENC}
T2S-MAKEP	5 N_H + 1 N_{FUZZ} +2 N_{PUF} +1 N_{ENC}	4 N_H+1 N_{FUZZ} +1 N_{ENC}

Table 6.2 T2S-MAKEP's computational complexity comparison.

Let's assume the use of a one-way hashing function such as *SHA-2* that has a worst-case running time of O(n), where n is the number of bits in the Hashing operation's outputs. Also, by using block ciphers, the average encryption and decryption times vary with respect to the key length, and its complexity can be considered O(n).

As illustrated in Fig. 6.6 and Fig. 6.7, both compared schemes have the same number of PUF and fuzzy extractor operations. The proposed protocols have a complexity of O(n) for the IoT devices as well as the server. Thus, the computational complexity of T2S-MAKEP mechanism is lower since hashing, encryption, and decryption operations in the presented schemes in Table 6.2 are higher than in our proposed one.

6.3.2.2 Communication Overhead

The communication overhead corresponds to the total number of transmitted bytes between the IoT device and the server during the authentication phase. To calculate the

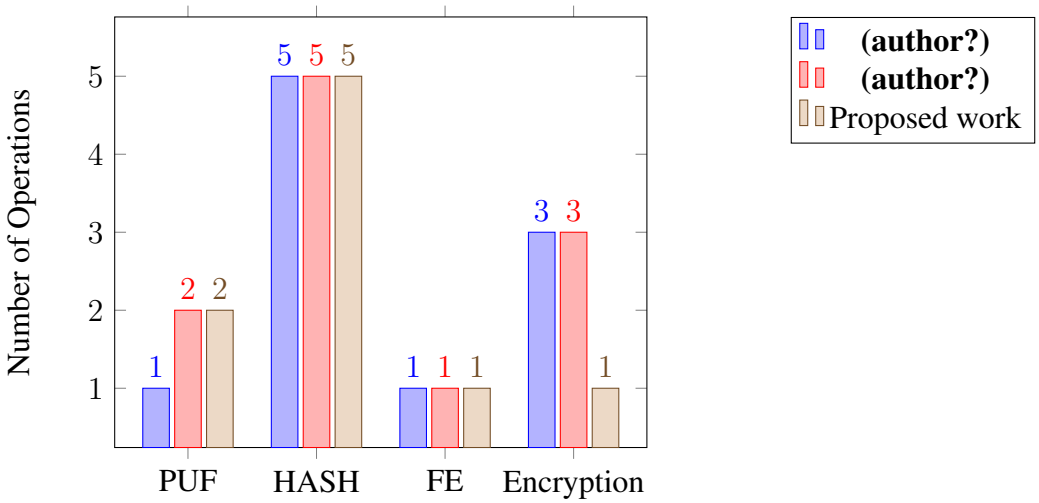

Fig. 6.6 The Computational Complexity on the device side.

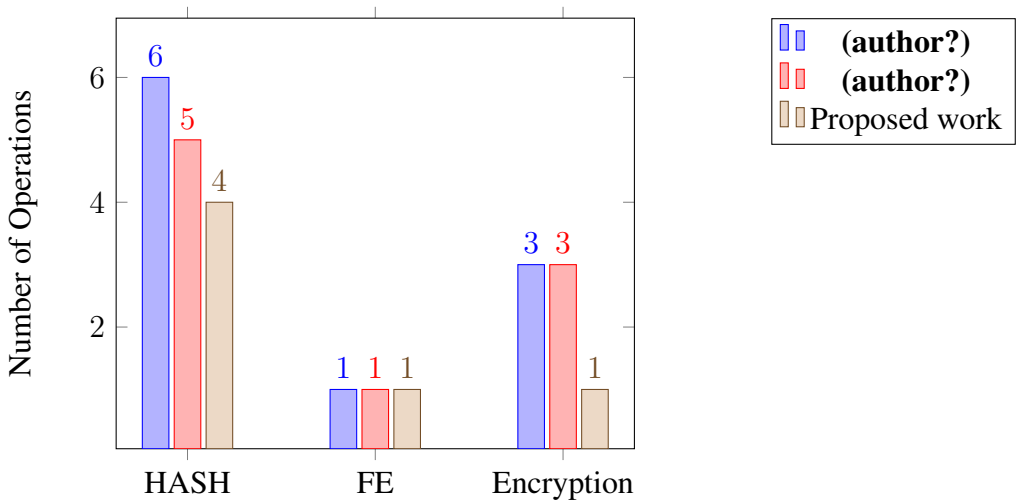

Fig. 6.7 The Computational Complexity on the server side.

length of each transmitted message, we use the message parameters that are communicated along with their sizes given in Table 6.5.

Message Parameters	Size in Bits
$Auth_{req}$	1
Hash function	256
Timestamps TS1, TS2, and TS3	48
Challenge	128
Helper data	64
Encrypted data	128

Table 6.3 Authentication messages' parameter values.

Based on the values in Table 6.5 and the protocol steps, we infer that the transmitted bytes in **step (1)**, **step (2)**, and **step (3)** of the authentication phase are 70, 78 and 54, respectively, which results-in 202 transmitted bytes between the IoT device and the server. The communication overhead of **(author?)**'s protocol is 248 bytes. As a result,the communication overhead in T2S-MAKEP scheme is lower than **(author?)**'s mechanism by 18.55%.

6.3.2.3 Storage Constraints

Compared to the most surveyed IoT PUF-based mutual authentication protocols that store a large number of CRPs on the server-side for each device, our proposed mutual authentication protocol is very efficient in terms of storage requirements. Rather than the IoT device does not store any secret or non-secret information, the server keeps only one CRP pair for each device's new authentication which makes our mechanism more scalable for a large number of IoT devices that could be deployed in the system. After establishing a secure connection, the server deletes the data used during the authentication process except for the device identity. Contrarily, the IoT devices in [34, 172] have a storage constraint: they have to keep secret information in their local memory.

6.3.3 T2T-MAKEP

6.3.3.1 Computational Complexity

The computational complexity of the authentication process is determined by the number of primary operations performed and their frequency. Initially, the fundamental operations of the proposed and compared schemes are hashing (N_H), encryption and decryption (N_E), PUF (N_P), and the error correction (N_F). T2T-MAKEP steps can be used to figure out how evaluating the identified criteria.

Table 6.4 shows the main operations carried out by IoT devices to achieve end-to-end authentication. We compare the proposed T2T-MAKEP's protocol with [36]

Table 6.4 T2T-MAKEP's computational complexity comparison.

Works	[36]	[119]	T2T-MAKEP
IoT Device A	$4N_H$+$1N_P$+$4N_E$+$1N_F$	$7N_H$+$2N_P$+$4N_E$	$8N_H$+$1N_P$+$4N_E$+$1N_F$
IoT Device B	$4N_H$+$1N_P$+$4N_E$+$1N_F$	$5N_H$+$2N_P$+$3N_E$	$4N_H$+$2N_P$+$3N_E$+$1N_F$

and [119] , which allow device-to-device authentication. Regarding error correction consideration, we can still conclude that our protocol has a slight advantage compared with the protocol in [119]. In general, the computation costs of T2T-MAKEP's scheme are similar to those of [36]. However, we do not consider the update phase in [36] that grows their operations considerably, making the computational complexity of the proposed protocols lower than the compared ones.

6.3.3.2 Communication Overhead

Total bytes sent and received during the authentication process is what we mean by "communication overhead". Table 6.5 lists the message parameters and their sizes, which we use to figure out the length of each message that's being sent.

Table 6.5 Authentication messages' parameter values.

Message Parameters	Size in Bits
$Auth_{req}$/Com_{req}	1
Hash function	256
Timestamps	48
Challenge	128
Helper data	64
Encrypted data	128
Device identity	48

Based on the values in Table 6.5 and the protocol's steps, we infer that the transmitted bytes in the T2S-MAKEP scheme are 186 bytes, which is higher than [177] and [178], which have similar transmitted bytes of 150 bytes.On the other hand, to calculate the communication overhead of the T2T-MAKEP protocol, we consider only the transmission between IoT end devices and compare it with the proposed mechanism in [36]. The communication overhead of the T2T-MAKEP protocol is 110 bytes, and 126 bytes for **(author?)**'s protocol. As a result, the communication overhead in our proposed mutual authentication scheme is lower than **(author?)**'s mechanism by 12.7%.

6.3.3.3 Storage Constraints

The same outcomes as T2S-MAKEP, as the T2T-MAKEP approach imposes no restrictions on the IoT device's storage for storing sensitive data. In addition, the server only retains a unique identifier for each IoT device in the system. Therefore, the server in

our suggested authentication process is better for a system with a large number of IoT devices.

6.3.4 LT2S-MAKEP

6.3.4.1 Computational Complexity

Table 6.6 shows the time required by the IoT device and the trusted server to achieve the authentication steps in LT2S-MAKEP proposed scheme and two similar other schemes presented in [172] and [34]. Where, both compared schemes have the same number of PUF and fuzzy extractor operations. LT2S-MAKEP have a complexity of O(n) for the IoT devices as well as the server. Thus, the computational complexity of LT2S-MAKEP is lower since no fuzzy extraction is required and the encryption operations are higher than in LT2S-MAKEP.

The table 6.6 compares the amount of time required by the IoT device and the trusted server to complete the authentication procedures for the LT2S-MAKEP scheme and two similar schemes provided in [172] and [34]. In instances where both competing schemes have fewer PUF operations than our suggested scheme. LT2S-MAKEP has an O(n) complexity for both IoT devices and the server. Therefore, LT2S-MAKEP is simpler to compute as it does not require fuzzy extraction. In contrast, encryption processes are more difficult with LT2S-MAKEP.

Works	IoT Device	Server
(author?)	5 N_H + 1 N_{FUZZ} +1 N_{PUF} +3 N_{ENC}	6 N_H+1 N_{FUZZ} + 3 N_{ENC}
(author?)	5 N_H + 1 N_{FUZZ} +2 N_{PUF}+3 N_{ENC}	5 N_H+1 N_{FUZZ}+3 N_{ENC}
LT2S-MAKEP	7 N_H +3 N_{PUF} +2 N_{ENC}	6 N_H +2 N_{ENC}

Table 6.6 LT2S-MAKEP's computational complexity comparison.

6.3.4.2 Communication Overhead

Based on the values in Table 6.5 and the protocol steps, we infer that the transmitted bytes in **step (1)**, **step (2)**, **step (3)** and **step (4)** of LT2S-MAKEP's authentication phase are 70, 54, 54 and 54 bytes, respectively, which results-in 232 transmitted bytes between the IoT device and the server. The communication overhead of **(author?)**'s protocol is 248 bytes. As a result,the communication overhead in our proposed mutual authentication scheme is lower than **(author?)**'s mechanism by 16 bytes and higher than T2S-MAKEP by 30 bytes.

6.3.4.3 Storage Constraints

LT2S-MAKEP imposes no constraints on the IoT device's storage for keeping sensitive data, achieving the same results as T2S-MAKEP and T2T-MAKEP. In addition, unlike the other two described methods, the LT2S-MAKEP server does not maintain FE information (Helper Data).

6.4 Conclusion

In this chapter, the implementation of the proposed protocols is given. First, we present the PyPuf simulator that is used to simulate PUF, and then we use it to implement a 16-APUF and generate a set of CRP. Then, we evaluated the performance of the generated data regarding five metrics: uniqueness, randomness, bit aliasing, uniformity, and reliability. Furthermore, we implement and use a fuzzy extractor mechanism and secure sketch to eliminate noise in the regenerated response. After that, we showed a given scenario where the proposed protocols could be applied and implemented. Finally, the performance evaluation of the proposed protocols is analyzed in terms of computational complexity, communication overhead, and storage constraints. In the next chapter, a general conclusion and future work will be presented.

CHAPTER 7

GENERAL CONCLUSION AND FUTURE WORK

Authentication is a crucial Internet of Things system and application security function. Using conventional authentication methods exposes the authentication task to various security threats and attacks. Using physical unclonable functions, the goal of this thesis is to build a secure, resilient, efficient, lightweight, and practical authentication system for IoT applications.

To secure authentication and communication, however, classic symmetric and asymmetric cryptographic methods require more computing power, larger memory, and higher energy sources. However, the limited memory capacity, processing power, and energy resources of IoT devices prevent the use of typical authentication procedures in IoT networks. Moreover, IoT devices can be discovered in public areas, and typical systems store secret keys in the device's volatile local memory, making them susceptible to physical attacks. In such a scenario, an adversary can easily gain physical access to the IoT device and recover the stored secret data or even clone the embedded circuitry. Utilizing the inherent disorder of physical things, Physical Unclonable Functions (PUFs) have recently been a hot topic in research and development to overcome these restrictions and problems. PUFs can produce unique secret information from the physical properties of an IoT device and use it as a unique device fingerprint, making them a very effective solution for the IoT authentication protocol.

Since it eliminates the need to maintain secret keys in the IoT memory, the PUF-based authentication mechanism is a very efficient solution for the IoT authentication protocol. On the basis of the randomness present in the physical properties of physical objects, PUFs extract unique secret information and utilize it as the IoT device identity, as a unique device fingerprint. The latter can therefore be utilized for IoT device authentication. It is also a low-cost security primitive, making it resistant to physical attacks and suited for IoT devices with limited resources.

There have been almost two decades of intensive research on PUFs since the concept was first introduced by Pappu *et al.* [52]. Physical unclonable functions have a completely different system than any other one-way function, especially with the challenge-response pair sets providing better reliability. Also, the level of defence is good with PUFs, and the variability of PUF systems allows users to choose the function with the

best characteristics according to the needs of the applications. Silicon PUF is one of the widely accepted hardware security primitives that finds application in authentication and secret key generation. It generates a secured key by the physical disorder nature of an electronic system. The physical structure of every electronic system is unique due to the inheriting differences during the manufacturing process using the same technology.

7.1 Thesis Contributions

To address this complex and challenging task, i.e., designing a robust PUF-based authentication protocol for IoT applications is not an easy task. This thesis consisted of two main parts: the literature review and the thesis contributions.

IoT and Physical Unclonable Functions (PUFs) were surveyed in the first part of the dissertation (chapters 2 and 3). Using this survey, we established the technical background for the IoT, including its characteristics, applications, architectures, and existing open issues related to IoT security. A survey of existing authentication techniques was also conducted. IoT PUF-based authentication and PUF's architecture are discussed in detail. To begin, we surveyed, discussed, and compared the state-of-the-art silicon PUF architectures and classified them into delay-based PUFs, memory-based PUFs, and analog electronic PUFs. We have also surveyed and classified existing Silicon PUF applications and use cases. Lastly, we examined IoT authentication protocols based on PUF over the past five years (since 2016).

The second part of this thesis (chapters 4 and 5) builds on the outcomes of the first part in that it addresses issues that are still unresolved in the state of the art, especially from the practical side of the existing PUF-based authentication protocol used with IoT systems, and proposes three contributions to advance the literature in this field.

A first contribution is T2S-MAKEP, for mutual authentication and key exchange between things and servers. By exploiting the randomness of a strong PUF, an IoT device with an Arbiter PUF could securely authenticate with its trusted server. The enrolment and authentication phases of this approach are separate. First, the device registers and stores credentials (CRP) on the server. In the second phase, both entities (IoT and server) authenticate and exchange messages. A fuzzy extractor solution is used to generate consistent secret keys.

Another contribution is T2T-MAKEP, a protocol for transferring keys and mutual authentication between two things. As a first step, it must authenticate with a trusted server using the T2S-MAKEP protocol. Using a fuzzy extractor solution, this scheme ensures error correction and noise elimination. Further, T2T-MAKEP does not require an update phase since no secret or non-secret information is stored on the local memory of the IoT end-device. Our schemes also generate a session key at the end of each successful authentication phase and exchange it securely between the communicating

entities.

LT2S-MAKEP is the third contribution, which allows lightweight mutual authentication and key exchange protocol between a thing and a server. Using the randomness of the integrated circuit of an IoT device, this is a low-cost protocol. It relies on two PUFs: the Arbiter PUF and the SRAM PUF. The first is used for encryption, whereas the second is used for authentication.

The proposed authentication protocols do not need to store any information on the local device memory, which avoids many attacks, especially physical-based ones. By relying on formal and informal security analysis, we proved that our developed protocol is secure against most existing attacks, especially physical ones.

7.2 Future Work

Additionally, we aim to advance the state-of-the-art by making recommendations for future research.

- Using the proposed schemes to achieve multi-hope authentication protocol.
- Deploying the proposed protocol with a blockchain architecture that exploits PUFs on the different mining nodes.
- Considering the energy consumption in our protocol by optimizing the exchanged messages while preserving the communication's reliability.
- Developing the proposed approach in order to support authorization management.
- Evaluating the proposed protocols in a real system setup.

REFERENCES

[1] R. Mahmoud, T. Yousuf, F. Aloul, and I. Zualkernan, "Internet of things (iot) security: Current status, challenges and prospective measures," in *2015 10th international conference for internet technology and secured transactions (ICITST)*, pp. 336–341, IEEE, 2015.

[2] A. Khanna and S. Kaur, "Internet of things (iot), applications and challenges: a comprehensive review," *Wireless Personal Communications*, vol. 114, no. 2, pp. 1687–1762, 2020.

[3] B. Halak, *Physically Unclonable Functions: From Basic Design Principles to Advanced Hardware Security Applications*. Springer, 2018.

[4] F. Zerrouki, S. Ouchani, and H. Bouarfa, "Towards a foundation of a mutual authentication protocol for a robust and resilient puf-based communication network," *Procedia Computer Science*, vol. 191, pp. 215–222, 2021.

[5] F. Zerrouki, S. Ouchani, and H. Bouarfa, "A survey on silicon pufs," *Journal of Systems Architecture*, vol. 127, p. 102514, 2022.

[6] J. W. Lee, D. Lim, B. Gassend, G. E. Suh, M. Van Dijk, and S. Devadas, "A technique to build a secret key in integrated circuits for identification and authentication applications," in *2004 Symposium on VLSI Circuits. Digest of Technical Papers (IEEE Cat. No. 04CH37525)*, pp. 176–179, IEEE, 2004.

[7] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Lightweight secure pufs," in *2008 IEEE/ACM International Conference on Computer-Aided Design*, pp. 670–673, IEEE, 2008.

[8] M. Rostami, M. Majzoobi, F. Koushanfar, D. S. Wallach, and S. Devadas, "Robust and reverse-engineering resilient puf authentication and key-exchange by substring matching," *IEEE Transactions on Emerging Topics in Computing*, vol. 2, no. 1, pp. 37–49, 2014.

[9] S. S. Avvaru, Z. Zeng, and K. K. Parhi, "Homogeneous and heterogeneous feed-forward xor physical unclonable functions," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 2485–2498, 2020.

[10] M. Khalafalla and C. Gebotys, "Pufs deep attacks: Enhanced modeling attacks using deep learning techniques to break the security of double arbiter pufs," in *2019 Design, Automation and Test in Europe Conference and Exhibition (DATE)*, pp. 204–209, IEEE, 2019.

[11] D. P. Sahoo, D. Mukhopadhyay, R. S. Chakraborty, and P. H. Nguyen, "A multiplexer-based arbiter puf composition with enhanced reliability and security," *IEEE Transactions on Computers*, vol. 67, no. 3, pp. 403–417, 2017.

[12] Q. Ma, C. Gu, N. Hanley, C. Wang, W. Liu, and M. O'Neill, "A machine learning attack resistant multi-puf design on fpga," in *2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 97–104, IEEE, 2018.

[13] Z. Huang, L. Li, Y. Chen, Z. Li, Q. Wang, and X. Jiang, "Rppuf: An ultra-lightweight reconfigurable pico-physically unclonable function for resource-constrained iot devices," *Electronics*, vol. 10, no. 23, p. 3039, 2021.

[14] P. H. Nguyen, D. P. Sahoo, C. Jin, K. Mahmood, U. Rührmair, and M. van Dijk, "The interpose puf: Secure puf design against state-of-the-art machine learning attacks," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 243–290, 2019.

[15] Z. Li, L. Zhu, M. Huang, Z. Chen, S. Chen, and B. Li, "Racing apuf: A novel apuf against machine learning attack with high reliability," in *2019 IEEE 4th International Conference on Signal and Image Processing (ICSIP)*, pp. 722–726, IEEE, 2019.

[16] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *2007 44th ACM/IEEE Design Automation Conference*, pp. 9–14, IEEE, 2007.

[17] A. Maiti and P. Schaumont, "Improved ring oscillator puf: An fpga-friendly secure primitive," *Journal of cryptology*, vol. 24, no. 2, pp. 375–397, 2011.

[18] M. Gao, K. Lai, and G. Qu, "A highly flexible ring oscillator puf," in *Proceedings of the 51st Annual Design Automation Conference*, pp. 1–6, 2014.

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

[20] C. Marchand, L. Bossuet, U. Mureddu, N. Bochard, A. Cherkaoui, and V. Fischer, "Implementation and characterization of a physical unclonable function for

iot: a case study with the tero-puf," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 1, pp. 97–109, 2017.

[21] R. Della Sala, D. Bellizia, and G. Scotti, "A novel ultra-compact fpga puf: The dd-puf," *Cryptography*, vol. 5, no. 3, p. 23, 2021.

[22] D. Suzuki and K. Shimizu, "The glitch puf: A new delay-puf architecture exploiting glitch shapes," in *International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 366–382, Springer, 2010.

[23] K. Shimizu, D. Suzuki, and T. Kasuya, "Glitch puf: extracting information from usually unwanted glitches," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 95, no. 1, pp. 223–233, 2012.

[24] Y. Yao, M. Kim, J. Li, I. L. Markov, and F. Koushanfar, "Clockpuf: Physical unclonable functions based on clock networks," in *2013 Design, Automation and Test in Europe Conference and Exhibition (DATE)*, pp. 422–427, IEEE, 2013.

[25] S. S. Kumar, J. Guajardo, R. Maes, G.-J. Schrijen, and P. Tuyls, "The butterfly puf protecting ip on every fpga," in *2008 IEEE International Workshop on Hardware-Oriented Security and Trust*, pp. 67–70, IEEE, 2008.

[26] A. Ardakani, S. B. Shokouhi, and A. Reyhani-Masoleh, "Improving performance of fpga-based sr-latch puf using transient effect ring oscillator and programmable delay lines," *Integration*, vol. 62, pp. 371–381, 2018.

[27] P. Simons, E. van der Sluis, and V. van der Leest, "Buskeeper pufs, a promising alternative to d flip-flop pufs," in *2012 IEEE International Symposium on Hardware-Oriented Security and Trust*, pp. 7–12, IEEE, 2012.

[28] Q. Chen, G. Csaba, P. Lugli, U. Schlichtmann, and U. Rührmair, "The bistable ring puf: A new architecture for strong physical unclonable functions," in *2011 IEEE International Symposium on Hardware-Oriented Security and Trust*, pp. 134–141, IEEE, 2011.

[29] K. Lofstrom, W. R. Daasch, and D. Taylor, "Ic identification circuit using device mismatch," in *2000 IEEE International Solid-State Circuits Conference. Digest of Technical Papers (Cat. No. 00CH37056)*, pp. 372–373, IEEE, 2000.

[30] P. Tuyls, G.-J. Schrijen, B. Škorić, J. Van Geloven, N. Verhaegh, and R. Wolters, "Read-proof hardware from protective coatings," in *International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 369–383, Springer, 2006.

[31] P. Sudhanya and P. M. Krishnammal, "Study of different silicon physical unclonable functions," in *2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, pp. 81–85, IEEE, 2016.

[32] F. Zerrouki, S. Ouchani, and H. Bouarfa, "Puf-based mutual authentication and session key establishment protocol for iot devices," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–19, 2022.

[33] T. A. Idriss, H. A. Idriss, and M. A. Bayoumi, "A lightweight puf-based authentication protocol using secret pattern recognition for constrained iot devices," *IEEE Access*, vol. 9, pp. 80546–80558, 2021.

[34] M. Kaveh, S. Aghapour, D. Martin, and M. R. Mosavi, "A secure lightweight signcryption scheme for smart grid communications using reliable physically unclonable function," in *2020 IEEE International Conference on Environment and Electrical Engineering and 2020 IEEE Industrial and Commercial Power Systems Europe (EEEIC/IandCPS Europe)*, pp. 1–6, IEEE, 2020.

[35] V. P. Yanambaka, S. P. Mohanty, E. Kougianos, and D. Puthal, "Pmsec: Physical unclonable function-based robust and lightweight authentication in the internet of medical things," *IEEE Transactions on Consumer Electronics*, vol. 65, no. 3, pp. 388–397, 2019.

[36] Y. Zheng, W. Liu, C. Gu, and C.-H. Chang, "Puf-based mutual authentication and key exchange protocol for peer-to-peer iot applications," *IEEE Transactions on Dependable and Secure Computing*, 2022.

[37] F. Najafi, M. Kaveh, D. Martín, and M. Reza Mosavi, "Deep puf: A highly reliable dram puf-based authentication for iot networks using deep convolutional neural networks," *Sensors*, vol. 21, no. 6, p. 2009, 2021.

[38] Z. Guan, H. Liu, and Y. Qin, "Physical unclonable functions for iot device authentication," *Journal of Communications and Information Networks*, vol. 4, no. 4, pp. 44–54, 2019.

[39] C. Pu and Y. Li, "Lightweight authentication protocol for unmanned aerial vehicles using physical unclonable function and chaotic system," in *2020 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN*, pp. 1–6, IEEE, 2020.

[40] J. Meng, X. Zhang, T. Cao, and Y. Xie, "Lightweight and anonymous mutual authentication protocol for iot devices with physical unclonable functions," 2021.

[41] M. N. Aman, S. A. Chaudhry, and F. Al-Turjman, "Rapidauth: Fast authentication for sustainable iot," in *International Conference on Forthcoming Networks and Sustainability in the IoT Era*, pp. 82–95, Springer, 2020.

[42] B. Kim, S. Yoon, Y. Kang, and D. Choi, "Puf based iot device authentication scheme," in *2019 International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 1460–1462, IEEE, 2019.

[43] M. A. Muhal, X. Luo, Z. Mahmood, and A. Ullah, "Physical unclonable function based authentication scheme for smart devices in internet of things," in *2018 IEEE International Conference on Smart Internet of Things (SmartIoT)*, pp. 160–165, IEEE, 2018.

[44] A. Mostafa, S. J. Lee, and Y. K. Peker, "Physical unclonable function and hashing are all you need to mutually authenticate iot devices," *Sensors*, vol. 20, no. 16, p. 4361, 2020.

[45] N. Kobeissi, G. Nicolas, and M. Tiwari, "Verifpal: cryptographic protocol analysis for the real world," in *International Conference on Cryptology in India*, pp. 151–202, Springer, 2020.

[46] M. Saadeh, A. Sleit, K. E. Sabri, and W. Almobaideen, "Object authentication in the context of the internet of things: A survey," *Journal of Cyber Security and Mobility*, pp. 385–448, 2020.

[47] S. Khanam, I. B. Ahmedy, M. Y. I. Idris, M. H. Jaward, and A. Q. B. M. Sabri, "A survey of security challenges, attacks taxonomy and advanced countermeasures in the internet of things," *IEEE access*, vol. 8, pp. 219709–219743, 2020.

[48] Q. F. Hassan, *Internet of things A to Z: technologies and applications*. John Wiley and Sons, 2018.

[49] K. K. Patel, S. M. Patel, *et al.*, "Internet of things-iot: definition, characteristics, architecture, enabling technologies, application and future challenges," *International journal of engineering science and computing*, vol. 6, no. 5, 2016.

[50] T. Nandy, M. Y. I. B. Idris, R. M. Noor, L. M. Kiah, L. S. Lun, N. B. A. Juma'at, I. Ahmedy, N. A. Ghani, and S. Bhattacharyya, "Review on security of internet of things authentication mechanism," *IEEE Access*, vol. 7, pp. 151054–151089, 2019.

[51] F. Zerrouki, S. Ouchani, and H. Bouarfa, "A low-cost authentication protocol using arbiter-puf," in *International Conference on Model and Data Engineering*, pp. 101–116, Springer, 2021.

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

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

[54] S. Madakam, V. Lake, V. Lake, V. Lake, *et al.*, "Internet of things (iot): A literature review," *Journal of Computer and Communications*, vol. 3, no. 05, p. 164, 2015.

[55] F. H. Al-Naji and R. Zagrouba, "A survey on continuous authentication methods in internet of things environment," *Computer Communications*, vol. 163, pp. 109–133, 2020.

[56] N. M. Kumar and P. K. Mallick, "The internet of things: Insights into the building blocks, component interactions, and architecture layers," *Procedia computer science*, vol. 132, pp. 109–117, 2018.

[57] P. Goyal, A. K. Sahoo, T. K. Sharma, and P. K. Singh, "Internet of things: Applications, security and privacy: A survey," *Materials Today: Proceedings*, vol. 34, pp. 752–759, 2021.

[58] M. M. Ogonji, G. Okeyo, and J. M. Wafula, "A survey on privacy and security of internet of things," *Computer Science Review*, vol. 38, p. 100312, 2020.

[59] Y. Gao, D. C. Ranasinghe, S. F. Al-Sarawi, O. Kavehei, and D. Abbott, "Emerging physical unclonable functions with nanotechnology," *IEEE access*, vol. 4, pp. 61–80, 2016.

[60] I. Verbauwhede and R. Maes, "Physically unclonable functions: manufacturing variability as an unclonable device identifier," in *Proceedings of the 21st edition of the great lakes symposium on Great lakes symposium on VLSI*, pp. 455–460, 2011.

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

[62] R. Maes and I. Verbauwhede, "Physically unclonable functions: A study on the state of the art and future research directions," in *Towards Hardware-Intrinsic Security*, pp. 3–37, Springer, 2010.

[63] J. H. M. Moreno, "Memristive modeling for hardware security applications," Master's thesis, National Institute of Astrophysics, Optics and Electronics, Research institution in San Andres Cholula, Puebla, Mexico, 2019.

[64] R. Maes and I. Verbauwhede, "A discussion on the properties of physically unclonable functions," in *TRUST 2010 Workshop, Berlin*, 2010.

[65] I. Papakonstantinou and N. Sklavos, "Physical unclonable functions (pufs) design technologies: Advantages and trade offs," in *Computer and Network Security Essentials*, pp. 427–442, Springer, 2018.

[66] E. I. Vatajelu, G. Di Natale, M. S. Mispan, and B. Halak, "On the encryption of the challenge in physically unclonable functions," in *2019 IEEE 25th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, pp. 115–120, IEEE, 2019.

[67] M. El-Hajj, A. Fadlallah, M. Chamoun, and A. Serhrouchni, "Secure puf: Physically unclonable function based on arbiter with enhanced resistance against machine learning (ml) attacks," in *SEIA'2019 Conference Proceedings*, p. 216, Lulu. com.

[68] S. Kumar *et al.*, *Analysis of Machine Learning Modeling Attacks on Ring Oscillator based Hardware Security*. PhD thesis, University of Toledo, 2018.

[69] A. Maiti, *A systematic approach to design an efficient physical unclonable function*. PhD thesis, Virginia Polytechnic Institute and State University, 2012.

[70] Y. Hori, T. Yoshida, T. Katashita, and A. Satoh, "Quantitative and statistical performance evaluation of arbiter physical unclonable functions on fpgas," in *2010 International Conference on Reconfigurable Computing and FPGAs*, pp. 298–303, IEEE, 2010.

[71] Y. Ravishankar, "PUFs – An extensive survey," masters thesis, ECE Department, George Mason University, Fairfax, Virginia, USA, Jul 2015.

[72] A. Kumar, R. S. Mishra, and K. Kashwan, "Puf based challenge response pair for secured authentication," *International Journal of Control Theory and Applications*, vol. 9, pp. 115–121, 2016.

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

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

[75] S. S. Avvaru and K. K. Parhi, "Feed-forward xor pufs: Reliability and attack-resistance analysis," in *Proceedings of the 2019 on Great Lakes Symposium on VLSI*, pp. 287–290, 2019.

[76] B. Gassend, D. Clarke, M. Van Dijk, and S. Devadas, "Silicon physical random functions," in *Proceedings of the 9th ACM conference on Computer and communications security*, pp. 148–160, 2002.

[77] U. Ruhrmair and J. Solter, "Puf modeling attacks: An introduction and overview," in *2014 Design, Automation and Test in Europe Conference and Exhibition (DATE)*, 2014.

[78] F. Ganji, *On the learnability of physically unclonable functions*. Springer, 2018.

[79] F. Ganji, D. Forte, and J.-P. Seifert, "Pufmeter a property testing tool for assessing the robustness of physically unclonable functions to machine learning attacks," *IEEE Access*, vol. 7, pp. 122513–122521, 2019.

[80] D. Chatterjee, D. Mukhopadhyay, and A. Hazra, "Puf-g: A cad framework for automated assessment of provable learnability from formal puf representations," in *Proceedings of the 39th International Conference on Computer-Aided Design*, pp. 1–9, 2020.

[81] U. Rührmair, J. Sölter, F. Sehnke, X. Xu, A. Mahmoud, V. Stoyanova, G. Dror, J. Schmidhuber, W. Burleson, and S. Devadas, "Puf modeling attacks on simulated and silicon data," *IEEE transactions on information forensics and security*, vol. 8, no. 11, pp. 1876–1891, 2013.

[82] F. Farha, H. Ning, K. Ali, L. Chen, and C. Nugent, "Sram-puf-based entities authentication scheme for resource-constrained iot devices," *IEEE Internet of Things Journal*, vol. 8, no. 7, pp. 5904–5913, 2020.

[83] E. Dubrova, O. Näslund, B. Degen, A. Gawell, and Y. Yu, "Crc-puf: A machine learning attack resistant lightweight puf construction," in *2019 IEEE European symposium on security and privacy workshops (EuroSandPW)*, pp. 264–271, IEEE, 2019.

[84] M. S. Mispan, H. Su, M. Zwolinski, and B. Halak, "Cost-efficient design for modeling attacks resistant pufs," in *2018 Design, Automation and Test in Europe Conference and Exhibition (DATE)*, pp. 467–472, IEEE, 2018.

[85] T. Kroeger, W. Cheng, S. Guilley, J.-L. Danger, and N. Karimi, "Making obfuscated pufs secure against power side-channel based modeling attacks," in

2021 Design, Automation and Test in Europe Conference and Exhibition (DATE), pp. 1000–1005, IEEE, 2021.

[86] X. Cai, G. Xie, S. Kuang, R. Li, and S. Li, "Efficient dpa side channel countermeasure with mim capacitors-based current equalizer," *Journal of Systems Architecture*, vol. 118, p. 102146, 2021.

[87] E. Ozturk, G. Hammouri, and B. Sunar, "Physical unclonable function with tristate buffers," in *2008 IEEE International Symposium on Circuits and Systems*, pp. 3194–3197, IEEE, 2008.

[88] D. Karakoyunlu and B. Sunar, "Differential template attacks on puf enabled cryptographic devices," in *2010 IEEE International Workshop on Information Forensics and Security*, pp. 1–6, IEEE, 2010.

[89] D. Merli, J. Heyszl, B. Heinz, D. Schuster, F. Stumpf, and G. Sigl, "Localized electromagnetic analysis of ro pufs," in *2013 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pp. 19–24, IEEE, 2013.

[90] L. Tebelmann, J.-L. Danger, and M. Pehl, "Self-secured puf: protecting the loop puf by masking," in *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pp. 293–314, Springer, 2020.

[91] U. Rührmair, X. Xu, J. Sölter, A. Mahmoud, M. Majzoobi, F. Koushanfar, and W. Burleson, "Efficient power and timing side channels for physical unclonable functions," in *International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 476–492, Springer, 2014.

[92] M. Ling, L. Wu, X. Li, X. Zhang, J. Hou, and Y. Wang, "Design of monitor and protect circuits against fib attack on chip security," in *2012 Eighth International Conference on Computational Intelligence and Security*, pp. 530–533, IEEE, 2012.

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

[94] I. A. B. Adames, J. Das, and S. Bhanja, "Survey of emerging technology based physical unclonable funtions," in *2016 International Great Lakes Symposium on VLSI (GLSVLSI)*, pp. 317–322, IEEE, 2016.

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

[96] T. Machida, D. Yamamoto, M. Iwamoto, and K. Sakiyama, "A new mode of operation for arbiter puf to improve uniqueness on fpga," in *2014 Federated Conference on Computer Science and Information Systems*, pp. 871–878, IEEE, 2014.

[97] D. P. Sahoo, S. Saha, D. Mukhopadhyay, R. S. Chakraborty, and H. Kapoor, "Composite puf: A new design paradigm for physically unclonable functions on fpga," in *2014 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pp. 50–55, IEEE, 2014.

[98] D. P. Sahoo, P. H. Nguyen, D. Mukhopadhyay, and R. S. Chakraborty, "A case of lightweight puf constructions: Cryptanalysis and machine learning attacks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 8, pp. 1334–1343, 2015.

[99] J.-Q. Huang, M. Zhu, B. Liu, and W. Ge, "Deep learning modeling attack analysis for multiple fpga-based apuf protection structures," in *2018 14th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT)*, pp. 1–3, IEEE.

[100] T. Machida, D. Yamamoto, M. Iwamoto, and K. Sakiyama, "A new arbiter puf for enhancing unpredictability on fpga," *The Scientific World Journal*, vol. 2015, 2015.

[101] J. Shi, Y. Lu, and J. Zhang, "Approximation attacks on strong pufs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2019.

[102] C. Gu, N. Hanley, and M. O'neill, "Improved reliability of fpga-based puf identification generator design," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 10, no. 3, pp. 1–23, 2017.

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

[104] D. Merli, D. Schuster, F. Stumpf, and G. Sigl, "Semi-invasive em attack on fpga ro pufs and countermeasures," in *Proceedings of the Workshop on Embedded Systems Security*, pp. 1–9, 2011.

[105] J. Miskelly, C. Gu, Q. Ma, Y. Cui, W. Liu, and M. O'Neill, "Modelling attack analysis of configurable ring oscillator (cro) puf designs," in *2018 IEEE 23rd International Conference on Digital Signal Processing (DSP)*, pp. 1–5, IEEE, 2018.

[106] M.-D. M. Yu and S. Devadas, "Recombination of physical unclonable functions," *35th Annual GOMACTech Conference*, pp. 1–4, 2010.

[107] E. Strieder, C. Frisch, and M. Pehl, "Machine learning of physical unclonable functions using helper data: Revealing a pitfall in the fuzzy commitment scheme," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 1–36, 2021.

[108] L. Tebelmann, M. Pehl, and V. Immler, "Side-channel analysis of the tero puf," in *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pp. 43–60, Springer, 2019.

[109] R. Nithyanand, R. Sion, and J. Solis, "Poster: Making the case for intrinsic personal physical unclonable functions (ip-pufs)," in *Proceedings of the 18th ACM conference on Computer and communications security*, pp. 825–828, 2011.

[110] W. Yu, Y. Wen, S. Köse, and J. Chen, "Exploiting multi-phase on-chip voltage regulators as strong puf primitives for securing iot," *Journal of Electronic Testing*, vol. 34, no. 5, pp. 587–598, 2018.

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

[112] C. Helfmeier, C. Boit, D. Nedospasov, and J.-P. Seifert, "Cloning physically unclonable functions," in *2013 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pp. 1–6, IEEE, 2013.

[113] R. Maes, P. Tuyls, and I. Verbauwhede, "Intrinsic pufs from flip-flops on reconfigurable devices," in *3rd Benelux workshop on information and system security (WISSec 2008)*, vol. 17, p. 2008, 2008.

[114] Y. Su, J. Holleman, and B. P. Otis, "A digital 1.6 pj/bit chip identification circuit using process variations," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 1, pp. 69–77, 2008.

[115] D. Schuster and R. Hesselbarth, "Evaluation of bistable ring pufs using single layer neural networks," in *International Conference on Trust and Trustworthy Computing*, pp. 101–109, Springer, 2014.

[116] D. Chatterjee, D. Mukhopadhyay, and A. Hazra, "Interpose puf can be pac learned.," *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 471, 2020.

[117] R. Helinski, D. Acharyya, and J. Plusquellic, "A physical unclonable function defined using power distribution system equivalent resistance variations," in *2009 46th ACM/IEEE Design Automation Conference*, pp. 676–681, IEEE, 2009.

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

[119] M. N. Aman, K. C. Chua, and B. Sikdar, "Mutual authentication in iot systems using physical unclonable functions," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1327–1340, 2017.

[120] T. Alladi, G. Bansal, V. Chamola, M. Guizani, *et al.*, "Secauthuav: A novel authentication scheme for uav-ground station and uav-uav communication," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 15068–15077, 2020.

[121] G. Bansal and B. Sikdar, "S-maps: Scalable mutual authentication protocol for dynamic uav swarms," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 11, pp. 12088–12100, 2021.

[122] W. Wang, Q. Chen, Z. Yin, G. Srivastava, T. R. Gadekallu, F. Alsolami, and C. Su, "Blockchain and puf-based lightweight authentication protocol for wireless medical sensor networks," *IEEE Internet of Things Journal*, vol. 9, no. 11, pp. 8883–8891, 2021.

[123] T.-F. Lee and W.-Y. Chen, "Lightweight fog computing-based authentication protocols using physically unclonable functions for internet of medical things," *Journal of Information Security and Applications*, vol. 59, p. 102817, 2021.

[124] P. Gope, O. Millwood, and B. Sikdar, "A scalable protocol level approach to prevent machine learning attacks on physically unclonable function based authentication mechanisms for internet of medical things," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 3, pp. 1971–1980, 2021.

[125] D. Kwon, Y. Park, and Y. Park, "Provably secure three-factor-based mutual authentication scheme with puf for wireless medical sensor networks," *Sensors*, vol. 21, no. 18, p. 6039, 2021.

[126] Q. Jiang, X. Zhang, N. Zhang, Y. Tian, X. Ma, and J. Ma, "Two-factor authentication protocol using physical unclonable function for iov," in *2019 IEEE/CIC International Conference on Communications in China (ICCC)*, pp. 195–200, IEEE, 2019.

[127] Q. Jiang, X. Zhang, N. Zhang, Y. Tian, X. Ma, and J. Ma, "Three-factor authentication protocol using physical unclonable function for iov," *Computer Communications*, vol. 173, pp. 45–55, 2021.

[128] K. Mershad, O. Cheikhrouhou, and L. Ismail, "Proof of accumulated trust: A new consensus protocol for the security of the iov," *Vehicular Communications*, vol. 32, p. 100392, 2021.

[129] Y.-N. Cao, Y. Wang, Y. Ding, H. Zheng, Z. Guan, and H. Wang, "A puf-based lightweight authenticated metering data collection scheme with privacy protection in smart grid," in *2021 IEEE Intl Conf on Parallel and Distributed Processing with Applications, Big Data and Cloud Computing, Sustainable Computing and Communications, Social Computing and Networking (ISPA/BDCloud/-SocialCom/SustainCom)*, pp. 876–883, IEEE, 2021.

[130] H. M. S. Badar, S. Qadri, S. Shamshad, M. F. Ayub, K. Mahmood, and N. Kumar, "An identity based authentication protocol for smart grid environment using physical uncloneable function," *IEEE Transactions on Smart Grid*, vol. 12, no. 5, pp. 4426–4434, 2021.

[131] S. Puchinger, S. Müelich, M. Bossert, M. Hiller, and G. Sigl, "On error correction for physical unclonable functions," in *SCC 2015; 10th International ITG Conference on Systems, Communications and Coding*, pp. 1–6, VDE, 2015.

[132] A. Shamsoshoara, A. Korenda, F. Afghah, and S. Zeadally, "A survey on hardware-based security mechanisms for internet of things," *arXiv preprint arXiv:1907.12525*, 2019.

[133] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," *SIAM journal on computing*, vol. 38, no. 1, pp. 97–139, 2008.

[134] F. Zerrouki, S. Ouchani, and H. Bouarfa, "A generation and recovery framework for silicon pufs based cryptographic key," in *International Conference on Model and Data Engineering*, pp. 121–137, Springer, 2021.

[135] J. X. Zheng and M. Potkonjak, "A digital puf-based ip protection architecture for network embedded systems," in *2014 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, pp. 255–256, IEEE, 2014.

[136] J. Zhang, Y. Lin, Y. Lyu, and G. Qu, "A puf-fsm binding scheme for fpga ip protection and pay-per-device licensing," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 6, pp. 1137–1150, 2015.

[137] Q. Guo, J. Ye, Y. Gong, Y. Hu, and X. Li, "Puf based pay-per-device scheme for ip protection of cnn model," in *2018 IEEE 27th Asian Test Symposium (ATS)*, pp. 115–120, IEEE, 2018.

[138] S. Kalanadhabhatta, D. Kumar, K. K. Anumandla, S. A. Reddy, and A. Acharyya, "Puf-based secure chaotic random number generator design methodology," *IEEE transactions on very large scale integration (VLSI) systems*, vol. 28, no. 7, pp. 1740–1744, 2020.

[139] T. Kaya, "A true random number generator based on a chua and ro-puf: design, implementation and statistical analysis," *Analog Integrated Circuits and Signal Processing*, vol. 102, no. 2, pp. 415–426, 2020.

[140] J. Calhoun, C. Minwalla, C. Helmich, F. Saqib, W. Che, and J. Plusquellic, "Physical unclonable function (puf)-based e-cash transaction protocol (puf-cash)," *Cryptography*, vol. 3, no. 3, p. 18, 2019.

[141] B. Yang, K. Yang, Z. Zhang, Y. Qin, and D. Feng, "Aep-m: Practical anonymous e-payment for mobile devices using arm trustzone and divisible e-cash," in *International Conference on Information Security*, pp. 130–146, Springer, 2016.

[142] L. B. Kish, K. Entesari, C.-G. Granqvist, and C. Kwan, "Unconditionally secure credit/debit card chip scheme and physical unclonable function," *Fluctuation and Noise Letters*, vol. 16, no. 01, p. 1750002, 2017.

[143] G. E. Suh, C. W. O'Donnell, and S. Devadas, "Aegis: A single-chip secure processor," *IEEE Design and Test of Computers*, vol. 24, no. 6, pp. 570–580, 2007.

[144] W. Xiong, A. Schaller, S. Katzenbeisser, and J. Szefer, "Software protection using dynamic pufs," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 2053–2068, 2019.

[145] V. Suresh and R. Manimegalai, "Spic-sram puf intergrated chip based software licensing model," in *International Symposium on Security in Computing and Communication*, pp. 377–388, Springer, 2018.

[146] F. Kohnhäuser, A. Schaller, and S. Katzenbeisser, "Puf-based software protection for low-end embedded devices," in *International Conference on Trust and Trustworthy Computing*, pp. 3–21, Springer, 2015.

[147] K. Mahmood, S. Shamshad, M. Rana, A. Shafiq, S. Ahmad, M. A. Akram, and R. Amin, "Puf enable lightweight key-exchange and mutual authentication protocol for multi-server based d2d communication," *Journal of Information Security and Applications*, vol. 61, p. 102900, 2021.

[148] J. Aarestad, P. Ortiz, D. Acharyya, and J. Plusquellic, "Help: A hardware-embedded delay puf," *IEEE Design and Test*, vol. 30, no. 2, pp. 17–25, 2013.

[149] J. Zhang, S. Rajendran, Z. Sun, R. Woods, and L. Hanzo, "Physical layer security for the internet of things: Authentication and key generation," *IEEE Wireless Communications*, vol. 26, no. 5, pp. 92–98, 2019.

[150] K. Rajaguru and R. Hansdah, "Symmetric key-based lightweight authentication protocols for rfid security," in *2018 32nd International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, pp. 488–495, IEEE, 2018.

[151] A. Kumari, V. Kumar, M. YahyaAbbasi, and M. Alam, "The cryptanalysis of a secure authentication scheme based on elliptic curve cryptography for iot and cloud servers," in *2018 international conference on advances in computing, Communication Control and Networking (ICACCCN)*, pp. 321–325, IEEE, 2018.

[152] A. Tewari and B. Gupta, "A mutual authentication protocol for iot devices using elliptic curve cryptography," in *2018 8th International Conference on Cloud Computing, Data Science and Engineering (Confluence)*, pp. 716–720, IEEE, 2018.

[153] P. Figueiredo e Silva, V. Kaseva, and E. S. Lohan, "Wireless positioning in iot: A look at current and future trends," *Sensors*, vol. 18, no. 8, p. 2470, 2018.

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

[155] M. Kaveh, D. Martín, and M. R. Mosavi, "A lightweight authentication scheme for v2g communications: A puf-based approach ensuring cyber/physical security and identity/location privacy," *Electronics*, vol. 9, no. 9, p. 1479, 2020.

[156] B. Blanchet, B. Smyth, V. Cheval, and M. Sylvestre, "Proverif 2.00: automatic cryptographic protocol verifier, user manual and tutorial," *Version from*, pp. 05–16, 2018.

[157] S. Li, T. Zhang, B. Yu, and K. He, "A provably secure and practical puf-based end-to-end mutual authentication and key exchange protocol for iot," *IEEE Sensors Journal*, vol. 21, no. 4, pp. 5487–5501, 2020.

[158] Y. Hu, L. Wu, Z. Chen, Y. Huang, X. Xu, K. Li, and J. Zhang, "Stt-mram-based reliable weak puf," *IEEE Transactions on Computers*, 2021.

[159] E. I. Vatajelu, G. Di Natale, and P. Prinetto, "Zero bit-error-rate weak puf based on spin-transfer-torque mram memories," in *2017 IEEE 2nd International Verification and Security Workshop (IVSW)*, pp. 128–133, IEEE, 2017.

[160] H. I. Ahmed, A. A. Nasr, S. Abdel-Mageid, and H. K. Aslan, "A survey of iot security threats and defenses," *International Journal of Advanced Computer Research*, vol. 9, no. 45, pp. 325–350, 2019.

[161] D. Dolev and A. Yao, "On the security of public key protocols," *IEEE Transactions on information theory*, vol. 29, no. 2, pp. 198–208, 1983.

[162] M. Barbosa, G. Barthe, K. Bhargavan, B. Blanchet, C. Cremers, K. Liao, and B. Parno, "Sok: Computer-aided cryptography," in *2021 IEEE Symposium on Security and Privacy (SP)*, pp. 777–795, IEEE, 2021.

[163] S. F. Doghmi, J. D. Guttman, and F. J. Thayer, "Searching for shapes in cryptographic protocols," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pp. 523–537, Springer, 2007.

[164] V. Cheval, S. Kremer, and I. Rakotonirina, "Deepsec: deciding equivalence properties in security protocols theory and practice," in *2018 IEEE Symposium on Security and Privacy (SP)*, pp. 529–546, IEEE, 2018.

[165] J. Bengtson, K. Bhargavan, C. Fournet, A. D. Gordon, and S. Maffeis, "Refinement types for secure implementations," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 33, no. 2, pp. 1–45, 2011.

[166] S. Escobar, C. Meadows, and J. Meseguer, "Maude-npa: Cryptographic protocol analysis modulo equational properties," in *Foundations of Security Analysis and Design V*, pp. 1–50, Springer, 2009.

[167] V. Cheval and B. Blanchet, "Proving more observational equivalences with proverif," in *International Conference on Principles of Security and Trust*, pp. 226–246, Springer, 2013.

[168] C. J. Cremers, "The scyther tool: Verification, falsification, and analysis of security protocols," in *International conference on computer aided verification*, pp. 414–418, Springer, 2008.

[169] B. Schmidt, S. Meier, C. Cremers, and D. Basin, "Automated analysis of diffie-hellman protocols and advanced security properties," in *2012 IEEE 25th Computer Security Foundations Symposium*, pp. 78–94, IEEE, 2012.

[170] Y.-W. Hu, T.-P. Zhang, C.-F. Wang, K.-K. Liu, Y. Sun, L. Li, C.-F. Lv, Y.-C. Liang, F.-H. Jiao, W.-B. Zhao, *et al.*, "Flexible and biocompatible physical unclonable function anti-counterfeiting label," *Advanced Functional Materials*, p. 2102108, 2021.

[171] M. H. Ameri, M. Delavar, and J. Mohajeri, "Provably secure and efficient puf-based broadcast authentication schemes for smart grid applications," *International Journal of Communication Systems*, vol. 32, no. 8, p. e3935, 2019.

[172] P. Gope and B. Sikdar, "Privacy-aware authenticated key agreement scheme for secure smart grid communication," *IEEE Transactions on Smart Grid*, vol. 10, no. 4, pp. 3953–3962, 2018.

[173] J. W. Byun, "End-to-end authenticated key exchange based on different physical unclonable functions," *IEEE Access*, vol. 7, pp. 102951–102965, 2019.

[174] N. Wisiol, C. Gräbnitz, C. Mühl, B. Zengin, T. Soroceanu, N. Pirnay, K. Mursi, and A. Baliuka, "pypuf: Cryptanalysis of physically unclonable functions," 2021.

[175] R. O'Donnell, *Analysis of boolean functions*. Cambridge University Press, 2014.

[176] M. Hiller, L. Kürzinger, and G. Sigl, "Review of error correction for pufs and evaluation on state-of-the-art fpgas," *Journal of Cryptographic Engineering*, vol. 10, no. 3, pp. 229–247, 2020.

[177] P. Gope and B. Sikdar, "Lightweight and privacy-preserving two-factor authentication scheme for iot devices," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 580–589, 2018.

[178] H. Wang, J. Meng, X. Du, T. Cao, and Y. Xie, "Lightweight and anonymous mutual authentication protocol for edge iot nodes with physical unclonable function," *Security and Communication Networks*, vol. 2022, pp. 1–11, 2022.

Appendices

Appendix A

PUF-Based regeneration response

This chapter displays all the background and the operations that need to be executed in the PUF-based response regeneration process using fuzzy extraction and Linear Block Codes.

A.1 Fuzzy extraction

A.1.1 Statistical Distance.

Also known as variation distance, it is the difference between the distributions of two random variables X and Y from the same set given by: $SD(X,Y) \stackrel{\Delta}{=} \frac{1}{2}\sum_v |Pr(X = v) - Pr(Y = v)|$. Here, the probability of a random variable X is written as $Pr(X)$.

A.1.2 Min-entropy.

The entropy specifies the amount of information contained in data. But when discussing security, one is often interested in the probability that the adversary predicts a random value. Thus, the best strategy is to guess the most probable value. However, the predictability of a random variable *X* is $max_x Pr[X = x]$, and correspondingly, the min-entropy $H_\infty(X)$ of a random variable *X* is: $H_\infty(X) = -log(max_x Pr[X = x])$

For two random variables *X* and *Y*, the average min-entropy of *X* given *Y* is: $H_\infty(X|Y) = -log(E_{y\leftarrow Y}[max_x Pr[X = x|Y = y]])\bar{-}log(E_{y\leftarrow Y}[2^{H_\infty(X|Y=y)}])$

A.1.3 Secure Sketching.

Secure Sketch (SS) [133] is a primitive performing error correction on a noisy data *w'* by eliminating noises and reconstructing the original one *w*, clean from noise. SS takes *w* as input and produces a public sketch or a public helper data *s*, without revealing enough information about *w*. Rec uses *s* to recover *w* from *w'* by cleaning noise while *w'*≈*w*.

Definition A.1.1. *A $\langle M, m, m', t\rangle$-secure sketch for the metric space $\{M\}$ is a pair of randomized procedures, sketching procedure (SS) and recovery procedure (Rec), defined as follows.*

- *The sketching procedure (SS) takes w ∈ M as input and returns a public helper data* $s \in \{0,1\}^*$.
- *The recovery procedure (Rec) takes w' ∈ M and s as input and outputs w.*

Secure sketch have to satisfy the following properties:

- ***Correctness or error tolerance***: $\forall w \in M$, $s \leftarrow \mathsf{SS}(w)$: $\mathsf{Rec}(w', s) = w$ if *dis(w,w')* $\leq t$.
- ***Security***: It requires that *s* does not leak too much information about *w*. Specifically, for any random variable *W* over *M* with min-entropy *m*, the secure sketch must ensure that average conditional min-entropy of *W* given *SS(W)* is at least *m'*. $\forall W \in M$, i.e. $H_\infty(W|\mathsf{SS}(W)) \geq m'$.

A.1.4 Randomness Extraction.

A randomness extractor (RE) [61] is a method to drive a key from an available non-uniform randomness source. It is used to derive near-uniform or 'perfect' randomness from non-uniform or 'imperfect' randomness sources. So, this perfect randomness can be used to generate uniform keys that can be used in cryptographic applications. Randomness extractor can be constructed from cryptographic hash functions.

Definition A.1.2. *Let Ext :* $\{0,1\}^n \rightarrow \{0,1\}^\ell$ *be a polynomial time probabilistic function which uses r bits of randomness. We say that Ext is an efficient* (n, m, ℓ, ϵ)*-strong extractor if for all random variables* $Y \leftarrow \{\{0,1\}\}^n$ *with* $H_\infty(Y) \geq m$*, it holds that:* $SD((Ext(Y;R),R);(U_\ell,R)) \leq \epsilon$*, where R and* U_ℓ *are uniform on* $\{0,1\}^r$ *and* $\{0,1\}^\ell$*, respectively.*

A.1.5 Fuzzy Extractor

Fuzzy Extractor (FE) [133] extracts uniformly random string from noisy non-uniform random data. This string can be used as a secure key in cryptographic applications. Definition A.1.3 formally defines a fuzzy extractor.

Definition A.1.3. *An* $\langle M, m, \ell, t, \epsilon\rangle$*-fuzzy extractor is a pair of randomized procedures, Generate (Gen) and Reproduce (Rep), satisfying the following properties.*

- *Gen generates, for a given input w ∈ M, an extracted string* $k \in \{0,1\}^\ell$ *and a helper data string* $p \in \{0,1\}^*$*, i.e.:*

  $\mathsf{Gen}(w) \rightarrow (k, P)$.
- *Rep takes an element w' ∈ W and a bit string* $p \in \{0,1\}^*$ *as inputs and produces an extracted string* $k' \in \{0,1\}^\ell$*, such that:*

  $\mathsf{Rep}(w', P) \rightarrow K$ `if` $dis(w; w') \leq t$.

FE satisfies the following properties.

- The correctness property of FE guarantees that if $dis(w, w') \leq t$ and (k, P) were generated by $(k, P) \leftarrow \mathsf{Gen}(w)$, then $\mathsf{Rep}(w', P) = k$. Otherwise, no guarantee is provided about the output of Rep.
- The security property guarantees that for any distribution W on M of min-entropy *m*, the string K is nearly uniform even for those who observe P: `if` (K, P) $\leftarrow \mathsf{Gen}(W)$, `then` SD ((K, P), (U,$P$)) $\leq \epsilon$.

A.2 Linear Block Codes

Error correction code (ECC) is commonly used to correct errors in data that are transmitted over noisy communication channels. When the sender wants to send a U binary information sequence, first, this latter is segmented into message blocks of fixed length k, denoted by m. Also, for a secure transition, the sender encodes each message block to so-called $codeword$ c, ($c = encode(m)$) with redundant information in the form of an ECC. The *codeword* is transmitted over a noisy channel to the receiver, and at the reception phase, the receiver checks, if there exist errors in the received *codeword* of data U (error detection). If an error found, ECC is used to reconstruct and recover the original and the true message m (m=decode(c)). The redundancy information allows the receiver to detect a fixed and limited number of errors that may occur in the message during the transition process, and to correct these errors without retransmitting the *codeword*.

Since we deal with only binary code, firstly, we define the finite field GF(2) of two elements as follows.

Definition A.2.4. *Given the binary field $GF(2) = \{0, 1\}$, we have:*

- *A binary word w of length n over $GF(2)$ is an n-tuple $w = \{w_0, \ldots, w_{n-1}\}$ of binary digits $w_i \in GF(2)$, $\forall i = 0, 1, 2, \ldots, n-1$.*
- *A binary block code of length n over $GF(2)$ is a nonempty set C of binary words w of length n each.*
- *Each element w of C is called a codeword in C.*

A set of 2^k distinct *codewords* w of length n each, over the binary field $GF(2) = \{0, 1\}$, is called a Binary Block Code $C(n, k)$. The latter is linear if for any two *codewords* in (n, k) results in another *codeword* in (n, k). (n, k, t) is a linear Binary Block Code which can correct t errors. A *codeword* $c \in C$ represents an $n-$bit sequence, formed from the *k* bit messages $m \in M$ and (nk) parity bits used to recover the transmitted *codeword* from errors. Linear block codes can detect up to $d - 1$ errors and

correct up to $t = (d-1)/2$ errors in the *n*-element *codeword*, where, *d* is the minimum number of bits in which any two distinct *codewords*.

A.2.1 Encoding

In linear block codes, the message m is encoded by multiplication with a generator matrix G to yield a *codeword* c:

$$c = m \cdot G \tag{A.1}$$

Recall that c is a $(1 \times n)$ row vector and m is a $(1 \times k)$ row vector. G has a dimension of $k \times n$, its rows are linearly independent and form a basis for C. In general, several G matrices generate the same code book, and only the mapping order from m to c changes. Two codes with the same code book are considered equivalent, no matter the order of the rows in C. A linear block code C is referred to as a (n, k) block code. For block codes, we can also define the matrix H, of dimension $(k \times n)$, which is called the parity check matrix of the code, such that c is a valid *codeword* $c \in C$, where:

$$c \cdot H^T = H^T \cdot c = 0 \tag{A.2}$$

H can therefore be used to validate that received data is indeed a *codeword*. If the received data r is not a valid *codeword*, then:

$$r \cdot H^T \neq 0 \tag{A.3}$$

Systematic codes are codes that still contain the original, unaltered, message bits. Their codewords can be partitioned into two parts: one containing the message and one containing the redundant parity bits. This makes them easy to decode. To achieve this their generator matrix is presented in the standard systematic form.

$$G = [P; I_k] \tag{A.4}$$

I_K is the $(k \times k)$ identity matrix, which keeps the *k* original bits in the *codeword* and *P* is the $k \times (n-k)$ parity matrix, which is unique to the code and generates the n redundant (parity) bits. Any linear block code can be put into systematic form, *i.e.* G of any linear block code can be written in the standard form. Hence, the parity check matrix can be constructed from their generator matrix as follows:

$$H = [I_{n-k}; P^T] \tag{A.5}$$

A.2.2 Decoding.

For the error correction in a linear block code, we correct invalid received data to the closest valid *codeword*. For this we define s, the syndrome, for a received vector r, where H^T is the transpose of H, by:

$$s = r \cdot H^T \tag{A.6}$$

It is known that an error has occurred (i.e. $r \notin C$) if $s \neq 0$. The vector e containing the occurred errors is defined as:

$$e = r + c \tag{A.7}$$

Recall that there is no difference between addition and subtraction in $GF(2)$, therefore also $r = e + c$. This leads to:

$$s = (e + c).H^T = e \cdot H^T + 0 = e \cdot H^T \tag{A.8}$$

If the syndrome as defined in Eq. A.6 is not equal to 0, then a value of e is chosen which satisfies Eq. A.8 with the lowest possible $w(e)$. In other words, we assume that the errors which occurred is the difference between r and the closest valid *codeword* c. The valid *codeword* which was transmitted, v, is then given by $v = e + r$. So, the message represented by v is then recovered. This is called minimum distance decoding.

Example A.2.1. *To transmit the data U using the code word (n, k), first U needs to be segmented into a set of messages of k information digits. Let $m = (m_0, \ldots, m_{k-1})$ be the message to be encoded, using Eq A.1, the corresponding code word is $c = (c_0, \ldots, c_{n-1})$= $(m_0, \ldots, m_{k-1}).G$. To encode the message $m = (110)$, we use $(6, 3)$ which has the following generator matrix and the transposed parity-check-matrix.*

$$G = \begin{bmatrix} g_0 \\ g_1 \\ g_2 \\ g_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}; H = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$$

$$c = m \cdot G = m_0 \cdot g_0, m_1 \cdot g_1, m_2 \cdot g_2 = 1 \cdot g_0, 0 \cdot g_1, 1 \cdot g_2 = 110100 + 011010 = 101110 \tag{A.9}$$

The codeword corresponding to the message (110) is (101110) which is divided into two parts, $n - k$ parity-check digits (101) and the message part $m = 110$. At the receiving side, the code word c is received as r. To recover the encoded message

m, the decoding procedure is needed. According to Eq. A.6, first we calculate the so called syndrome $s = r \cdot H^T$, if $s = 0$ (all-zeros vector), this means that r is a valid codeword and we may conclude that there is no error in r. Or, since $r = c + e$, if $s = r \cdot H^T = 0$ this means e is also a codeword which is known as undetectable error. Otherwise ($s \neq 0$), the received codeword r contains errors.

A.2.3 Locating the Error Pattern.

To locate this error, we refer to Eq. A.7, where $r = (r_0, r_1, r_2, \ldots, r_{n-1})$ represents a received *codeword* of *n* elements, and $e = (e_0, e_1, e_2, \ldots, e_{n-1})$ which referred to as the *error pattern*. After assuming that r has errors bits, Eq. A.8 is used to locate the error pattern.

Based on (6,3) code, we use Eq. A.8 to determine the syndrome corresponding to each correctable error pattern, with computing $e.H^T$, as follows.

$$s = (e_0, e_1, e_2, e_3, e_4, e_5) \cdot H^T$$

Table A.1 Syndrome lookup table.

Error pattern e	Syndrome s
000000	000
000001	101
000010	011
000100	110
001000	001
010000	010
100000	100
010001	111

Since each of the mentioned error patterns of the results listed in Table A.1 has a one-to-one relationship with each syndromes, so solving for a syndrome identifies the specific error pattern that corresponds to that syndrome. As we showed by using (6, 3) code, the corespondent *codeword* of $m = 110$ is $c = 101110$, assume that the vector $r = 001110$ is received. From Eq. A.7, we compute the syndrome by: $s = (001110) \cdot H^T = 100$. From Table A.1, we can verify that the error pattern is $e = 100000$. Then, using Eq. A.7, the corrected vector is estimated by $c^* = r+e = 001110+100000 = 101110$ where c^* is the actual transmitted code word. The error correction procedure yields $c^* = c$, which means that the output m^* will correspond to the actual message $m^* = m = 110$. The possible codewords for the code (6,3) are: (000000, 001101, 010110, 011011, 100111, 101010, 110001, 111100). For this list of *codewords*, the minimal distance is $d = 3$, so (6,3) can detect up to 2 errors and correct up to $t = 1$ error, and it can be

given as (6,3,1).

A.3 T2S-MAKEP Verifpal Code

```
attacker[active]

principal Thing_A[
knows private ID_A
    generates TS_1
    H_ID_A=HASH(ID_A)
    m_1=HASH(ID_A,TS_1)
]
Thing_A → Trusted_server:H_ID_A,TS_1,m_1
principal Trusted_server[
knows private ID_A
_=ASSERT(HASH(ID_A),H_ID_A)
_=ASSERT(HASH(ID_A,TS_1),m_1)
]
principal Trusted_server[
    knows public C_A_i
    knows private k_A_i
    generates TS_2
    m_2=HASH(ID_A,C_A_i,k_A_i,TS_2)
]
Trusted_server → Thing_A:TS_2,m_2
principal Thing_A[
    knows private k_A_i
    _=ASSERT(HASH(ID_A,C_A_i,k_A_i,TS_2),m_2)
]
principal Thing_A[
    generates TS_3
    knows private R_A_ii
    m_3=HASH(ID_A,TS_3,k_A_i,R_A_ii)
    m_4=ENC(k_A_i,R_A_ii)
]
Thing_A → Trusted_server:TS_3,m_3,m_4
principal Trusted_server[
    R_A_ii_=DEC(k_A_i,m_4)
    _=ASSERT(HASH(ID_A,TS_3,k_A_i,R_A_ii_),m_3)
]

queries[
authentication? Thing_A → Trusted_server: m_1
authentication? Trusted_server → Thing_A: m_2
authentication? Thing_A → Trusted_server: m_3
authentication? Thing_A → Trusted_server: m_4
confidentiality? ID_A
confidentiality? k_A_i
confidentiality? R_A_ii
freshness? m_1
freshness? m_2
freshness? m_3
]
```

A.4 T2T-MAKEP Verifpal Code

```
attacker[active]

principal Thing_B[
    knows private ID_B
]
principal Thing_A[
    generates TS_1
    knows private ID_A
    knows public H_ID_B
    H_ID_A=HASH(ID_A)
    m_1=HASH(ID_A,H_ID_B,TS_1)
]
Thing_A → Trusted_server:H_ID_A,TS_1,m_1
principal Trusted_server[
    knows private ID_A
    knows private ID_B
    _=ASSERT(HASH(ID_B),H_ID_B)
    _=ASSERT(HASH(ID_A),H_ID_A)
    _=ASSERT(HASH(ID_A,H_ID_B,TS_1),m_1)
    knows private k_A_i
]
principal Thing_A[
    knows private k_A_i
]
principal Trusted_server[
    knows public C_B_i
    knows private k_B_i
    generates TS_2
    m_2=HASH(ID_A,C_B_i,k_B_i,TS_2)
    m_3=ENC(k_A_i,k_B_i)
]
Trusted_server → Thing_A :TS_2,m_2,m_3
principal Thing_A[
    k_B_i_=DEC(k_A_i,m_3)
      _=ASSERT(HASH(ID_A,C_B_i,k_B_i_,TS_2),m_2)
    ]
principal Thing_A[
    generates TS_3
    m_4=HASH(C_B_i,k_A_i,TS_3)
    m_5=ENC(k_B_i_,k_A_i)
    SAB=HASH(CONCAT(k_A_i,k_B_i_))
    ]
Thing_A → Thing_B:H_ID_A,TS_3,m_4,m_5
principal Thing_B[
    knows private k_B_i
    k_A_i_=DEC(k_B_i,m_5)
    _=ASSERT(HASH(C_B_i,k_A_i_,TS_3),m_4)
    ]
principal Thing_B[
    SBA=HASH(CONCAT(k_A_i_,k_B_i))
]
principal Thing_B[
    generates TS_4
    knows private R_B_ii
    m_6=HASH(ID_B,TS_4,k_B_i,R_B_ii)
    m_7=ENC(k_B_i,R_B_ii)
]
Thing_B → Trusted_server:TS_4,m_6,m_7
```

```
principal Trusted_server[
    R_B_ii_=DEC(k_B_i,m_7)
    _=ASSERT(HASH(ID_B,TS_4,k_B_i,R_B_ii_),m_6)
]
queries[
authentication? Thing_A → Trusted_server: m_1
authentication? Trusted_server → Thing_A: m_2
authentication? Trusted_server → Thing_A: m_3
authentication? Thing_A → Thing_B: m_4
authentication? Thing_A → Thing_B: m_5
authentication? Thing_B → Trusted_server: m_6
authentication? Thing_B → Trusted_server: m_7
confidentiality? ID_A
confidentiality? ID_B
confidentiality? k_A_i
confidentiality? k_B_i
confidentiality? R_B_ii
freshness? m_1
freshness? m_2
freshness? m_4
freshness? m_6
]
```

A.5 LT2S-MAKEP Verifpal Code

```
attacker[active]

principal Thing_A[
    knows private ID_A
    generates TS_1
    H_ID_A=HASH(ID_A)
    m_1=HASH(ID_A,TS_1)
]
Thing_A → Trusted_server:H_ID_A,TS_1,m_1
principal Trusted_server[
knows private ID_A
_=ASSERT(HASH(ID_A),H_ID_A)
_=ASSERT(HASH(ID_A,TS_1),m_1)
]
principal Trusted_server[
    knows public C_A_i
    knows private k_SRAM_A
    knows private R_APUF_A_i
    generates TS_2
    m_2=HASH(ID_A,C_A_i,TS_2)
]
Trusted_server → Thing_A:TS_2,m_2

principal Thing_A[
_=ASSERT(HASH(ID_A,C_A_i,TS_2),m_2)
]

principal Thing_A[
    knows private k_SRAM_A
    knows private R_APUF_A_i_
    generates TS_3
    m_3=HASH(ID_A,TS_3,R_APUF_A_i_)
    m_4=ENC(k_SRAM_A,R_APUF_A_i_)
]

Thing_A → Trusted_server:TS_3,m_3,m_4

principal Trusted_server[
    R_APUF_A_i__=DEC(k_SRAM_A,m_4)
    _=ASSERT(HASH(ID_A,TS_3,R_APUF_A_i__),m_3)
]
principal Trusted_server[
    _=ASSERT(R_APUF_A_i__,R_APUF_A_i)
]
principal Trusted_server[
    k_SA_i=HASH(R_APUF_A_i__)
]
principal Thing_A[
    k_AS_i=HASH(R_APUF_A_i_)
]
principal Thing_A[
    generates TS_4
    knows private R_APUF_A_ii
    m_5=HASH(ID_A,TS_4,R_APUF_A_ii)
    m_6=ENC(k_SRAM_A,R_APUF_A_ii)
]
Thing_A → Trusted_server:TS_4,m_5,m_6
principal Trusted_server[
```

```
    R_APUF_A_ii_=DEC(k_SRAM_A,m_6)
    _=ASSERT(HASH(ID_A,TS_4,R_APUF_A_ii_),m_5)
]
queries[
authentication? Thing_A → Trusted_server: m_1
authentication? Trusted_server → Thing_A: m_2
authentication? Thing_A → Trusted_server: m_3
authentication? Thing_A → Trusted_server: m_4
authentication? Thing_A → Trusted_server: m_5
authentication? Thing_A → Trusted_server: m_6
confidentiality? ID_A
confidentiality? k_SA_i
confidentiality? k_AS_i
confidentiality? k_SRAM_A
confidentiality? R_APUF_A_i_
confidentiality? R_APUF_A_ii
freshness? m_1
freshness? m_2
freshness? m_3
freshness? m_5
]
```