PRACTICAL LIGHTWEIGHT SECURITY: PHYSICAL UNCLONABLE FUNCTIONS AND THE INTERNET OF THINGS

Vom Fachbereich Informatik der
Technischen Universität Darmstadt genehmigte

Dissertation

zur Erlangung des akademischen Grades
Doktor-Ingenieur (Dr.-Ing.)
von

Nikolaos Athanasios Anagnostopoulos

geboren in Thessaloniki, Griechenland

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Referenten:	Prof. Dr. Stefan Katzenbeisser Universität Passau
	Prof. Dr. Marc Fischlin Technische Universität Darmstadt
Tag der Einreichung:	27.10.2021
Tag der Prüfung:	10.03.2022

D17

Darmstadt, 2022

Practical Lightweight Security: Physical Unclonable Functions and the Internet of Things

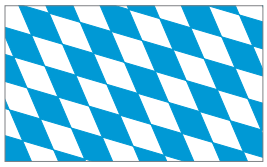
Praktische Leichtgewichtige Sicherheit: Physische Unklonbare Funktionen und das Internet der Dinge

Zur Erlangung des akademischen Grades Doktor-Ingenieur (Dr.-Ing.)
genehmigte Dissertation von Nikolaos Athanasios Anagnostopoulos aus Thessaloniki, Griechenland
Tag der Einreichung: 27.10.2021, Tag der Prüfung: 10.03.2022
Darmstadt — D 17

1. Gutachten: Prof. Dr. Stefan Katzenbeisser, Universität Passau
2. Gutachten: Prof. Dr. Marc Fischlin, Technische Universität Darmstadt

TECHNISCHE UNIVERSITÄT DARMSTADT

Security Engineering Group
Computer Science Department



Practical Lightweight Security: Physical Unclonable Functions and the Internet of Things
Praktische Leichtgewichtige Sicherheit: Physische Unklonbare Funktionen und das Internet der Dinge

Genehmigte Dissertation von Nikolaos Athanasios Anagnostopoulos aus Thessaloniki, Griechenland

1. Gutachten: Prof. Dr. Stefan Katzenbeisser, Universität Passau
2. Gutachten: Prof. Dr. Marc Fischlin, Technische Universität Darmstadt

Tag der Einreichung: 27.10.2021
Tag der Prüfung: 10.03.2022

Darmstadt — D 17

Erklärungen zur Dissertation gemäß §8 und §9 der Promotionsordnung der TU Darmstadt

Ich versichere hiermit, dass die elektronische Version meiner Dissertation mit der schriftlichen Version übereinstimmt. Hiermit bestätige ich, dass zu einem vorherigen Zeitpunkt noch keine Promotion versucht wurde. Hiermit versichere ich auch, dass die vorliegende Dissertation selbstständig und nur unter Verwendung der angegebenen Quellen verfasst wurde. Die Arbeit hat bisher noch nicht zu Prüfungszwecken gedient.

Declarations pursuant to §8 and §9 of the Doctoral Regulations of TU Darmstadt*

I hereby certify that the electronic version of my Dissertation matches the written version. Hereby I confirm that no doctorate has been attempted before. Hereby I also certify that the present dissertation was written independently and only using the sources indicated. The work has so far not served for examination purposes.

Darmstadt, den 27.10.2021

(Nikolaos Athanasios Anagnostopoulos)

* English translation for information purposes only.

Dedication

The author would like to dedicate this work to all those people who accepted him, put up with him, and helped him, both in his personal life and in finishing his doctoral studies, including his parents, sister and extended family, his friends and acquaintances, his supervisors and colleagues, and in particular his supervising professor, Prof. Dr. Stefan Katzenbeisser, as well as the other members of his doctoral examination committee, namely, Prof. Dr. Marc Fischlin, Prof. Dr. Reiner Hähnle, Prof. Dr. Thomas Schneider, and Prof. Dr. Sebastian Faust.

The author is an avid supporter and proud member of the Greek sports club named Άρης, which is based in his hometown, Thessaloniki, in Greece, and takes an interest in the Dutch football club of Twente, which is based in Enschede, in the Netherlands, and in the German football club of Darmstadt, which is based in the German city of the same name.

Finally, the author would also like to thank God for his achievements so far. Nevertheless, the author would like to note his agreement with the following statement:

> "Life is unfair, but you gotta make the best of whatever you have..."
>
> — inspired by "Malcolm in the Middle", a television series running from 2000 to 2006.

Acknowledgements

The author would like to thank the German Research Foundation (Deutsche ForschungsGemeinschaft – DFG) for supporting his research within Project P3: "Hardware-Entangled Security" (project number 236615297) of the DFG Collaborative Research Centre (CRC): "CROSSING – Cryptography-Based Security Solutions: Enabling Trust in New and Next Generation Computing Environments" (SonderForschungsBereich – SFB 1119), as well as within Projects: "PUFMem: Intrinsic Physical Unclonable Functions from Emerging Non-Volatile Memories" (project number 440182124) and "NANOSEC: Tamper-Evident PUFs Based on Nanostructures for Secure and Robust Hardware Security Primitives" (project number 439892735) of the Priority Program "Nano Security: From Nano-Electronics to Secure Systems" (SchwerPunktProgramme – SPP 2253).

The author would also like to thank the following institutions for supporting his university studies so far:

- the Aristotle University of Thessaloniki, located in Greece,
- the EIT Digital Master School, located in the European Union,
- the Technical University of Berlin, located in Germany,
- the University of Twente, located in the Netherlands,
- the Technical University of Eindhoven, also located in the Netherlands,
- the Technical University of Darmstadt, located in Germany,
- the Technical University of München, also located in Germany, and
- the University of Passau, also located in Germany.

Editorial note

Throughout this doctoral dissertation, the author uses the terms “we”, “our”, etc., in order to describe his work. This is done in order to underline that research is always a cooperative effort and that the author would not have been able to publish most of his research, if any at all, if other people had not taken time off of their own work to review and discuss the author’s work. The author is deeply grateful for their time, effort and help. Additionally, the author makes use of relevant segments of his published works, noting, in the beginning of each section, the works on which the relevant section is partially based upon.

Abstract

In this work, we examine whether Physical Unclonable Functions (PUFs) can act as lightweight security mechanisms for practical applications in the context of the Internet of Things (IoT). In order to do so, we first discuss what PUFs are, and note that memory-based PUFs seem to fit the best to the framework of the IoT. Then, we consider a number of relevant memory-based PUF designs and their properties, and evaluate their ability to provide security in nominal and adverse conditions. Finally, we present and assess a number of practical PUF-based security protocols for IoT devices and networks, in order to confirm that memory-based PUFs can indeed constitute adequate security mechanisms for the IoT, in a practical and lightweight fashion.

More specifically, we first consider what may constitute a PUF, and we redefine PUFs as inanimate physical objects whose characteristics can be exploited in order to obtain a behaviour similar to a highly distinguishable (i.e., "(quite) unique") mathematical function. We note that PUFs share many characteristics with biometrics, with the main difference being that PUFs are based on the characteristics of inanimate objects, while biometrics are based on the characteristics of humans and other living creatures. We also note that it cannot really be proven that PUFs are unique per instance, but they should be considered to be so, insofar as (human) biometrics are also considered to be unique per instance.

We, then, proceed to discuss the role of PUFs as security mechanisms for the IoT, and we determine that memory-based PUFs are particularly suited for this function. We observe that the IoT nowadays consists of heterogeneous devices connected over diverse networks, which include both high-end and resource-constrained devices. Therefore, it is essential that a security solution for the IoT is not only effective, but also highly scalable, flexible, lightweight, and cost-efficient, in order to be considered as practical. To this end, we note that PUFs have been proposed as security mechanisms for the IoT in the related work, but the practicality of the relevant security mechanisms has not been sufficiently studied.

We, therefore, examine a number of memory-based PUFs that are implemented using Commercial Off-The-Shelf (COTS) components, and assess their potential to serve as *acceptable* security mechanisms in the context of the IoT, not only in terms of effectiveness and cost, but also under both nominal and adverse conditions, such as ambient temperature and supply voltage variations, as well as in the presence of (ionising) radiation. In this way, we can determine whether memory-based PUFs are truly suitable to be used in the various application areas of the IoT, which may even involve particularly adverse environments, e.g., in IoT applications involving space modules and operations.

Finally, we also explore the potential of memory-based PUFs to serve as adequate security mechanisms for the IoT in practice, by presenting and analysing a number of cryptographic protocols based on these PUFs. In particular, we study how memory-based PUFs can be used for key generation, as well as device identification, and authentication, their role as security mechanisms for current and next-generation IoT devices and networks, and their potential for applications in the space segment of the IoT and in other adverse environments. Additionally, this work also discusses how memory-based PUFs can be utilised for the implementation of lightweight reconfigurable PUFs that allow for advanced security applications. In this way, we are able to confirm that memory-based PUFs can indeed provide flexible, scalable, and efficient security solutions for the IoT, in a practical, lightweight, and inexpensive manner.

Zusammenfassung

In dieser Arbeit untersuchen wir, ob Physische Unklonbare Funktionen (PUFs) als leichtgewichtige Sicherheitsmechanismen für praktische Anwendungen im Kontext des Internets der Dinge (IdD) fungieren können. Zu diesem Zweck diskutieren wir zunächst, was eine PUF ist, und stellen fest, dass speicherbasierte PUFs am besten in den Rahmen des Internets der Dinge passen. Anschließend betrachten wir eine Reihe relevanter Designs für speicherbasierte PUFs und ihre Eigenschaften, und bewerten ihre Fähigkeit, unter nominellen und ungünstigen Bedingungen Sicherheit zu bieten. Schließlich präsentieren und bewerten wir eine Reihe praktischer PUF-basierter Sicherheitsprotokolle für IdD-Geräte und -Netzwerke, um zu zeigen, dass speicherbasierte PUFs auf praktische und leichtgewichtige Weise tatsächlich angemessene Sicherheitsmechanismen für das IdD sind.

Insbesondere betrachten wir zuerst, was eine PUF ausmachen kann, und definieren PUFs als leblose physische Objekte neu, deren Eigenschaften ausgenutzt werden können, um ein Verhalten zu erlangen, das einer hoch unterscheidbaren (d.h. „(ganz) einzigartigen“) mathematischen Funktion ähnlich ist. Wir stellen fest, dass PUFs viele Merkmale mit der Biometrie teilen, mit dem Hauptunterschied, dass PUFs auf den Eigenschaften lebloser Objekte basieren, während Biometrie auf den Eigenschaften von Menschen und anderen Lebewesen basiert. Wir stellen außerdem fest, dass es nicht wirklich bewiesen werden kann, dass PUFs pro Instanz einzigartig sind, aber trotzdem als solche betrachtet werden sollten, sofern (menschliche) Biometrie auch als pro Instanz einzigartig angesehen werden kann.

Anschließend diskutieren wir die Rolle von PUFs als Sicherheitsmechanismen für das IdD und ermitteln, dass speicherbasierte PUFs für diese Funktion besonders gut geeignet sind. Wir bemerken hier, dass das IdD heutzutage aus heterogenen Geräten besteht, die über verschiedene Netzwerke verbunden sind, die sowohl High-End- als auch ressourcenbeschränkte Geräte umfassen. Daher ist es wichtig, dass eine Sicherheitslösung für das IdD nicht nur effektiv, sondern auch hoch skalierbar, flexibel, leichtgewichtig und kosteneffizient ist, um als praktisch angesehen werden zu können. Zu diesem Zweck stellen wir fest, dass PUFs in der entsprechenden Arbeit als Sicherheitsmechanismen für das IdD vorgeschlagen wurden, die Praktikabilität der relevanten Sicherheitsmechanismen jedoch nicht ausreichend untersucht wurde.

Wir untersuchen daher eine Reihe von speicherbasierten PUFs, die mithilfe von kommerziellen Standardkomponenten („von der Stange“) implementiert werden, und bewerten ihr Potenzial, als *akzeptable* Sicherheitsmechanismen im Kontext des IdD zu dienen; nicht nur im Hinblick auf die Effektivität und Kosten, aber auch sowohl unter nominellen als auch unter ungünstigen Bedingungen wie Schwankungen der Umgebungstemperatur und der Versorgungsspannung sowie in Gegenwart von (ionisierender) Strahlung. Auf diese Weise können wir bestimmen, ob speicherbasierte PUFs wirklich für die Verwendung in den verschiedenen Anwendungsbereichen des IdD geeignet sind, die sogar besonders ungünstige Umgebungen betreffen können, z.B. in IdD-Anwendungen mit Weltraummodulen und -operationen.

Schließlich untersuchen wir auch das Potenzial speicherbasierter PUFs, als angemessene Sicherheitsmechanismen für das IdD in der Praxis zu dienen, indem wir eine Reihe von kryptografischen Protokollen, die auf diesen PUFs basieren, präsentieren und analysieren. Insbesondere untersuchen wir, wie speicherbasierte PUFs für die Schlüsselgenerierung sowie die Geräteidentifikation und -authentifizierung verwendet werden können, ihre Rolle als Sicherheitsmechanismen für IdD-Geräte und -Netzwerke der aktuellen und nächsten Generation und ihr Potenzial für Anwendungen im Weltraum-Segment des IdD und in anderen ungünstigen Umgebungen. Darüber hinaus wird in dieser Arbeit erläutert, wie speicherbasierte PUFs für die Implementierung von leichtgewichtigen, rekonfigurierbaren PUFs verwendet werden können, die erweiterte Sicherheitsanwendungen ermöglichen. Auf diese Weise können wir bestätigen, dass speicherbasierte PUFs tatsächlich flexible, skalierbare und effiziente Sicherheitslösungen für das IdD auf praktische, leichtgewichtige, und kostengünstige Weise bereitstellen können.

Contents

1 Introduction

Partially Based Upon Peer-Reviewed Content Published In

- N. A. Anagnostopoulos, T. Arul, Y. Fan, J. Lotichius, C. Hatzfeld, F. Fernandes, R. Sharma, F. Tehranipoor & S. Katzenbeisser, "Securing IoT Devices Using Robust DRAM PUFs", Proceedings of the 2018 Global Information Infrastructure and Networking Symposium (GIIS 2018), IEEE, 2018. DOI: 10.1109/GIIS.2018.8635789
- N. A. Anagnostopoulos, T. Arul, Y. Fan, M. Kumar & S. Katzenbeisser, "AR-PUFs: Advanced Security Primitives for the Internet of Things and Cyber-Physical Systems", Proceedings of the 2019 IEEE International Conference on Consumer Electronics (ICCE 2019), IEEE, 2019. DOI: 10.1109/ICCE.2019.8661840
- L. Negka, G. Gketsios, N. A. Anagnostopoulos, G. Spathoulas, A. Kakarountas & S. Katzenbeisser, "Employing Blockchain and Physical Unclonable Functions for Counterfeit IoT Devices Detection", Proceedings of the 1st International Conference on Omni-Layer Intelligent Systems (COINS 2019), pp. 172-178, ACM, 2019. DOI: 10.1145/3312614.3312650
- N. A. Anagnostopoulos, S. Ahmad, T. Arul, D. Steinmetzer, M. Hollick & S. Katzenbeisser, "Low-Cost Security for Next-Generation IoT Networks", ACM Transactions on Internet Technology, vol. 20, iss. 3, art. 30, ACM, 2020. DOI: 10.1145/3406280
- N. Mexis, N. A. Anagnostopoulos, S. Chen, J. Bambach, T. Arul & S. Katzenbeisser, "A Lightweight Architecture for Hardware-Based Security in the Emerging Era of Systems of Systems", ACM Journal on Emerging Technologies in Computing Systems, vol. 17, iss. 3, art. 43, ACM, 2021. DOI: 10.1145/3458824
- N. Mexis, N. A. Anagnostopoulos, S. Chen, J. Bambach, T. Arul & S. Katzenbeisser, "A Design for a Secure Network of Networks Using a Hardware and Software Co-Engineering Architecture", Proceedings of the SIGCOMM 2021 Poster and Demo Sessions (SIGCOMM 2021), pp. 65-67, ACM, 2021. DOI: 10.1145/3472716.3472849

In recent years, Physical Unclonable Functions (PUFs) have been proposed as efficient lightweight security mechanisms, especially for resource-constrained devices. PUFs are inanimate physical objects, the characteristics of which allow them to act as physical implementations of highly distinct functions. Therefore, PUFs can be utilised for the identification and authentication of inanimate physical objects, such as digital chips, as well as for the generation of random numbers and secure cryptographic keys. However, over the last few years, the capability of PUFs to serve as security primitives has been disputed, due to a number of successful attacks against them being proposed in the relevant literature [1, 2, 3, 4, 5, 6, 7, 8, 9]. In order to address the doubt cast on the potential of PUFs to serve as adequate security mechanisms, this work explores the suitability of memory-based PUFs for the implementation of practical and lightweight security solutions, especially in the context of the Internet of Things (IoT) [1].

[1] The Internet of Things (IoT) refers to a network of devices where data are exchanged, processed, and utilised by the different sensors, actuators, and other electronic devices connected to this network, potentially also leading into actions being taken, without direct human intervention, supervision, or control. Within this framework, devices communicate with each other, and decide on different actions that should be performed by the network's actuators, based on the data received from the various sensors of the network, as well as from other electronics, and according to a set of

In particular, this work first examines and attempts to redefine the concept of PUFs, in general, with particular emphasis on the nature of PUFs. Subsequently, PUFs are discussed as a solution to the need for cost-efficient and practical security, especially with regards to the IoT. In this context, the relation between cost and security is discussed, in order to investigate whether memory-based PUFs, which tend to be intrinsic components of IoT devices, can be considered as a practical and viable security mechanism for such devices.

Additionally, as it is noted that IoT devices and modules need to operate under diverse, and potentially adverse, conditions, the robustness of well-known implementations of memory-based PUFs to abrupt changes in ambient environmental factors, is intensively tested. More specifically, the quality of the responses of such PUFs under such adverse conditions as temperature and voltage variations, as well as the presence of nuclear radiation, is examined in detail, in order to assess the security properties of these PUFs under such conditions, and determine their aptitude for the implementation of security mechanisms in the context of the IoT. For this purpose, this work presents and discusses a number of well-known metrics that can be utilised to assess and evaluate the quality of the responses acquired from such PUFs, in order to estimate the level of security that these PUFs can provide.

Finally, a number of PUF-based protocols and security schemes are described and discussed in the context of IoT applications, in order to validate the security of such PUF implementations in real use case scenarios. In this way, we are able to analyse the security that memory-based PUFs can provide in a comprehensive and extensive manner, in order to determine whether they can truly be utilised for the implementation of practical and lightweight security mechanisms, especially in the context of cost-efficient and resource-restrained applications.

1.1 A New Definition for PUFs

Physical Unclonable Functions (PUFs) are inanimate physical objects whose characteristics can be exploited in order to obtain a behaviour similar to a mathematical function. This means that for each input, which is referred to as a *challenge*, each PUF should produce only a particular output, which is referred to as a *response*. PUFs are most often based on the exploitation of minor manufacturing variations, which cause the characteristics of the physical objects that serve as PUFs to be highly identifiable and, in most cases, (quite) unique, per PUF instance.

The uniqueness of the characteristics upon which the functionality of PUFs is based, and the near impossibility of controlling the manufacturing process in such a way as to reproduce the minor manufacturing variations that lead to each PUF implementing a specific, yet unpredictable (without a measurement), function, drove researchers to the conclusion that PUFs are unclonable [10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23]. However, this conclusion has now been disproven for the most well-known implementations of PUFs [1, 5, 6, 7, 8, 9, 24, 25, 26, 27, 28], therefore, the author believes that the term "Physical Unclonable Functions" should rather be replaced by the term *"Physical Unique Functions"*, which not only describes the current notion of PUFs in a more factual way, but also allows for a more comprehensive, yet also more precise, definition of the concept of PUFs.

Essentially, the concept of PUFs is highly associated with their application in digital security, where PUFs are used to provide unique security tokens that can be used to identify each relevant device, e.g.,

predefined rules. The IoT has found wide application both in space and in terrestrial applications, and especially in the implementation of such concepts as smart homes, smart cities, smart grids, and smart vehicles.

their responses, or be shared among devices as a common secret, e.g., keys based on their responses. Therefore, the defining property of PUFs is the uniqueness of the function that each of them implements, rather than its unclonability, which has been disproven, and which could never be fully guaranteed in the absence of *perfect* security. The unpredictability of PUFs also suffers from imperfections, as, in order to be utilised, PUFs need to be measured, through the acquisition of responses corresponding to specific selected challenges, in pairs known as Challenge-Response Pairs (CRPs). Finally, even the uniqueness of PUFs cannot be proven formally, as the comparison of all CRPs for all past, current and future PUFs is rather unfeasible even for a very specific type and model of a PUF. Therefore, the claim that PUF CRPs are unique per instance is based on statistical models and calculations, rather than being truly proven in a definitive manner. Nevertheless, assuming either that an adequately large number of responses (and, therefore, also of CRPs) exists or that each PUF response has an adequately large size, it can be easily proven that the probability of two PUFs implementing the same function becomes adequately low.

We note that the actual way in which PUFs are used requires their measurement, which precludes the factual existence of perfect security, as it reveals the exact nature and values of, at least, a subset of their CRPs, making the function implemented by a PUF somewhat predictable. However, this is a requirement for every security primitive and, in fact, the main reason why perfect security can never be achieved. In general, the PUF's function, as it transpires through its CRPs, should be known to legitimate parties and unknown to attackers. Therefore, the PUF's operation should actually be predictable by legitimate parties and unpredictable by attackers, a fact that holds true for the operation of every security primitive. Hence, although unpredictability is a crucial factor for the operation of PUFs as security mechanisms, it is not truly the most defining one. It is rather the assumed property of (almost) genuine uniqueness for the PUF CRPs that causes PUFs to be highly unpredictable, and thus allows for their utilisation as security mechanisms of adequate quality. In this way, it is guaranteed that even if a subset of PUFs are compromised, by being cloned or by becoming (more or less) predictable, potentially through the acquisition of a (large) subset of their CRPs, the ability of PUFs to serve as adequate security mechanisms will not be affected significantly and in a general manner, as the other PUF instances will probably not be affected. Therefore, we propose that the acronym "PUFs" should stand for *"Physical Unique Functions"* rather than for "Physical Unclonable Functions".

Moreover, PUFs have, quite often, been defined in different ways that have proven to be either factually incorrect or not fully adequate. In particular, the fact that PUFs are physical objects has often been overlooked in literature relevant to the role of PUFs in cryptography, while, sometimes, the security properties of certain PUF implementations have been used in order to define the overall concept of PUFs. More specifically, PUFs have sometimes been *defined* not only as unclonable, but also as tamper-proof, tamper-resistant, or tamper-evident [12, 13, 14, 15, 16, 17, 19, 20, 21, 22, 23, 29], or as a generic set of functions with specific security properties [30, 31, 32]. Adopting these definitions, however, would mean that most well-known examples of PUFs cannot truly be considered as PUFs, since they do not conform to the corresponding definitions, or that also non-physical functions, such as common mathematical functions, could perhaps be considered as PUFs. In order to address such issues, we propose that PUFs be defined as inanimate physical objects whose characteristics can be exploited in order to obtain a behaviour similar to a highly distinguishable (i.e., "(quite) unique") mathematical function [2].

[2] While this definition may not seem precise enough, in reality, it is as precise as it is possible in order to provide an accurate description of PUFs. In general, physical objects, such as PUFs, are extremely difficult to characterise in a precise, yet

1.2 On the Nature of PUFs

As already mentioned, the existence of PUFs is based on minor manufacturing imperfections, i.e., on the unpredictability of the minuscule operations of the manufacturing process, of inanimate physical objects. Essentially, PUFs are very similar to human biometrics. Much like biometrics are based on the uniqueness of humans, with each human exhibiting slightly different physical characteristics, PUFs are based on the uniqueness of inanimate physical objects, with each inanimate physical object possessing slightly different physical characteristics. Additionally, the human characteristics used as biometrics are mostly based on the way each human being is "created" or "produced", i.e., conceived, and may change over time or due to external factors. Similarly, the physical characteristics that are used as PUFs are based on the way each inanimate physical object is created or produced, and also may change over time or due to external factors. Therefore, one could easily claim that PUFs are essentially the "biometrics" of inanimate physical objects [3].

The comparison between PUFs and human biometrics becomes especially relevant in the context of security. Much like PUFs, human biometrics are not truly unclonable and remain somehow unpredictable, until such time as they are measured. Most importantly, the security of human biometrics is based on their (relative) uniqueness, i.e., on the fact that they are highly distinguishable. Again, much like PUFs, the uniqueness of human biometrics cannot truly be proven, not only because the acquisition of the biometrics of all humans currently alive is an unfeasible task, but also because there is no way to measure the biometrics of all the humans that lived in the past or will be born in the future, in the same way that we cannot really measure all the PUFs that currently exist, and even if that was feasible, there would be no way to do the same for all the past and future PUFs. Therefore, the uniqueness of human biometrics is based on statistical estimations, rather than being truly proven. The same holds true for the uniqueness of PUFs as already discussed. This is an important point, as it leads us to assume that

accurate, manner, as we discuss in Section 1.2. Additionally, even simple concepts and objects usually cannot be defined in a truly exact manner, e.g., according to most definitions of a triangle, a line segment containing three points can be considered a "degenerate" triangle. Moreover, the mere existence of mathematical singularities can be considered as a good indicator of the limitations of the human definitions even for rather theoretical concepts, let alone for real-life objects. Regarding PUFs in particular, their uniqueness naturally degenerates when both the number of their CRPs and the size of their responses diminish. Therefore, we cannot demand that all PUFs realise genuinely unique functions, per PUF instance, but rather only that each PUF realises a rather unique, i.e., highly distinguishable, function. Ideally, however, each PUF instance would realise a unique function and its behaviour would never change. Nevertheless, this is not really true for real-world PUFs. Therefore, we are obliged to provide a somewhat imprecise definition, in order to accurately define real-life objects that are quite often not ideal.

[3] We note here the existence of well-known human features that are not unique enough to be considered as adequate biometrics, e.g., height, weight, etc., in the same way that features of well-known PUFs are also not unique enough to be considered for use, e.g., the number of Static Random Access Memory (SRAM) cells in an SRAM PUF. Nevertheless, it is evident that these features do play a role in the applications of both biometrics and PUFs. We also note that biometrics of low entropy do exist, e.g., a partial fingerprint, or a partial DeoxyriboNucleic Acid (DNA) sample, in the same way that PUFs of low entropy also exist, e.g., an SRAM PUF consisting of only ten SRAM cells. Such cases rather stretch the term "unique", which refers to the general case of either a biometric or a PUF. As we have already discussed, determining the limits of the term "unique" in the case of PUFs is as difficult as doing so in the case of biometrics, as not all instances of PUFs or biometrics, or even a large enough number of them, can ever be available. For example, it is rather impossible to predict with full certainty that a future SRAM PUF will not have the same response as a current one, but we can, with high probability, assume that this will not be the case, in the same way that it cannot be proven with certainty that Plato and the reader do not have the same fingerprints, but it can be claimed that there is a high probability that this is not the case. Finally, both biometrics and PUFs require some form of error correction in order to be used efficiently in practice, as, e.g., fingerprints may be somewhat blurred, in the same way that SRAM PUF responses may be a little noisy, e.g., when they contain the values of unstable cells.

PUFs should be considered as adequate security mechanisms for applications similar to those for which human biometrics are employed (e.g., identification and authentication), as long as they are as secure (e.g., as unique and robust) as the human biometrics employed in such applications.

In particular, it should be noted here that two factors are critical in order to assess the security that human biometrics or PUFs can provide: the uniqueness (per instance of them) of the characteristics utilised as biometrics, or for the implementation of a PUF, and the difficulty of successfully predicting these characteristics for a particular instance of human biometrics or of a PUF. In general, it is assumed that for any PUF CRP, the relevant challenge is public information and, therefore, the secret that an attacker would need to guess successfully is the relevant response. Since the response of a PUF consists of a number of bits, the relevant number of different responses that a PUF may produce is at most 2^N, assuming that each response of such a PUF consists of N bits. Therefore, the probability of an attacker predicting the relevant PUF response correctly is $\frac{1}{2^N}$, which means that, even if each PUF response consists of only 1 bit, the attacker cannot gain an advantage over an oracle that predicts randomly. Thus, it should be evident that, if PUF CRPs are of full entropy and are not correlated, then a PUF that produces either responses of a (relatively) high N or a (relatively) high number of responses, can be considered as secure.

Therefore, the security of PUFs lies not in their (most often fictional) ability to be tamper-proof or unclonable, but rather on their inherent characteristics that define their entropy and the correlation between the bits of their responses. Hence, research should focus on PUFs of (very) high entropy and (very) low correlation, or address these issues, if they exist, rather than attempt to provide additional security features on inherently insecure primitives, through intricate schemes and extremely complex structures.

Furthermore, as there is an innate relation among the cost of production, the cost of attacks, and the cost of potential (security) countermeasures, in conjunction with the expected damages and the risk of incurring such damages, one would need to first investigate the applications that are relevant to PUFs, before one could decide which attack scenarios (and, thus, also which countermeasures) may be appropriate to examine for the relevant use case scenarios. In general, it should be evident that one would usually not apply an expensive security solution to a use case in which the potential damages are considered to be of low cost, or there is a low risk of a successful attack. In the same way, a security solution should not be far more expensive than the original scheme (e.g., device) to which it is applied, or it will most probably not be adopted.

Therefore, the general suitability of PUFs as practical security mechanisms is also dependent on their cost, as well as the cost of attacks that may be expected, the level of risk, and the amount of damages that such attacks may cause. This principle holds true also for biometrics, e.g., fingerprints may be cloned and are not extremely difficult to acquire, yet the cost of a fingerprint-based authentication scheme is low enough to allow for them to be used to authenticate users on commercial devices, such as laptops or cell phones. In this case, the risk of a successful attack is rather high, but the potential damages can be considered to be relatively insignificant, especially in comparison to the security provided by the usage of fingerprints as a biometric, in this particular use case scenario of authentication. Thus, we have observed, in our daily lives, the widespread adoption of a rather insecure biometric for a security application in devices that are extensively used every day worldwide.

Based on this observation, this work investigates practical attacks against PUFs, relevant to use case scenarios appropriate for them, in order to examine whether PUFs can provide an *acceptable* level of *practical* security, as in general, much like biometrics (and any other security solution), PUFs do not truly need to be perfectly secure to be adopted widely as a security mechanism, but rather need to only be *acceptably* secure [4] and practical, in relation to their cost, the attacks against which they can protect, the risk of such attacks being successful, and the potential damages that they may cause. This may be considered as a valuable contribution of this work to the relevant scientific field, as it means that this work is in itself rather practical, and not only a theoretical treatise of PUFs as ideal objects, which, however, do not correspond to the real-world implementations of PUFs.

Revisiting the definition of PUFs from the perspective of their nature, one could consider the definition of a very common material, e.g., glass, and its characteristics. While glass is a material that most, if not all, of us see every day around us, coming up with a concise definition for it is not as easy as it may appear. In particular, glass may be defined as an amorphous, i.e., non-crystalline, solid material, which is produced by cooling rapidly and, thereby, solidifying from a molten state without crystallising. Nevertheless, this definition not only suffers from a number of shortcomings, but it also lacks one of the most important properties of the glass that we come across in our daily lives; it misses the fact that most of the material that we consider as glass is transparent or, at least, semi-transparent (e.g., translucent). On the other hand, there is, of course, glass that appears rather opaque, or which may act as a mirror on either of its sides. Following this line of thought, the question arises of whether such glass, which is non-transparent and is rather reflective on one or all of its sides (which may be typically installed in the windows of skyscrapers), should also be considered as glass; a further question arises regarding common glass mirrors, especially since a mirror consists not only of what we would normally think of as glass, but also of some reflective material, such as a layer of metal. Similar issues may arise regarding the uniqueness of PUFs, i.e., since the uniqueness of CRPs per PUF instance is a rather defining characteristic of PUFs, there ought to be fringe cases of this property. For example, a PUF whose response consists of only 2 bits and which has only 4 CRPs [5] cannot be considered as one that exhibits genuine uniqueness. Nevertheless, much like reflective glass panels and other non-translucent glass, this is rather an edge case.

Another important similarity between a potential definition of glass and the definition of PUFs has to do with the potential security (and privacy) properties of the relevant objects. For example, while it is common knowledge that bullet-proof glass exists, it would not appear rational to define glass in general as bullet-proof, as this is not a common property of all instances of glass, while at the same time also not being a defining property of glass. It would, therefore, not appear to be rational to consider glass being bullet-proof as an ideal material, apart from specific applications, for which this property would be

[4] The concept of *acceptable* security, as well as the relevant notion of *acceptable risk*, are rather present in our daily lives, as email passwords are required to be changed after a certain number of months by most providers, and our credit and other bank cards every couple of years, in order to be considered as secure, even though, in most cases, the relevant secrets have not been compromised. Fortunately, digital and other inanimate security primitives can be easily replaced, if they cannot be adequately modified, in order to (continue to) provide an *acceptable* level of security, unlike the human biometrics that (have to) rather remain, more or less, the same for a(n actual) lifetime, notwithstanding whether their secrecy has been compromised or not.

[5] A PUF with responses of two bits could indeed have more than four CRPs, as long as it can be challenged by more than four different challenges; in this case, only its potential responses to any challenge are limited to four different cases, but not the overall CRPs.

desirable. The same holds true for PUFs being tamper-proof, which is a property that is not shared by all PUFs, and which may be desirable in some applications, but which one cannot truly expect PUFs to have in order to be considered as suitable security mechanisms. Furthermore, even bullet-proof glass may be able to resist being broken by a bullet, but will most probably not be able to resist all types of bullets, or a huge amount of bullets being shot at it. In the same way, even a tamper-proof, tamper-resistant, or tamper-evident PUF may not behave in the same way against different forms of tampering. For example, a PUF may be resistant to (normal-size) probing, but not to microprobing, or its exposure to heat may be evident, while its exposure to a voltage higher than the nominal may not be as evident, or vice versa. For instance, if the PUF object is a microcrystal, the effects of thermal annealing may be evident on it, while its exposure to significant voltage may have no visible effects on it, or the opposite. Moreover, while opaque glass may provide privacy, no rational person would define all instances of glass as opaque, or even expect them to ideally be opaque, as one of the most sought-after properties of what we commonly consider as glass is exactly the opposite property of most glass being most often transparent. Therefore, we can also conclude that the inclusion of non-defining properties in a definition is highly problematic.

In a similar fashion, while most of the glass used in our windows is not bullet-proof, common glass is an almost essential material for most buildings as it allows the sun-light to come inside, while at the same time protecting the interior from wind and rain, in an extremely cost-efficient manner. In other words, it offers *acceptable* protection in terms of its cost, its potential risk of being damaged and the potential amount of damages. One of the goals of this work, therefore, is to investigate if PUFs can also provide *acceptable* security, when we consider the equilibrium among production costs, potential attacks, and damages. Nevertheless, similarly to the cases of common glass and bullet-proof glass, this equilibrium may differ depending on the actual use case scenario. In no case, however, could we define glass to be bullet-proof in general, as this is simply not generally true, and would just constitute a failed attempt to achieve *security by definition*. On the contrary, most of us seem to accept in our daily lives that glass is rather hard, but it is not unbreakable; it is brittle. Indeed, the use of glass is extremely widespread. Therefore, even if most PUFs are not tamper-proof or truly unclonable, they may still provide an *acceptable* level of security; high enough to justify their widespread usage as a security mechanism.

Moreover, there may exist extremely simple ways of improving the level of security that PUFs can provide, in ways similar to those in which the thermal isolation provided by glass windows can be significantly increased, without any notable shortcomings, e.g., by the simple usage of two glass window frames, one after the other, where normally only one window frame would exist (a common feature in Eastern Europe), or by utilising completely removable “storm windows” or, more recently, “insulating glass” panels, which are made of two, or more, distinct layers of glass with a vacuum existing between them [6]. It is a goal of this work to indeed investigate whether some form of “additive” security can be achieved through the utilisation of multiple PUFs at the same time or through the combination of PUFs with other security primitives [7].

[6] This scheme is known as double, triple, etc. glazing, or as double-paned, triple-paned, etc. windows.

[7] This concept is known as a *reconfigurable* PUF (rPUF), in which a PUF is combined with a mechanism, known as a Reconfiguration Module (RM), that transforms the original PUF response into a different bit-string, in an unpredictable and uncontrollable way for an external observer who knows only the CRPs of the original PUF [33]. In this way, the entropy of the rPUF is increased in comparison to the entropy of the standalone PUF as the RM serves as an additional source of entropy, which is used each time to “reconfigure” the response of the standalone PUF.

Furthermore, revisiting the original definition of glass as an amorphous, non-crystalline, material, we should note that such a definition would imply that such materials as rock crystal (which is the transparent, quite often birefringent, form of the quartz mineral) as well as any form of "microcrystalline glass" would not typically be considered as glass, based on that definition. Therefore, the problem of providing a concise definition, which is sufficiently general and, yet, also adequately specific, should be evident. This issue becomes even more evident and relevant by the fact that the first well-known PUF proposed, in 2001 by Pappu [34], was indeed composed of a transparent epoxy material that contained a small number of randomly distributed reflective glass spheres. Thus, we need to note that not all PUFs consist of optical materials, but rather any (inanimate) physical object could be utilised to implement a PUF as long as its characteristics exhibit a behaviour that resembles a rather distinguishable ("unique") mathematical function. Moreover, such a definition would be in agreement with the first definitions of a PUF as a Physical One-Way Function (POWF) [34, 35] and a physical random function [36, 37].

Nevertheless, the inherent problem with the definition of PUFs as physical one-way functions [34, 35] was that this definition did not just define PUFs as physical objects implementing one-way functions, but rather went some steps further and demanded that the simulation of such functions be computationally demanding and the physical objects difficult to clone. As already stated, this could be considered as an attempt to ensure security by definition and, therefore, the definition could not be considered as truly valid, especially in consideration of novel attacks that proved that the simulation and cloning of PUFs were indeed feasible. Additionally, the existence of "genuine" one-way functions is an unsolved problem in computer science, in the field of computational complexity and polynomial problems. In general, while we know that for some functions that can be computed in polynomial time [8], it appears to be hard to calculate their inverse in polynomial time, it has not been proven that such an inversion of any of these functions (based on the image (i.e., the output) of such a function for a random input) is indeed not feasible in polynomial time, in all cases. In the case of PUFs, this would mean that for a response to a random challenge, which can be calculated in polynomial time, it should be hard to identify the relevant challenge within polynomial time, by "inverting" the function implemented by the relevant PUF [9]. Again, this may appear to be true in the general case for PUFs, but has not (yet) been proven in a definitive manner. Even worse, due to the implications of such a proof to the general complexity theory, the existence of one-way functions may at some point be proven to be unprovable, in general.

Physical random functions were defined in some works [36, 37, 38] as functions embodied by a physical device that are easy to evaluate and hard to characterise, and in other subsequent works [38, 39, 40] as random functions that can only be evaluated with the help of a particular physical system. These definitions are very similar to the one proposed in this work, with some important differences. Firstly, the requirement that a physical random function be hard to characterise alludes to its perceived unclonability, and, as we have already stated, this notional unclonability has rather been disproven for most of the well-known PUFs. Secondly, in our definition a PUF [10] is itself a physical object that essentially realises a (rather unique) function, instead of a (random) function that is evaluated only through

[8] "Polynomial time" refers to a time complexity that is upper bounded by a polynomial expression.

[9] It should be noted that one-wayness cannot on its own guarantee the security of a PUF. If an attacker has access to the PUF, he/she can obviously generate any and, potentially, all CRPs of that PUF, even if such a PUF implements a one-way function.

[10] The term "Physical Unclonable Function (PUF)" was actually first suggested in the works regarding physical random functions and was used as a synonym for a physical random function [36, 37, 38, 39, 40].

physical means. This difference is important, because our definition demands a PUF to be a physical object instead of a mathematical function, while a physical random function can be a mathematical function which is "evaluated" only through a physical system; the latter definition may include also mathematical functions that are not embedded in physical objects, but are evaluated only through physical means. Finally, the term "random" (much like the term "unique" in our own definition) is rather hard to define in a precise manner and could indicate randomness without the essential uniqueness that PUFs should exhibit, i.e., uniqueness implies also an inherent complexity [11], apart from the required randomness, in order to achieve the distinguishability that leads to the notion of uniqueness. This complexity is also implied by the notion of one-wayness, as it is the defining factor that would disallow the inversion of a one-way function within polynomial time for every random set of an image (i.e., an output) and an input.

Additionally, it has been observed that PUFs whose responses are always of the same constant size, in a way indeterminate of the size of their challenge [12], appear to exhibit an avalanche effect, with small changes in the physical object itself (or to the challenges used, i.e., in the measurement process) causing large changes to the values of the relevant responses [34]. This observation, however, seems to hold true for most, if not all, well-known PUFs. Furthermore, Pappu noted that PUFs whose responses are always of the same constant size, in a way indeterminate of the size of their challenge, appear to produce responses that are "collision-resistant" [34]. This also holds true for most PUFs whose responses are of an adequate size, due to their inherent complexity and the complexity of the relevant measurement process. However, while resistance to collisions would indicate that two challenges to the same PUF cannot result (very often) in the same response, which is not the case for some types of well-known PUFs, such as delay-based PUFs, the requirement that PUF CRPs be rather unique per PUF instance indicates only that two instances of a PUF type should not have exactly the same CRPs, while some of the responses of a particular PUF instance to different challenges can be the same (and, therefore, not all PUFs exhibit collision resistance), and some (but, preferably, not all) of the responses of different PUF instances to the same challenges can also be the same. Ideally, however, the response of any PUF instance to any challenge would be different from the response of any other PUF instance to the same challenge, as Figure 1.1 shows. As already explained, this may not always be feasible, due to the size of the responses that some PUFs provide.

Moreover, the avalanche effect noted by Pappu is particularly significant as an indicator of the true nature of PUFs, which are rather based on the results of a *chaotic* process. As the manufacturing process includes minor variations, most often on the microscopic scale or at the nanoscale level, and these variations tend to be dependent upon a large number of different factors, the manufacturing process is indeed a process that follows the chaos theory, as it is a dynamical system whose apparently random state is actually governed by deterministic laws, but is extremely sensitive to its initial conditions, which, in this case, consist of both internal and external factors. One may consider the case of the "paper PUF" [41, 42, 43, 44, 45, 46, 47], whose responses are based upon the alignment of the fibers of a paper sheet. In this case, the challenge is the way light (be it regular sun-light, laser light, etc.) hits upon the inter-twisted (wood) fibers of the paper sheet, while the response is based upon the way the fibers of the

[11] The requirement for complexity in the context of PUFs is indeed noted in the works that concern physical random functions [36, 37, 38, 39, 40], but is not truly discussed or defined.

[12] These PUFs were defined by Pappu in 2001 as Physical One-Way Hash Functions (POWHFs) [34].

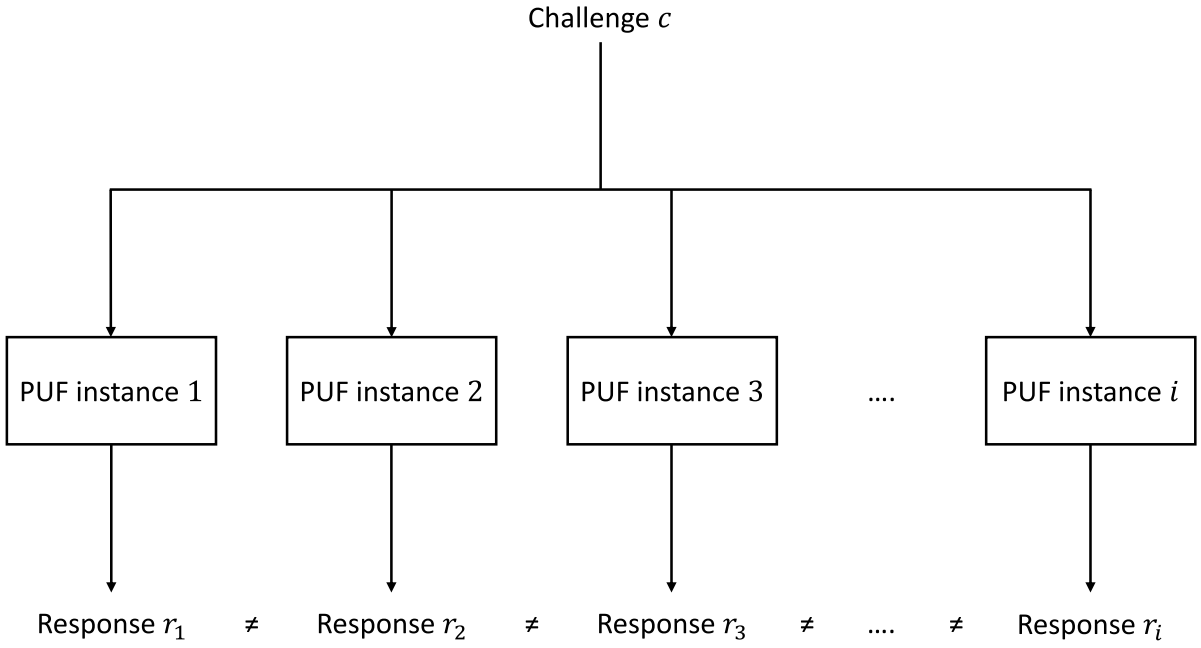


Figure 1.1: Principle of operation of an ideal PUF. For any challenge c, the responses of all PUF instances must differ. This is obviously not true for PUFs that produce responses of a (relatively) small size. In this case, a set of multiple CRPs may be used in order to distinguish such PUFs, if an adequate number of CRPs exist.

paper affect the light and, in this way, produce a specific (speckle) pattern (which may be captured as an image, etc.). In this case, during the manufacturing process, the fibers of the paper are inter-twisted in a rather random manner. However, the way in which these fibers are aligned could also be considered as deterministic, but extremely dependent on the initial conditions of the pulp mixture and the relevant machinery, i.e., the initial position of each fiber in the pulp and its movement, the potential presence and movement of mixers, stirrers, etc. Additionally, the way in which fibers are separated from the pulp in order to form what will then become paper sheets, is also important, but can again be seen as either a rather random process, or as an actually deterministic one that is extremely dependent on the initial conditions of the relevant physical system, which is comprised by the pulp mixture and the machinery extracting a selection of the fibers to form paper sheets. Therefore, the production of the paper PUF can essentially be considered as a chaotic process, although it initially appears to be rather random. We can also note that PUFs are indeed physical "random" functions, in the sense of physical objects realising "random" functions, the randomness of which, however, is the product of a complex chaotic production (or, even, generation, or measurement) process that leads to the inherent (relative) uniqueness that is essential to the (usually, complex) functions realised by PUFs.

In general, while the physical object produced by such a chaotic process (in this case, a paper sheet) may appear to be a rather static physical system, this is not truly the case, even considering the object on its own. Over an adequately long time, the fibers of the paper will "age", i.e., they may oxidise and turn from white to yellow, they may start to crumble or loosen, etc. Additionally, external factors, such as exposure to sun-light, to heat, or to other adverse conditions (dirt, etc.), can also affect the physical object. Furthermore, also the measurement of such a system is a rather chaotic process, as, e.g., in the case of a paper PUF, the angle of the light, its intensity as well as the setup used to capture the response

can have dramatic effects on the values of the response measured, exactly due to the microscopic scale of the characteristics that need to be evaluated, which affect the degree of precision being required in order to provide a particular challenge to the PUF and get the response expected. Nevertheless, we note that this is yet another similarity of PUFs to human biometrics, such as the human fingerprints, the iris of the human eye, etc.

Thus, similarly to human biometrics, the measurement of PUFs also requires a stage of error correction, in order to produce a result that can be considered as accurate and robust. It should be noted that this is yet another indication that PUFs are complex systems, whose behaviour changes over time, in a way similar to how human biometrics also age. Therefore, PUFs should be studied through models of relevant complex dynamic systems, rather than in a theoretical manner based on the notion that their behaviour remains unchanged even over long periods of time. In general, from a philosophical point of view, PUFs abide by the observations of the Ionian Greek philosopher Heraclitus regarding the impermanence of physical objects, i.e., the fact that physical objects change with the flow of time. We can, therefore, conclude that PUFs should be considered as physical objects (which change with the flow of time) rather than as mathematical functions (which are not affected by time).

1.3 PUFs as Security Mechanisms in the Context of the Internet of Things (IoT)

As technology keeps constantly advancing, electronic devices are becoming more and more autonomous. Additionally, an ever-increasing number of devices get connected with each other over the Internet. These two facts have led to the conception of an interconnected network of devices where data are not exchanged between humans, but between sensors, actuators and other electronics that are incorporated into or connected to these devices. Such a network of "things" is referred to as the *Internet of Things* (IoT) and is used for the implementation of such concepts as smart homes, smart cities, and smart vehicles.

In this context, different devices can communicate and collaborate with each other, in order to decide upon which actions to perform (i.e., how to operate), based on a set of predefined rules and data stemming either from their own sensors or from the sensors of other devices or from other electronics connected to the IoT. The concept of the IoT is not limited only to commercial and scientific applications, but has also been utilised in the industry within the framework of the fourth industrial revolution (Industry 4.0), where technologies are fused in industrial environments in the context of *Cyber-Physical Systems* (CPSs) and *Industrial IoT* (IIoT).

However, in order to keep their costs minimal, IoT devices are usually based on low-cost designs, which tend to be resource-constrained. Therefore, they most often do not incorporate dedicated security solutions, such as Trusted Platform Modules (TPMs) or other security (co-)processors [13]. It should be stressed that, in general, the addition of any hardware component, including those that may be required in order to secure a system, comes at an increased cost, which could, therefore, prove detrimental to the adoption of IoT devices. Thus, as most such devices do include one or more memory components, memory-based PUFs can be easily implemented in these devices, in order to serve as practical and cost-efficient security solutions for them, as most security architectures require some kind of security anchor,

[13] Most interestingly, the forthcoming Windows 11 Operating System (OS) currently requires a TPM version 2.0 to be enabled in order for this OS to be installed, which means that it may not be able to be installed in a wide range of devices.

such as a PUF, a TPM, or other hardware or software components or schemes that can be utilised, e.g., for secure key storage.

Additionally, we need to note that as technology keeps evolving, the notion of what the IoT is, has also been slowly changing into a more comprehensive definition that encompasses more and more contexts and scientific fields. IoT use cases are no longer limited to sensors for dedicated uses that transmit data to some remote aggregator, which may process them and potentially decide upon some action to be performed. Instead, IoT usage has currently become almost ubiquitous, as IoT devices find application in such fields as agriculture, transportation and vehicles, healthcare, pharmaceuticals and medical devices, industrial and maritime applications, or even in space applications and within such pervasive concepts as smart cities.

Moreover, autonomous vehicles can also be considered as a segment of IoT devices, as information, which may stem from sensors of a single vehicle or even of other vehicles, flows between different devices, leading to actuators in different vehicles, or even in the surrounding infrastructure, being activated. Currently, the concept of the IoT has even acquired a space segment, as devices in space can also exchange information that leads into actions being taken without human involvement or presence.

The ubiquitous nature of the IoT has led to an increasing use of a novel systems architecture paradigm, in which systems no longer follow the classical von Neumann architecture. In the novel architecture paradigm, individual computer systems do follow the von Neumann architecture, but at the same time form parts of a larger system, where sensors play the role of input devices and actuators the role of output devices, while the most significant part of the computation may take place at (potentially remote) aggregating devices, to which the information collected from the sensors is transmitted to be processed and which then transmit commands to appropriately control the relevant actuators. In this case, the buses of the overall system are the wired and/or wireless channels between the different physical devices, which are, therefore, rather exposed to attackers. Additionally, each such system of sensors, computer devices, actuators, and channels, most often, forms just a part of a larger system that constitutes a *system of systems*. Furthermore, a new generation of the IoT, i.e., IoT 2.0, is emerging, taking advantage of the currently evolving technologies of artificial intelligence, the blockchain, and machine learning. This new generation of the IoT is also deeply based on the constant consolidation of pre-existing systems and sub-systems, of a widely heterogeneous nature, into larger systems, which leads to the formation of systems of systems.

For example, a smart traffic management system may utilise satellite, road infrastructure, drone and other vehicular systems as its “sensors”, in order to detect the current state of traffic on a highway. The relevant data may be transmitted wirelessly or through wires to remote server and cloud systems for processing through the utilisation of machine learning and other relevant computer intelligence algorithms. Finally, a set of instructions may be transmitted to relevant “smart” actuator systems, such as the road signal infrastructure and “smart” vehicles, such as autonomous cars. The most safety-critical instructions may be stored in a dedicated blockchain, in order to prevent future repudiation, in case of an accident. In this case, each system mentioned, forms only a small part of the overall smart traffic management system, while in turn being composed of several even smaller sub-systems. For example, a “smart” car is composed of a number of Electronic Control Units (ECUs), each of which controls and performs different functions of the car. For example, one ECU may be connected to lidar, radar and other optical acquisition

sensors and it may control the brakes and the steering of the car, while another ECU may be connected to temperature and humidity sensors and it may control the air conditioning system. In the aforementioned example, an autonomous car may act as a sensor for the overall smart traffic system, if one of its sensors detects an accident further on the road or traffic congestion, and the relevant information is transmitted to the road infrastructure. In the same example, however, another autonomous car may act as an actuator for the overall traffic management system, by starting to reduce its speed after having received the information about the accident or the traffic congestion. In both cases, each car's relevant ECU will have processed the data locally, while the overall system's computation will have taken place in remote servers, potentially on the cloud, in order for the smart traffic management system to assess the information received regarding the need for reduced speed and to decide which vehicles should be notified. In this case, it is not only evident that the whole system will take advantage of the IoT, and even of potential IoT 2.0 capabilities, but also that the overall system is indeed based upon a hierarchical multilayered system-of-systems architecture.

In this framework, the security architecture also needs to follow a multilayered approach, where both individual components and the systems that these components constitute need to be secured, as well as the communication among different components, and among different (sub)systems. For example, in a smart traffic management system, where a drone's detection of traffic congestion has led to an autonomous car reducing its speed, the drone, the road infrastructure, and the autonomous car must be secure systems, and their communication must also be secure, in order for traffic management to be conducted in a secure, safe and trusted way. However, this also means that the subsystems of the drone and their communication, the subsystems of the road infrastructure and their communication, and the subsystems of the autonomous car and their communication must also be secure. Therefore, as the security of each system is dependent on the security of its subsystems and the security of their communication, it is evident that the security architecture needs to indeed follow a multilayered approach.

In general, the pervasive nature of the IoT entails the utilisation of a wide variety of devices and communication protocols, in order to address the largely diverse use cases and applications in which the IoT is employed. Additionally, even for a single use case, a large number of heterogeneous systems are utilised in a collaborative manner, in order to provide the optimal result. Therefore, security solutions for the IoT need to be applicable to both high-end devices, such as infrastructure and cloud servers, which may have significant capabilities, and low-end resource-constrained devices and electronics, such as simple sensors. Moreover, the widespread adoption of the IoT has been based on its relatively low cost, which is based upon the economies of scale developed through the mass production of most of its components. It must be noted that the IoT was initially consisting almost exclusively of relatively low-end devices, as well as primitive sensors and actuators, utilising the remarkably low cost of production of modern commercial electronics. Therefore, within this context, *intrinsic* PUFs constitute extremely suitable security mechanisms for the IoT, as such PUFs are realised on inherent components of devices and systems, and, therefore, do not require the addition of hardware for their construction or operation, which keeps the costs associated with their production and operation minimal.

Memory-based PUFs, in particular, not only constitute low-cost security solutions, as memories are most often inherent components of modern computer systems, but can also be lightweight, practical and flexible in their role as security mechanisms for the IoT. More specifically, as most memory-based

PUFs are intrinsic, and the software required for their operation as well as the relevant data created and processed are of a rather small size, such PUFs can be considered as extremely lightweight. Additionally, some memory-based PUFs, such as the Dynamic Random Access Memory (DRAM) retention-based PUFs, Row Hammer PUFs, DRAM access latency-based PUFs, and Flash PUFs, can be accessed at run-time. Run-time access makes these PUFs practical, as, otherwise, a (device) reboot may have been required. Furthermore, it has been noted that DRAM retention-based PUFs can potentially provide an exponential amount of CRPs [48]. In general, DRAM retention-based PUFs and Row Hammer PUFs can generate multiple CRPs and, therefore, also multiple different keys. On the contrary, other memory-based PUFs, such as the Static Random Access Memory (SRAM) and the DRAM PUFs that are based on the start-up values of the SRAM and the DRAM memory modules, respectively, can only provide a single CRP. Moreover, contemporary memories tend to have a large size, which allows for the usage of different memory segments of varying size as PUFs. This enables the creation of multiple keys of different lengths from the same memory-based PUF instance. The ability of a memory-based PUF to generate multiple CRPs and, therefore, also multiple keys, usually of varying length, allows it to be used in a highly flexible manner and makes it suitable for a number of different applications.

Finally, we also need to note that, in order to keep costs at a minimum, while offering the maximum level of security possible, PUFs can only be implemented in devices that are part of the two lowermost layers of the Internet of Things Reference Model (as shown in Figure 1.2) [14]. However, previous works have only proposed the use of PUFs in the first layer of the Internet of Things Reference Model, i.e., the existing literature has so far only considered PUFs in the "Things" layer that consists of edge IoT devices, sensors and actuators. Nevertheless, the use of PUFs at this layer does not always scale well, as it requires helper data and a cryptographic token to be stored, for each communication pair, on at least one of the two devices involved. To overcome this problem, our work suggests employing PUFs also, or potentially even exclusively, at the second layer of the reference model, where network connectivity functions and communication devices, including routers and switches, reside. In this way, the PUF functionality potentially needs to be instantiated only once for each single device to establish communication with all the other devices connected to the same network. More specifically, in this case,

[14] The Internet of Things Reference Model was proposed by Cisco in 2014 [49], and demonstrates the seven layers in which an IoT system can be divided. In particular, the lowermost layer (layer 1) concerns the physical edge devices, such as sensors and actuators, and other devices that collect data and perform actions. Layer 2 concerns the connectivity devices, such as routers and other network modules that provide communication between the devices of layer 1 and the computing devices of layer 3, which analyse the data collected from layer 1 and prepare them for storage and usage by higher layers, as well as send out commands to the actuators of level 1, based on the analysed data. Layer 4 deals with the storage of the received data, i.e., the accumulation of data sets for further use, the creation and management of databases, etc. Layer 5 allows the utilisation of the data stored in layer 4 by higher layers, through their aggregation and conversion into a single format suitable to be used by applications and other software that form layer 6, which concerns the utilisation of the data collected by the edge devices in applications that may serve reporting, analytics, control, or a plethora of other, purposes. Finally, the topmost layer (layer 7) deals with the interaction between the IoT and conventional systems and networks, humans, and businesses. In general, the Internet of Things Reference Model views the IoT not as an isolated network of devices that would consist of the three lowermost layers of the Reference Model, but as a network of devices that is connected to database systems, allows for the implementation of applications and services based on the data gathered and exchanged among its lowermost layers, and aims to provide individuals, businesses, and conventional systems, with enhanced information and services. In this work, we examine how PUFs, i.e., hardware-based security primitives, that are inherent components of devices in the two lowermost layers of the Internet of Things Reference Model, could serve as security anchors that may potentially be used to secure every layer of this Reference Model, and especially the three lowermost layers that form the core of the IoT, in a lightweight and practical manner.

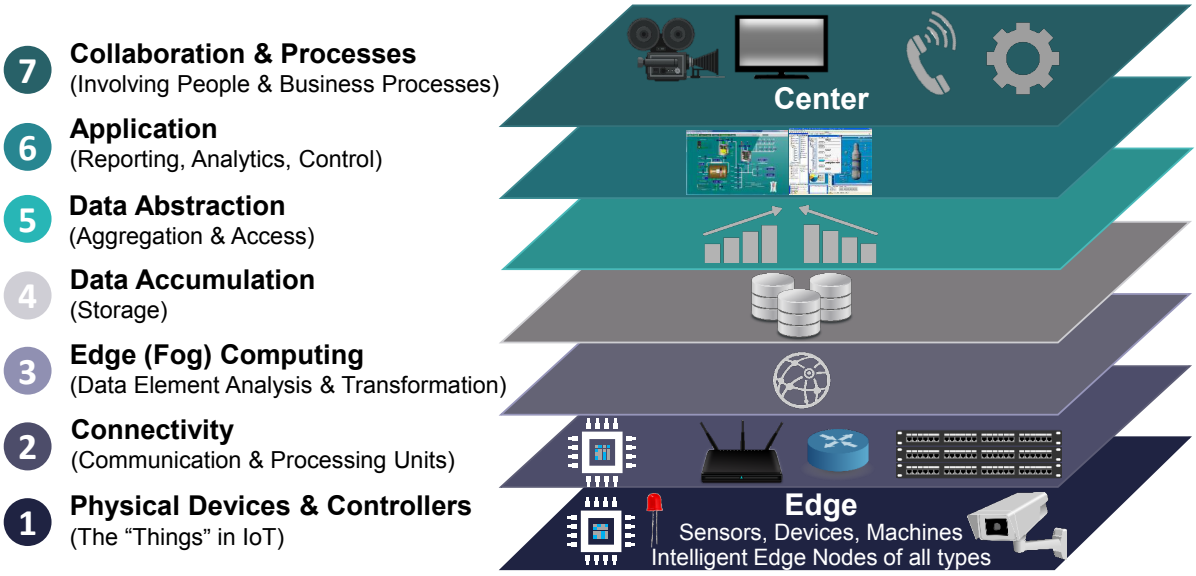


Figure 1.2: The Internet of Things Reference Model (as modified from [49]).

all information exchanged in the network controlled by a particular network device, e.g., a router, will be authenticated and encrypted using the secret of that router's PUF. In this way, each network node will only need to have knowledge of that particular secret in order to communicate with the router itself and any other device on that network, while in the usual case, it would need to have knowledge of the secrets of all the network devices it would like to be able to communicate in a secure manner with.

A brief discussion of previous works relevant to the role of PUFs as security mechanisms for the IoT is provided in Section 2.2, where a number of such recent works are examined in some detail and their potential shortcomings are considered, in comparison to the contributions made by this work.

1.4 Scope of This Work

Due to the pervasive and ubiquitous nature of the IoT and the diversity of PUF implementations proposed in the literature, it is imperative to define and clarify the scope of this work, in order to subsequently discuss the challenges that it addresses, as well as its contributions and impact. In this work, we focus on intrinsic memory-based PUFs and their potential to serve as adequate security mechanisms in the context of the IoT, as such PUFs can provide security in a practical and lightweight manner, and some of them (SRAM PUFs) are already available for commercial applications [50]. In general, exactly because of the diversity of the IoT devices and networks and the extremely pervasive nature of the IoT, the mechanisms, schemes, and protocols employed to secure the IoT, need to provide a practically acceptable level of security both under nominal conditions and under environmental variations, and in the framework of both heterogeneous devices and diverse networks. We, therefore, examine the robustness of memory-based PUFs to variations of their ambient environment as well as their potential for securing different IoT devices and networks, in order to investigate whether such PUFs can truly serve as practical security mechanisms for the diverse application areas of the IoT. More specifically, this work does not aim to prove that such PUFs, or any PUFs for that matter, can provide perfect security, but rather that the security that such PUFs provide is of a practically acceptable level, as explained before. Thus, in order to be considered as acceptable security solutions, the examined PUFs do not need to be truly immune to

quite intricate attacks, but should rather be robust to environmental changes and other factors that can serve as fairly simple attack vectors.

Furthermore, we also acknowledge that not all PUFs provide on their own a particularly high level of security. In particular, the security of memory-based PUFs is highly dependent on the secrecy of their CRPs. Nevertheless, intrinsic memory-based PUFs may provide an acceptable level of security for IoT devices, especially low-end resource-constrained devices that have no other inherent hardware security primitives. In this case, in order to assess whether such PUF-based security solutions are practical, we need to take into account their lightweight and cost-efficient nature and, thus, test them against simple attacks of a similar level of cost, e.g., against attacks based on environmental variations. In general, extremely lightweight security primitives, such as intrinsic memory-based PUFs, may be too limited to be able to provide an exceptionally high level of security on their own and, therefore, such PUFs may not always be suitable to secure applications that are extraordinarily security-critical. Thus, it also follows that such security primitives do not, in general, have to be immune to excessively expensive, intricate, or rather peculiar attacks in order to be considered as secure.

For these reasons, and in order to determine if PUFs can provide a practical security solution, our work examines the potential role of intrinsic memory-based PUFs in the security of the IoT. In order to determine the level of security that such PUFs can provide, it investigates the effects of environmental variations on them, and proposes potential attacks based on these effects as well as efficient countermeasures. Then, it also proposes and discusses potential security schemes and protocols based on these PUFs, which can be utilised to secure both IoT devices and IoT communication networks, in order to prove the suitability of such PUFs as security mechanisms for the IoT. We also note that these PUFs can provide an enhanced level of security when they are utilised in conjunction with other hardware or software security primitives and, therefore, such composite schemes could indeed be used to adequately secure even high-end devices utilised in security-critical applications, as our work proposes.

Finally, it is important to note that all the security schemes, protocols and designs proposed concern Commercial Off-The-Shelf (COTS) electronics, rather than industrial devices, Field-Programmable Gate Arrays (FPGAs) or Application-Specific Integrated Circuits (ASICs). In this way, our work aims to examine the practical applications of PUFs in consumer electronics, rather than in devices dedicated to specific applications, in order to investigate the capability of PUFs to serve as security mechanisms in general-use devices, such as the ones broadly utilised in the IoT. Additionally, all the PUFs examined, were realised in hardware components that were not dedicated to serving as security primitives. In particular, the memory-based PUFs utilised in this work, were implemented on memories that were serving as storage components for COTS devices that can be used in the IoT. In this way, not only these PUFs are cost-efficient, but also the usage of the relevant components as PUFs can (and should) be controlled, as it affects the operation of the overall device. In general, in this way, it can also be tracked and, potentially, controlled when the PUF is used and by whom. Finally, it is important to state that, for the purposes of this work, the hardware and software security mechanisms and schemes, such as hash functions, encryption algorithms, etc., that are employed in the framework of security mechanisms, schemes, or protocols based on the PUFs examined in this work, are assumed to be inherently secure, and the examination of their security characteristics and quality falls outside the scope of this work, to the extent that their security is not dependent upon the PUFs considered in this work.

1.5 Challenges Addressed by This Work

As it is already evident, the main research question that this work addresses, concerns the role of PUFs in enhancing the security of the IoT. In order to provide an adequate assessment of PUFs as security mechanisms, it addresses a number of challenges that PUFs would need to overcome in order to be considered as an adequate security solution. These challenges are examined in the form of the following research questions that this work aims to address:

- What can be considered as a PUF?

  In order to address PUFs as security mechanisms for the IoT, it is important to first define what a PUF is, how it works, and what its essential properties are. Section 1.1 provides a definition of PUFs, introducing the concept of "Physical *Unique* Functions", while Section 1.2 elaborates upon the nature of PUFs, their operation principles, and their essential properties. A brief definition of the IoT is provided in the very beginning of Section 1, and is further discussed in Section 1.3.

- Why do PUFs appear to be suitable for enhancing the security of the IoT?

  In order to address whether PUFs are truly suitable for practical IoT applications, we must first examine what makes them appear as a potentially fitting security solution for the IoT. This is addressed in Section 1.3, where also another research question is addressed:

  "Which PUF types are the (most) suitable to serve as security mechanisms for the IoT?".

  This question is addressed by identifying and presenting the advantages of memory-based PUFs over other security mechanisms, including other PUFs types, in serving as security mechanisms for the IoT. Relevant memory-based PUF designs are examined in Section 3.1.

- Can memory-based PUFs serve as adequate and acceptable security solutions for the IoT?

  In order for memory-based PUFs to constitute practical security mechanisms for the IoT, they need to provide adequate and acceptable security. To this end, we present and discuss the most well-known designs of memory-based PUFs in Section 3.1, and then discuss the most relevant metrics that can be used to assess their quality characteristics in Section 3.2. In general, however, it is important not only to establish that memory-based PUFs can provide adequate security under both nominal and adverse environmental conditions, which is done in Section 4, but we also need to address whether memory-based PUFs can provide an acceptable level of security, when the relevant costs are considered. This is an essential property in order for a security mechanism to be adopted in practice, which, albeit, is most often not addressed in scientific works. In our work, this is done in Section 3.3. A security mechanism may provide adequate security, but, for it to be adopted in practice, it also needs to provide it at an *acceptable* cost. Additionally, a cost-efficient security mechanism still needs to provide an *acceptable* level of security, in order to be considered as a practical security solution. To this end, the concept of *acceptable* risk is introduced in order to examine the balance between security and cost.

  The ability of memory-based PUFs to provide adequate security is not only evaluated under nominal conditions, in Section 4.1, but also under ambient temperature and supply voltage variations, in Section 4.2 and Section 4.3, respectively, and in the presence of ionising radiation, in Section 4.4,

as a security mechanism needs to also function adequately in adverse conditions, in order to be considered as practical. For this purpose, the relevant quality metrics that have been introduced in Section 3.2 are utilised.

- Can memory-based PUFs be utilised to provide security in realistic applications and use case scenarios, in the context of the IoT?

  An important test in order to confirm that memory-based PUFs can provide adequate and acceptable security for the IoT, is to ensure that they can indeed function as security mechanisms in realistic applications and use case scenarios, in the context of the IoT. This is done in Section 5, where both conventional and more advanced security applications of memory-based PUFs are examined in the context of practical IoT applications.

By managing to adequately address these questions, this work aims to thoroughly address the main research question that it examines, namely:
"**Can memory-based PUFs be utilised in order to provide security for IoT devices, networks, and applications in a practical and lightweight manner?**".

1.6 Contributions and Impact of This Work

Our work considers memory-based PUFs as security mechanisms for COTS devices and their communication, in the context of the IoT. In order to address the contributions and impact of our work in a proper manner, we first need to briefly consider the state of the art before this work, which is discussed in more detail in Section 2. In particular, memory-based PUFs are commonly thought to belong to a category of PUFs known as *weak* PUFs. *Weak* PUFs have only a limited number of CRPs, and the most well-known weak PUF, the SRAM PUF, only has a single CRP. On the contrary, *strong* PUFs are supposed to have such a large number of CRPs that their complete characterisation within a limited time frame is not possible [51, 52]. This means that the function that a weak PUF realises can be revealed through its CRPs, which therefore need to remain secret in order for such a PUF to provide security, while for a strong PUF, a limited collection of its CRPs should not allow an attacker to reveal the function that it implements.

Nevertheless, it is also worth noting that, although strong PUFs are supposed to provide a higher level of security than weak PUFs, the most well-known implementations of strong PUFs are highly vulnerable to modelling and machine learning attacks [25, 26, 27, 28]. Furthermore, some works have explicitly defined strong PUFs based on the inability or difficulty of predicting their response to a randomly selected challenge, even if a large number of other CRPs of theirs are known [27, 53]. These works refer to well-known strong PUF implementations, such as the Arbiter PUF and the Ring Oscillator (RO) PUF, as strong PUF *candidates*, and prove that these implementations do not fulfill the criteria of the strong PUF definition they provide. Therefore, we note that even the existence of a (strictly defined) strong PUF has been put in doubt. Moreover, most, if not all, strong PUF implementations are not intrinsic, in contrast to the majority of weak memory-based PUF implementations.

Additionally, the distinction between "strong" and "weak" PUFs is not always obvious, as it is not easy to define what constitutes a large enough number of CRPs to prevent the complete characterisation of the relevant PUF within a limited time frame, as well as to define how limited the relevant time frame

should be. In general, however, delay-based PUF implementations have been considered as "strong" and memory-based ones as "weak". Nevertheless, the ring oscillator PUF, a delay-based PUF, has been described in some publications as a "weak" PUF [51] and in others as a "strong" one [27]. Additionally, as we discuss in other sections, DRAM PUFs can quite often provide multiple CRPs, and even a very large number of them [48]. However, in this work, we refrain from judging whether the different memory-based PUFs are "weak" or "strong", and only examine the amount of CRPs that they can potentially allow for, with the potential exception of the SRAM PUF that is, in general, considered as a "weak" PUF, as most of its implementations can only provide a single CRP. Finally, we also note that memory-based PUFs may be considered as weak, but are most often intrinsic and can, therefore, provide low-cost security solutions, in a lightweight, practical and flexible manner that has established them as competent security mechanisms for the IoT.

For this purpose, this work examines memory-based PUFs in their role as security mechanisms for the IoT and makes the following contributions:

- It redefines PUFs as physical objects that realise particular rather unique functions (with errors), rather than as actual functions. It also addresses the essential properties of a PUF, stating the requirement for relative uniqueness of the function that each PUF instance realises through its CRPs, which ideally would exhibit genuine "absolute" uniqueness. This work explains that genuine uniqueness cannot be truly guaranteed or proven for PUFs, in a similar way that such uniqueness cannot be truly guaranteed or proven for biometrics, e.g., fingerprints. Similarly to biometrics, it is impossible to prove that, in general, no two instances of PUFs will ever provide the same CRPs, in the same way that it is rather impossible to prove that no two instances of some biometrics, e.g., of fingerprints, will ever provide the same minutiae, as both PUFs and biometrics exhibit a degree of error in their measurements, and as it is truly impossible to verify the uniqueness of all their instances as well as to avoid matches of CRPs and minutiae between two different instances of PUFs and biometrics, respectively, when the set of all possible CRPs and minutiae is rather small. This work also examines the desirable properties of PUFs, and discusses why such properties as unclonability or tamper-evidence should not be included in the definition of PUFs.

- This work then considers PUFs as security mechanisms for the IoT, in terms of their suitability for such applications, and presents the advantages of using memory-based PUFs over other categories of PUFs. It presents the inherent need for security in the IoT, and the difficulty of addressing it due to the significant heterogeneity of IoT devices and networks, which include both extremely resource-constrained and high-end devices as well as highly diverse communication networks and protocols. This work, therefore, focuses on proving that memory-based PUFs constitute a feasible security solution even for extremely resource-constrained devices, which do not inherently incorporate other security mechanisms, such as TPMs, as well as that PUF-based security solutions can be utilised to secure diverse communication networks and protocols.

- This work also examines the existing literature regarding PUFs in general, as well as related works regarding the utilisation of PUFs as security mechanisms for the IoT. Furthermore, it also examines works relevant to reconfigurable PUFs, which can potentially provide advanced security solutions for the IoT. In particular, we present works related to the general concept of PUFs, demonstrating

the historical development of PUFs as a security primitive, and then examine the role of PUFs in the context of the IoT through a brief presentation of relevant works, discussing very briefly the shortcomings of such works, which this work tries to address. Finally, a short presentation of works related to the concept of reconfigurable PUFs introduces the potential advantages and shortcomings of such PUFs as security mechanisms, in order to introduce the concept of Advanced Reconfigurable PUFs (AR-PUFs) in this work, i.e., a new category of reconfigurable PUFs that allow for advanced security applications.

- Additionally, this work introduces a number of memory-based PUFs, which are based on COTS components and not on FPGA or ASIC designs, and briefly examines their individual characteristics, in the context of their potential to serve as security solutions for the IoT. For this purpose, a number of quality metrics for the responses of such PUFs are presented and examined. Moreover, this work also highlights the relation between security and cost, a relation that is highly critical for the practical applications of security solutions, but which often tends to be neglected in scientific works. To this end, the role of COTS components being used instead of FPGA or ASIC designs is essential.

- Subsequently, this work evaluates the performance of the memory-based PUFs that have been presented in it, under both nominal and adverse conditions, in order to examine their suitability as security mechanisms for the IoT. IoT devices and components find application in diverse use cases, ranging from smart transportation and smart grids to autonomous robots as well as space modules and applications. Therefore, security mechanisms for the IoT must be able to function correctly even in adverse environments. For this reason, in this work, we examine the effects of ambient temperature and supply voltage variations on the operation of memory-based PUFs, as well as their resilience to ionising radiation.

- Finally, this work proposes a number of different protocols utilising PUFs as security mechanisms for the IoT. In particular, we examine how memory-based PUFs can be used for key agreement, device identification, and authentication, their role as security mechanisms for current and next-generation IoT devices and networks, and their potential for applications in the space segment of the IoT and in other adverse environments. Moreover, this work also examines lightweight reconfigurable PUFs as security primitives that allow for advanced security applications, in the form of AR-PUFs.

The aim of this work is to examine whether PUFs can be utilised as adequate security mechanisms in the IoT. In particular, we examine whether memory-based PUFs can provide an *acceptable* level of security for IoT applications, in terms of both cost and risk mitigation, as well as under both nominal and adverse conditions. Additionally, we also investigate the security potential of such PUFs both in the usual context of key agreement, device identification, and authentication, as well as in the context of more advanced security applications and more adverse environments. In this way, we are able to clearly prove the ability of such PUFs to serve as adequate and efficient security mechanisms for the IoT in practice. Therefore, this work has the potential to change the way PUFs are perceived in the scientific community, as it demonstrates – through the use of COTS components – the ability of PUFs to provide practical lightweight security and, thus, to address the need for security in IoT devices and

networks, the use of which is already widespread and even keeps expanding. As this work also addresses the issue of extreme conditions and adverse application scenarios in the context of the IoT, it serves to clearly show beyond reasonable doubt that memory-based PUFs should be seriously considered as security mechanisms for the IoT, both in industrial as well as in commercial applications.

1.7 Brief Outline of This Work

The rest of this work is organised in the following way. Section 2 presents an overview of the works most relevant to our work. Section 3 introduces further insights into memory-based PUF designs and the quality metrics relevant to them. Section 4, then, provides an evaluation of such PUF designs under nominal conditions and a variety of adverse environmental conditions, in order to examine potential vulnerabilities of such PUFs. Relevant attacks and countermeasures are proposed and discussed in the same Section. Subsequently, Section 5 examines a number of PUF-based security schemes and analyses their potential for serving as practical security solutions for the IoT. Finally, Section 6 concludes this work, by validating its stated contributions.

2 Background and Related Work

Partially Based Upon Peer-Reviewed Content Published In

- N. A. Anagnostopoulos, S. Katzenbeisser, J. Chandy & F. Tehranipoor, "An Overview of DRAM-Based Security Primitives", Cryptography, vol. 2, iss. 2, MDPI, 2018. DOI: 10.3390/cryptography2020007
- N. A. Anagnostopoulos, T. Arul, Y. Fan, M. Kumar & S. Katzenbeisser, "AR-PUFs: Advanced Security Primitives for the Internet of Things and Cyber-Physical Systems", Proceedings of the 2019 IEEE International Conference on Consumer Electronics (ICCE 2019), IEEE, 2019. DOI: 10.1109/ICCE.2019.8661840
- N. A. Anagnostopoulos, S. Ahmad, T. Arul, D. Steinmetzer, M. Hollick & S. Katzenbeisser, "Low-Cost Security for Next-Generation IoT Networks", ACM Transactions on Internet Technology, vol. 20, iss. 3, art. 30, ACM, 2020. DOI: 10.1145/3406280

In this section, we discuss the existing literature that is relevant to this work. In particular, we first examine works on PUFs in general, we then discuss works related to the role of PUFs as security mechanisms for the IoT, and finally, we examine works relevant to reconfigurable PUFs. Through this brief presentation of works that already existed before our own, we are able to first provide background information about the historical development of PUFs as a security primitive, then examine their potential usage as a security solution for the IoT, and finally present the concept of reconfigurable PUFs that allows for potentially advanced security solutions in the context of IoT applications.

2.1 Related Works Regarding the General Concept of PUFs

Although a number of publications considered the usage of physical characteristics for identification, authentication, and anti-counterfeiting [43, 54, 55, 56, 57], the first major investigation regarding the potential of a physical structure to serve as a hardware-entangled one-way function was only conducted in 2001, by Pappu et al. [34, 35]. This study was essentially implementing and evaluating the idea for a unique identification tag found in [56] [15].

However, this study concentrated on the usage of a dedicated disordered optical structure as a security primitive, and required additional hardware for its identification. The first intrinsic PUFs were proposed and examined in 2002 by Gassend et al. [36, 37], who, in the same publications [36, 37], also proposed the term "Physical Unclonable Function (PUF)". However, again, the primitives proposed (ar-

[15] The relevant (caption) text in [56] reads: "IDENTIFICATION "TAGS" could be placed on some weapon systems to verify that they have been made at declared, monitored production facilities. One type of tag would consist of a sparkle-containing epoxy that could be applied to features of weapon systems such as solid-fuel rocket motors. When the tag is viewed under special lighting, it shows a unique reflectance pattern that cannot be duplicated; a weapon system could be "fingerprinted" by photographing this pattern. Notice the differences between the two sparkle patterns spread over similar lettering." Nevertheless, this idea was also noted in a report of the Defense Policy Panel of the House Committee on Armed Services of the United States Congress [58] that reads "Many experts have suggested that accountable systems could be painted with metal-impregnated paint to provide a unique signature for each missile, called a "tag"." Since works of the Congress of the United States of America (USA) are automatically in the public domain by U.S. law, the concept of a PUF based upon this idea may also not be covered by copyright.

biter PUFs) were dedicated hardware components that, although implemented in silicon, would require additional hardware in order to be evaluated.

A major breakthrough happened in 2007 with the introduction of the SRAM PUF [15, 59], which was the first PUF that not only was implemented in silicon, but was also based on a hardware component, the SRAM, that is part of most contemporary computer systems, and which could also be evaluated without the need for additional hardware. Nevertheless, the SRAM PUF is based on the start-up values of the SRAM, and can therefore only be evaluated at boot-time. This means that the system needs to be restarted every time its PUF response is required. Additionally, the SRAM PUF has only a single CRP, whereas all previous well-known PUF implementations could provide multiple CRPs, forcing a distinction between PUFs with a single or very few CRPs, which are referred to as "weak" PUFs, and PUFs with a large number of CRPs, which are called "strong" [51, 52] [16].

However, at the same time, modelling and machine-learning attacks against the so-called "strong" PUFs [27, 28, 60, 61], forced the development of more resistant implementations, such as the eXclusive OR (XOR) arbiter PUF and the lightweight arbiter PUF, which were, however, then attacked using more sophisticated modelling and machine-learning attacks. Therefore, while research still continues in order to find a strong PUF that is also immune to modelling and machine-learning attacks, most of the delay-based implementations, such as arbiter and ring oscillator PUFs, are proven to be extremely vulnerable to them. For this reason, the majority of the most recent PUF implementations are based on memory elements, such as SRAMs, DRAMs, and Flash memories.

Regarding the prevalence of memory-based PUF implementations in recent publications, this is clearly demonstrated in Table 2.1, which presents a brief overview of the historical development of the most well-known PUFs. In this table, we use a colour classification to distinguish between memory-based, delay-based, and other PUF implementations. A more comprehensive taxonomy of PUFs can be found in [62].

Memory elements are inherent components of most modern computer systems, therefore allowing for cost-efficient PUF implementations, especially in resource-constrained devices, such as IoT hardware [83, 84]. Additionally, DRAM PUFs based on the start-up values of the DRAM offer a significantly larger amount of entropy than SRAM PUFs because of the much larger size of DRAMs found in most modern computer systems [85]. Moreover, DRAM decay-based PUFs can also offer run-time access to their responses, addressing in this way one of the major shortcomings of the SRAM PUF [86, 87]. Furthermore, the most recent implementation of the DRAM latency-based PUF can offer a significantly lower evaluation time than the DRAM decay-based PUF [88], therefore making its evaluation much more practical. Finally, the Row Hammer PUF is an implementation that can significantly increase the entropy extracted from the decay characteristic of the DRAM [82].

Moreover, a DRAM decay-based PUF can inherently provide multiple CRPs, and not only a single one, as the retention characteristic of DRAM cells is dependent on time and temperature. In a similar fashion, the DRAM latency-based PUF can also provide multiple CRPs, when a set of different values is used for the time parameters of the read and the write operations of the DRAM. Even the DRAM PUF based on the start-up values of the DRAM can be considered to be able to provide a number of CRPs, if

[16] As already noted, the distinction between "strong" and "weak" PUFs is not always obvious, as it is not easy to define what constitutes a large enough number of CRPs. In general, however, delay-based PUF implementations have been considered as "strong" and memory-based ones as "weak".

Table 2.1: History of the Development of the Most Well-Known Physical Unclonable Functions (PUFs). The PUFs presented are colour-coded according to their type.

Year	PUF Introduced
2001–2002	Optical PUF [34, 35]
2002–2003	Arbiter PUF [36, 37]
2004	Feed-Forward Arbiter PUF [60]
2006	Coating PUF [63]
2007	SRAM PUF [15, 59], Ring Oscillator PUF [64], Latch PUF [65], XOR Arbiter PUF [64]
2008	Lightweight Arbiter PUF [66], Butterfly PUF [67], Flip-Flop PUF [68]
2010	Glitch PUF [69]
2011	Flash PUF [70], Current-based PUF [71], Bistable Ring PUF [72]
2012	Buskeeper PUF [73], DRAM Decay PUF [74, 75, 76]
2014	Bitline PUF [77], Transient Effect Ring Oscillator PUF [78]
2015	DRAM Start-Up PUF [79], DRAM Latency PUF [80]
2016	MEMS PUF [81]
2017	Row Hammer PUF [82]
	Memory-based PUFs Delay-based PUFs Other PUFs

large segments of a single DRAM are considered as different components of the PUF, each of which can be queried independently and provide its own (single) CRP.

Many of the advantages of DRAM-based PUFs are also shared with Flash-memory-based PUFs, which can also be accessed at run-time and provide multiple CRPs. Additionally, such PUFs can be constructed using COTS components [89]. Moreover, as Flash memories tend to be inherently present in most systems that utilise a non-volatile memory, such PUFs can also be intrinsic. Thus, Flash-memory-based PUFs can also be considered as lightweight, cost-efficient, flexible and practical security solutions for the IoT.

Finally, in recent years, new PUFs have been proposed based on novel non-volatile memories and other novel materials, such as Carbon Nano-Tubes (CNTs) [90, 91]. Novel non-volatile memories, such as Resistive Random Access Memories (ReRAMs or RRAMs) and Magnetoresistive Random Access Memories (MRAMs) may replace current Random Access Memories (RAMs) in the future, even volatile RAM. Therefore, relevant ReRAM [18, 92, 93, 94, 95] and MRAM [95, 96, 97, 98] PUFs should also be examined in the future, regarding their potential to serve as adequate security mechanisms for the devices they are incorporated into. Nevertheless, such structures are currently in an experimental stage and, although they have entered mass production, such memory components are currently not part of most commercial and industrial systems. Likewise, CNT technology currently seems to have rather promising applications, with CNTs potentially being used as a material for the production of transistors, or for the production of memory cells. However, CNTs have also not yet become a material that is used in the inherent components of most commercial and industrial systems.

2.2 Works Relevant to the Role of PUFs as Security Mechanisms for the Internet of Things (IoT)

In recent years, a variety of PUF implementations have been proposed for enhancing the security of IoT devices and communications, demonstrating that not only there is a need for low-cost security in the IoT field, but also that PUFs can be used to adequately address this demand. Here, we provide a brief overview of works related to this topic, noting, however, that most, if not all, consider the implementation of a PUF on each individual IoT device, an action that would have a significant overall cost in terms of general resources, storage, and power, if one takes into account the sheer number of IoT devices that are currently being used, which were estimated to be between 20 billion and 100 billion in 2020 [99, 100]. Additionally, we note that most works have so far failed to prove whether the PUF-based solutions that they propose are truly practical in the context of the IoT. As we will show, most of the examined works have considered "strong" PUFs for enhancing the security of IoT devices and networks, and/or have only examined potential PUF-based solutions in a rather theoretical manner, failing in this way to suggest a truly practical security solution for the IoT.

In 2016, Halak et al. [99] published an overview of PUF-based security solutions for the communication of IoT devices. The authors note that a significant portion of the data sent and received in the IoT, deal with private information, which, therefore, needs to be protected from potential eavesdropping and/or from being tampered with. The authors also note that most IoT devices have limited resources and, thus, the usual cryptographic schemes cannot be applied to them, while PUFs can be easily implemented in such devices with a minimal cost. To this end, the authors present and discuss a number of existing PUF-based security solutions for the IoT, as well as their outstanding reliability and security problems, with a particular focus on PUF-based authentication and PUF-based encryption/decryption schemes. However, this work provides a rather generic overview, and does not provide a thorough discussion regarding the implementation and testing of a comprehensive PUF-based protocol that would secure both IoT devices and networks. In particular, the authors do not consider the diversity of IoT networks, or the number of IoT devices deployed, which may make protocols that are based on the implementation of a PUF on each IoT device, very difficult to scale.

Also in 2016, Aman et al. [101] published a position paper regarding the usage of PUFs for IoT security. This work also notes the significant drawbacks of applying classical cryptography to IoT devices, especially due to potential physical and side-channel attacks. The authors then show that IoT devices can be efficiently protected from such security attacks with the help of PUFs. They also propose a preliminary PUF-based protocol for the mutual authentication of IoT systems, which is proven to be suitable for the realisation of efficient and strong security protocols for IoT devices. However, a thorough discussion regarding the implementation and testing of a comprehensive PUF-based protocol that would secure both IoT devices and networks, is again missing.

Again in 2016, Idriss et al. [102] noted that, due to the resource-constrained nature of IoT devices, traditional cryptography cannot be applied to them. They, therefore, proceed to evaluate the performance of several strong delay-based PUFs, i.e., of the arbiter PUF, the XOR PUF, the Lightweight Secure (LS) arbiter PUF, and the feed-forward arbiter PUF, in different IoT-related scenarios. The authors distinguish among cases where an adversary can have full, limited, or no access to the CRPs exchanged between devices, while also taking into account the manufacturing costs of the relevant PUFs, in order to suggest, in each case, the most optimal strong PUF design for the different security schemes con-

sidered. This work again lacks a protocol implementation, and suffers from the issues associated with strong PUFs, which have been discussed in Section 1.6.

Once again in 2016, Wallrabenstein [103] also noted that IoT devices require cost-efficient security solutions that can be offered in the form of PUFs. The author presents and evaluates an authentication protocol for IoT devices that utilises both PUFs and elliptic curves, which is an improved version of a PUF-based protocol proposed in the relevant literature [104]. The author also proposes a number of PUF-based algorithms that can be utilised for device enrollment, authentication, decryption, and digital signature generation, and evaluates their performance on a resource-constrained device, in order to prove their suitability for enhancing the security of IoT devices. Nevertheless, the author actually proposes a strong PUF implementation on an FPGA, which suffers from the issues discussed in Section 1.6. Moreover, the author also does not consider the diversity of IoT networks, or the number of IoT devices deployed, which may make protocols that are based on the implementation of a PUF on each IoT device, very difficult to scale.

Yet again in 2016, Mukhopadhyay [105] presented another overview of PUF-based security solutions for the IoT. The author notes that security issues affect the adoption of the IoT and discusses a potential security vulnerability of a common IoT setup. He notes that PUFs can act as a low-cost security mechanism for IoT devices, although a number of issues still persist. The author also proposes a lightweight PUF construction and attacks against it, as well as an authentication protocol and attacks against it. Finally, the author discusses the potential of PUF-based protocols for securing IoT devices and networks. Nevertheless, also this work considers a strong PUF implementation on an FPGA device, which suffers from the issues discussed in Section 1.6. This author also does not consider the diversity of IoT networks, or the number of IoT devices deployed, which may make protocols that are based on the implementation of a PUF on each IoT device, very difficult to scale.

Moreover, between 2016 and 2018, a series of papers were published, discussing PUF-based authentication for IoT devices, with a focus on devices used in biometric, fitness, and healthcare applications [84, 106, 107, 108, 109, 110, 111]. In these works, the authors again note the apparent suitability of PUFs to serve as adequate security solutions for IoT devices, especially in privacy-critical use cases, where personal data are directly exchanged through the IoT. Nevertheless, these works are based on measurements performed using DRAM modules connected to an FPGA in order to realise and test a DRAM PUF and, therefore, cannot be considered to truly evaluate and test a practical security solution for the IoT. Furthermore, they also do not consider the diversity of IoT networks, or the number of IoT devices deployed, which may make protocols that are based on the implementation of a PUF on each IoT device, very difficult to scale.

In 2017, Chatterjee et al. [112] proposed a PUF-based communication protocol for the IoT. Their protocol utilises an identity-based encryption scheme, and the Weil and Tate pairing algorithms for elliptic curves. In order to generate the identity of each device, which is used for message encryption, the protocol uses a PUF. The authors also provide formal proofs of the protocol's security in the Session Key Security Model and the Universally Composable Framework. Finally, the authors present a proof-of-concept implementation that incurs only low hardware and software overheads. However, also these authors consider a strong PUF implementation on an FPGA device, which suffers from the issues discussed in Section 1.6. Additionally, the authors also do not consider the diversity of IoT networks, or the

number of IoT devices deployed, which may make protocols that are based on the implementation of a PUF on each IoT device, very difficult to scale.

In 2018, Lipps et al. [113] presented a proof-of-concept implementation of an SRAM PUF-based protocol for IoT device authentication using a COTS device. The authors note that the large number of interconnected devices allows for an increased number of attacks against the IoT, and propose the usage of SRAM PUFs as security primitives for the implementation of symmetric key cryptography and device authentication. They evaluate their PUF implementation using the entropy of the PUF responses as a metric, and test the stability of this implementation under both temperature and voltage variations. Their results show that the proposed PUF implementation is robust to both voltage and temperature variations, as it exhibits a high amount of entropy under all the conditions tested. However, their protocol does not take into account the diversity of IoT networks, or the number of IoT devices deployed, which may make protocols that are based on the implementation of a PUF on each IoT device, very difficult to scale.

Also in 2018, Mahalat et al. [114] published a work regarding a PUF-based lightweight protocol for the secure Wi-Fi authentication of IoT devices. The authors note that security issues hinder the adoption of IoT devices, and propose the utilisation of PUFs to address this issue, due to the resource-constrained nature of low-end IoT devices. The authors also observe that current Wi-Fi protocols cannot guarantee sufficient security against a number of different threats, e.g., Media Access Control (MAC) spoofing. They further note that current PUF-based IoT protocols are not viable due to their resource and computation requirements, and propose a lightweight PUF-based protocol that can establish a secure connection and generate a random number to be used as a seed for keys produced by the underlying Wi-Fi protocol. The proposed protocol is proven to be secure and effective, while requiring low resources. Nevertheless, this work is rather theoretical, and the proposed protocol requires a strong tamper-proof PUF. Strong PUFs suffer from the issues discussed in Section 1.6, and the most well-known strong PUFs should not be considered as tamper-proof [8].

Based on this brief overview of the most recent and most relevant literature, we can conclude that existing works do not comprehensively address the issue of practical lightweight security for the IoT using PUFs. In particular, most works require a strong PUF or other hardware additions, which would raise the manufacturing costs of IoT and potentially hinder its further adoption. Additionally, we note here again that strong PUFs have been proven to be so vulnerable to attacks that their very existence has been called into question. The few works that consider weak PUFs do not test in depth whether such PUFs can truly act as practical security solutions that would secure both IoT devices and networks. Moreover, most works suggest that one PUF be implemented in each IoT device, in order to be used as a root of trust or security anchor. Nevertheless, given the huge number of IoT devices being used, such a security solution may not be able to scale well. In general, we observe that most of the related work tends to be rather theoretical and to not fully address the security of current and/or next-generation IoT devices, networks, applications and use cases in a comprehensive manner. In order to address these issues, our work discusses, examines, and evaluates in depth the ability of weak memory-based PUFs to serve as adequate, scalable and lightweight security mechanisms for IoT devices and networks, in practice.

2.3 Related Works Regarding Reconfigurable PUFs

Reconfigurable PUFs utilise a Reconfiguration Module (RM) in order to enhance the entropy of the responses of a particular PUF. Such RMs can be hash or encryption functions, or even other PUFs. In general, reconfigurable PUFs (rPUFs) are quite similar to the concept of *controlled* PUFs. Controlled PUFs [36, 38, 39, 40] utilise hash functions as a way to isolate the PUF from the untrusted external environment [17]. In particular, challenges and responses are, respectively, hashed before they are used and after they are produced.

In this way, an attacker not only cannot access the actual challenge provided to the PUF, which is a hash of the raw challenge, but he/she cannot also know the raw response produced by the PUF, as only a hash of it is provided as the final response. Additionally, as the hash can be salted or keyed, the actual challenge, as well as the final response, can be easily *reconfigured* by changing the salt(s) or the key(s) being used. In this case, the final response is dependent not only on the raw challenge, but also on the salt (or key) being used in the hashing of the raw challenge, as well as the salt (or key) being used in the hashing of the raw response. Therefore, the dependence between the (raw) challenge and the (final) response, which are the only things that an external attacker can observe, is not absolute, and thus modelling and machine learning attacks (which are based on the collection and observation of CRPs, in order to determine the function that connects the challenges to the responses) can be prevented.

Reconfigurable PUFs can be viewed as a simplification of controlled PUFs, where only one part of the CRP is altered, namely the response. For example, a salted hashing of the raw PUF response does break the absolute dependency of the final response on the challenge. On the other hand, in this case, the PUF is not fully isolated from the untrusted external environment and, if the salt is not regularly changed, modelling and machine learning attacks may still be possible. Nevertheless, rPUFs can also be considered as an extension of controlled PUFs as their RM does not need to be a hash, or a particular algorithm, but can also be any source of entropy, including Random Number Generators (RNGs) or other PUFs.

A number of reconfigurable PUFs have been proposed in the relevant literature. Naturally, the most important issue in the context of reconfigurable PUFs is the reconfiguration mechanism, i.e., the way in which the RM works. In particular, reconfigurable PUFs can be categorised into *logically* and *physically* reconfigurable PUFs [115, 116]. Logically reconfigurable PUFs employ a logical interface, such as a hash algorithm, in order to reconfigure the PUF's response, while physically reconfigurable PUFs are based on the intrinsic ability of the original PUF characteristics to be reconfigured in a physical way, i.e., to significantly change through a physical process, e.g., through temperature or voltage variations [33], or aging [117]. In our work, we will examine logically reconfigurable PUFs, as such PUFs can be easily implemented with COTS electronics being regularly used in IoT and CPS devices.

[17] The original definition of controlled PUFs in [38] is rather problematic, as it defines them as PUFs "that can only be accessed via an algorithm that is physically linked to each one of them in an inseparable way (i.e., any attempt to circumvent the algorithm will lead to the destruction of the PUF)". Nevertheless, as PUFs are physical objects (quite often hardware) and "algorithms" usually refer to software, e.g., hash functions, it is rather difficult, if not impossible, to find any instance that can truly match the definition, i.e., a PUF that really is inseparable from an algorithm in such a way that accessing it in a different way will destroy the PUF. While it may be possible to physically link a PUF to some algorithm, e.g., through a hardware implementation of the relevant algorithm, it is rather difficult to ensure that the PUF cannot be accessed otherwise, especially by an attacker that may have unrestrained physical and logical access to the overall module constituting the "controlled PUF".

In 2009, Kursawe et al. [33] published the first work regarding reconfigurable PUFs. Two physically reconfigurable PUFs were proposed, one that was based on reconfiguring an optical PUF, the PUF proposed by Pappu et al. in [34, 35], and another one that was based on a phase-change memory. In this case, the goal was to use the proposed reconfigurable PUFs in order to protect non-volatile storage against invasive physical attacks.

In 2011, Katzenbeisser et al. [17] proposed the concept of logically reconfigurable PUFs, a concept that they noted that was directly related to controlled PUFs. The authors also noted that the overhead introduced by converting a regular PUF to a logically reconfigured one was rather small, and that the concept could be optimised in terms of speed or area, in order to be applicable to resource-constrained devices. Logically reconfigurable PUFs are shown to be both forward and backward unpredictable. Finally, their potential applications could include usage as authentication tags and tokens.

In the same year, Eichhorn et al. [118] introduced a reconfigurable PUF that utilised a hash algorithm as its RM, in a logically reconfigurable PUF design. In this case, the state of a Non-Volatile Memory (NVM) was used as a salt when the hash of the PUF response was calculated. As the state of the NVM could be changed, the response of the rPUF could also be reconfigured. The overall cost for the implementation of this solution was estimated to be rather low.

Again in 2011, Lao and Parhi [119] compared the use of different RMs. In particular, the use of a Linear Feedback Shift Register (LFSR), and a hash function were studied as reconfiguration methods for the challenges of a PUF, while output recombination, and reconfigurable designs based on the feed-forward arbiter PUF, and a PUF based on a combination of the MUX and DeMUX functions, were also examined.

More recent works by Zhang et al. [115, 120] proposed a number of novel reconfigurable PUF designs based on phase-change memory. Additionally, finite state machines have been proposed both for the implementation of controlled [121] and of reconfigurable [122] PUFs. Moreover, reconfigurable PUFs based on bistable rings [123], ring oscillators [123, 124], hybrid oscillators [125], arbiters [126], as well as novel SRAM designs [127], have also been proposed. Furthermore, it has been noted that reconfigurable PUFs whose configuration is permanently altered each time they are reconfigured, such as reconfigurable optical PUFs [33, 128], can be utilised as PUFs whose CRP set is erasable. We note here that Rührmair et al. [53, 129] had suggested the notion of *erasable* PUFs in which a single CRP could be erased, with the remaining CRPs remaining unaffected.

Rather interestingly, a number of works have suggested the use of a PUF as an RM [130, 131]. In this case, the responses of one PUF are modified through the operation mechanism of another PUF. Even more interestingly, the response of a particular PUF to a specific challenge can be fed back to the same PUF as a challenge, resulting in a new response that can be used as the final response [48]. In this way, the same PUF can act as both the regular PUF and its own RM in a self-reconfiguration scheme. Moreover, we need to note that the DRAM decay-based PUF can be considered as a reconfigurable PUF by its definition, as its challenge is based not only on the memory addresses being used, but also on the time that the relevant DRAM cell values are allowed to decay, as well as the relative ambient temperature, as its response is strongly dependent on the ambient temperature.

In this work, we focus on reconfigurable PUFs that are comprised of COTS components and lightweight software, in order to allow for the implementation of advanced security applications, es-

pecially for IoT devices and networks, in a lightweight, cost-efficient, and practical manner. In particular, we propose and examine two protocols that allow to re-establish security in a device after it has been compromised. In this way, we propose a new paradigm of security, which recognises that successful attacks are rather unavoidable and therefore does not attempt to claim perfect security at all times, but rather aims to ensure an *acceptable* balance between security, risk and costs, and allow for successful recovery of security in a practical way, after a successful attack.

3 Memory-Based PUF Designs and Relevant Quality Metrics

Partially Based Upon Peer-Reviewed Content Published In

- W. Xiong, A. Schaller, N. A. Anagnostopoulos, M. U. Saleem, S. Gabmeyer, S. Katzenbeisser & J. Szefer, "Run-Time Accessible DRAM PUFs in Commodity Devices", Proceedings of the 18th International Conference on Cryptographic Hardware and Embedded Systems (CHES 2016), vol. 9813 of "Lecture Notes in Computer Science (LNCS)", pp. 432-453, Springer, 2016. DOI: 10.1007/978-3-662-53140-2_21
- A. Schaller, W. Xiong, N. A. Anagnostopoulos, M. U. Saleem, S. Gabmeyer, S. Katzenbeisser & J. Szefer, "Intrinsic Rowhammer PUFs: Leveraging the Rowhammer Effect for Improved Security", Proceedings of the 2017 IEEE International Symposium on Hardware Oriented Security and Trust (HOST 2017), IEEE, 2017. DOI: 10.1109/HST.2017.7951729
- N. A. Anagnostopoulos, T. Arul, M. Rosenstihl, A. Schaller, S. Gabmeyer & S. Katzenbeisser, "Low-Temperature Data Remanence Attacks Against Intrinsic SRAM PUFs", Proceedings of the 21st Euromicro Conference on Digital System Design 2018 (DSD 2018), pp. 581-585, IEEE, 2018. DOI: 10.1109/DSD.2018.00102
- N. A. Anagnostopoulos, S. Katzenbeisser, J. Chandy & F. Tehranipoor, "An Overview of DRAM-Based Security Primitives", Cryptography, vol. 2, iss. 2, MDPI, 2018. DOI: 10.3390/cryptography2020007
- A. Schaller, W. Xiong, N. A. Anagnostopoulos, M. U. Saleem, S. Gabmeyer, B. Škorić, S. Katzenbeisser & J. Szefer, "Decay-Based DRAM PUFs in Commodity Devices", IEEE Transactions on Dependable and Secure Computing (TDSC), vol. 16, iss. 3, pp. 462-475, IEEE, 2018. DOI: 10.1109/TDSC.2018.2822298
- N. A. Anagnostopoulos, T. Arul, Y. Fan, J. Lotichius, C. Hatzfeld, F. Fernandes, R. Sharma, F. Tehranipoor & S. Katzenbeisser, "Securing IoT Devices Using Robust DRAM PUFs", Proceedings of the 2018 Global Information Infrastructure and Networking Symposium (GIIS 2018), IEEE, 2018. DOI: 10.1109/GIIS.2018.8635789
- N. A. Anagnostopoulos, T. Arul, Y. Fan, M. Kumar & S. Katzenbeisser, "AR-PUFs: Advanced Security Primitives for the Internet of Things and Cyber-Physical Systems", Proceedings of the 2019 IEEE International Conference on Consumer Electronics (ICCE 2019), IEEE, 2019. DOI: 10.1109/ICCE.2019.8661840
- N. A. Anagnostopoulos, Y. Fan, T. Arul, R. Sarangdhar & S. Katzenbeisser, "Lightweight Security Solutions for IoT Implementations in Space", Proceedings of the 2019 IEEE Topical Workshop on Internet of Space (TWIOS 2019), IEEE, 2019. DOI: 10.1109/TWIOS.2019.8771257
- L. Negka, G. Gketsios, N. A. Anagnostopoulos, G. Spathoulas, A. Kakarountas & S. Katzenbeisser, "Employing Blockchain and Physical Unclonable Functions for Counterfeit IoT Devices Detection", Proceedings of the 1st International Conference on Omni-Layer Intelligent Systems (COINS 2019), pp. 172-178, ACM, 2019. DOI: 10.1145/3312614.3312650
- N. A. Anagnostopoulos, T. Arul, M. Rosenstihl, A. Schaller, S. Gabmeyer & S. Katzenbeisser, "Attacking SRAM PUFs Using Very-Low-Temperature Data Remanence", Microprocessors and Microsystems, vol. 71, art. 102864, Elsevier, 2019. DOI: 10.1016/j.micpro.2019.102864

Partially Based Upon Additional Peer-Reviewed Content Published In

- D. Püllen, N. A. Anagnostopoulos, T. Arul & S. Katzenbeisser, "Using Implicit Certification to Efficiently Establish Authenticated Group Keys for In-Vehicle Networks", Proceedings of the 2019 IEEE Vehicular Networking Conference (VNC 2019), IEEE, 2019. DOI: 10.1109/VNC48660.2019.9062785
- T. Arul, N. A. Anagnostopoulos, S. Reißig & S. Katzenbeisser, "A Study of the Spatial Auto-Correlation of Memory-Based Physical Unclonable Functions", Proceedings of the 2020 European Conference on Circuit Theory and Design (ECCTD 2020), IEEE, 2020. DOI: 10.1109/ECCTD49232.2020.9218302
- N. A. Anagnostopoulos, S. Ahmad, T. Arul, D. Steinmetzer, M. Hollick & S. Katzenbeisser, "Low-Cost Security for Next-Generation IoT Networks", ACM Transactions on Internet Technology, vol. 20, iss. 3, art. 30, ACM, 2020. DOI: 10.1145/3406280

In this section, we present and discuss a number of memory-based PUF implementations, which are based on COTS components, and examine their characteristics and their potential to act as adequate security mechanisms for the IoT. In particular, we introduce here a number of novel memory-based PUF designs that will then be further examined in more detail for the purposes of this work, especially regarding their ability to provide adequate security under adverse conditions (Section 4) and in the context of practical security protocols for the IoT (Section 5). Additionally, we also present and examine metrics that can be utilised for the evaluation of the quality of the responses of the aforementioned memory-based PUF designs, in order to then utilise them to evaluate, in Section 4, the security that such PUFs can provide under nominal and adverse conditions. Finally, in this section, we also discuss the relation between security and cost, in the context of practical security for the IoT, in order to determine if the memory-based PUF designs presented in this section can provide an *acceptable* level of security, in terms of both cost and risk mitigation. We observe here that, although the relation between cost and security is rather essential for practical security applications, especially in the context of the IoT, which is rather based on economies of scale, this relation does tend to be rather often overlooked in scientific works, which quite often examine rather theoretical and idealised use cases and application scenarios.

3.1 Intrinsic Memory-Based PUF Designs on IoT Devices

As already discussed, in recent years, a variety of memory-based PUF designs have been proposed for enhancing the security of IoT devices and communications. Nevertheless, it needs to be noted that, although memory-based PUFs were firstly proposed some time ago (the SRAM PUFs in 2007, the Flash PUFs in 2011, and DRAM-based PUFs in 2012), the advantages of using such PUFs for securing IoT devices and networks were not immediately investigated in a thorough manner. As Section 2 demonstrates, most of the previous works tended to focus on the potential of *strong* PUFs for securing the IoT, broadly disregarding the well-known issues of such PUFs. Additionally, works regarding memory-based PUFs did not always present PUF designs that could be utilised immediately in commercial devices, as the relevant memories were most often not inherent components of a pre-existing computer system, but rather individual components tested using an FPGA [15, 70, 74].

Table 3.1: Overview of the Novel Memory-Based PUFs Examined in This Work

Device	Memory Model	Size	PUF Type
Texas Instruments Stellaris LM4F120 LaunchPad evaluation board (EK-LM4F120XL)	on-die SRAM of the LX4F120H5QR MCU	32 KB → 2^{18} bits	SRAM start-up
Texas Instruments Tiva C Series TM4C123G LaunchPad evaluation board (EK-TM4C123GXL)	on-die SRAM of the TM4C123GH6PM MCU	32 KB → 2^{18} bits	SRAM start-up
ST Microelectronics STM32F407 Discovery (STM32F407G-DISC1)	embedded SRAM on the STM32F407VGT6 MCU	192 KB → 1.5×2^{20} bits (64 KB → $0.5 \times 2^{20} = 2^{19}$ bits are Core-Coupled Memory (CCM))	SRAM start-up
ST Microelectronics STM32 Nucleo-64 NUCLEO-L152RE board	embedded SRAM on the STM32L152RET6 MCU	80 KB → 1.25×2^{19} bits	SRAM start-up
Qualcomm Atheros QCA9500 wireless communication module of the TP-Link Talon AD7200 router	SRAM [†] of the Qualcomm Atheros QCA9500	"blob_fw_code": 256 KB → 2^{21} bits (labelled as the firmware code RAM) "blob_uc_code": 128 KB → 2^{20} bits (labelled as the microcode code RAM)	SRAM start-up
PandaBoard ES Rev B3	package-on-package (on the Texas Instruments OMAP 4460 System on a Chip (SoC)) Elpida (Micron) EDB8164B3PF-8D-F DRAM	1 GB → 2^{33} bits	DRAM decay, and DRAM Row Hammer
Intel Galileo Gen 2	2 × on-board Micron MT41K128M8 DRAM	2 × 128 MB → $2 \times 2^{30} = 2^{31}$ bits	DRAM decay
Waveshare NandFlash Board (A) (removable external Flash memory module)	Samsung K9F1G08U0E NAND Flash memory	1 Gbit → 2^{30} bits	Flash program disturbance
ST Microelectronics STM32F407 Discovery (STM32F407G-DISC1)	embedded NAND Flash memory on the STM32F407VGT6 MCU	1 MB → 2^{23} bits	Flash program disturbance
ST Microelectronics STM32 Nucleo-64 NUCLEO-L152RE board	embedded NAND Flash memory on the STM32L152RET6 MCU	512 KB → 2^{22} bits	Flash program disturbance

[†] The nature of this memory is not certain. Its characteristics, however, point decisively towards SRAM. Moreover, different memory regions, as defined for the wil6210 driver (https://wireless.wiki.kernel.org/en/users/drivers/wil6210), may actually utilise instances of different memory modules.

In this work, we present a number of novel designs for memory-based PUF implementations, which utilise only COTS components that are most often integral parts of pre-existing devices. In this way, we are able to utilise the advantages that such PUFs offer as security mechanisms (which were discussed in Section 1), in order to propose the implementation of practical lightweight security protocols for the IoT in Section 5. In particular, in this subsection, we examine the individual characteristics of existing as well as novel PUF designs based on SRAMs, DRAMs and Flash memories, within the context of the IoT. An overview of the novel memory-based PUFs examined in this work is provided in Table 3.1. Although these PUFs are considered as *weak*, we aim to prove that they can be used to adequately address the demand for low-cost security for the IoT in practice, by confirming that they can act as flexible, scalable, and cost-efficient security mechanisms in a practical manner, providing lightweight and yet adequate security.

Moreover, as already discussed, memory-based PUFs have different requirements based on the way they operate, e.g., the SRAM PUF requires a reboot in order to function properly, as it is based on the start-up values of the relevant SRAM. Table 3.2 compares the requirements for the operation of the different memory-based PUFs discussed in this work. We note that the time required to generate an adequate PUF response is, at most, a few minutes, even if a reboot is required. In order to convert the produced PUF response into a unique token (such as a key), e.g., by using a (reverse) fuzzy extractor scheme, only a few additional seconds are required [132], even when a resource-constrained device is used for this purpose, as [133] proves.

In general, we note that the responses of memory-based PUFs tend to be noisy, which means that each time a particular PUF is queried with the same challenge, it will provide a response that may potentially be slightly different from past and future responses for this challenge. In order to stabilise a PUF response and, at the same time, transform such a response into a cryptographic token, e.g., a key, a bit selection/correction scheme needs to be employed, as, otherwise, the noise observed in PUF responses corresponding to the same challenge and PUF instance could affect the PUF's reliability [134, 135]. Such schemes typically consist of two phases, an enrollment phase that needs to be run only once, and a reconstruction phase, which is supposed to be run every time the relevant cryptographic token needs to be used. During the one-off enrollment phase, a PUF response for a particular challenge is somehow combined with a pre-selected instance of the relevant cryptographic token, e.g., with a specific key, in order to produce what is known as helper data. Such helper data may, for example, consist of redundancy bits, or data indicating the positions of stable bits in the relevant PUF response, depending on the exact characteristics of the relevant PUF response. Then, every time the relevant cryptographic token needs to be utilised, it can be reconstructed by combining the relevant helper data with a new, potentially noisy, PUF response for the same challenge as the one used in the one-time enrollment phase.

The bit selection/correction mechanism discussed here, is most often based upon the use of a (reverse) fuzzy extractor scheme [16, 76, 136, 137, 138, 139], which incorporates a particular Error Correction Code (ECC) [76, 138, 139], with the helper data, in this case, being essentially redundancy bits containing information relevant to the error correction of the combination of the PUF response for a particular challenge and the cryptographic token that is produced and utilised in the enrollment phase. In this way, the combination of the helper data and a highly similar, up to some noise, PUF response for the same challenge, during the reconstruction phase, can remove an adequate number of errors from this

potentially noisy PUF response, for it to be successfully transformed into the cryptographic token that was utilised in the enrollment phase. Nevertheless, a simple bit selection scheme, where the most stable bits of a particular response are selected and arranged in a particular way, in order to produce a cryptographic token, such as a key, can also be successfully utilised for the same purpose. In general, both the way in which the cryptographic token and the PUF response are combined, during the enrollment phase, as well as the way in which the helper data are combined with another PUF response produced by the same challenge, during the reconstruction phase, in order to reproduce the cryptographic token chosen during the enrollment phase, are highly dependent upon the exact bit selection/correction mechanism being utilised. Moreover, it should be rather evident, that the challenge-response pair, the cryptographic token, and the helper data are the most important components of this process, which can be utilised for the implementation of secure key generation, storage, and agreement protocols, as well as for more advanced security applications. Furthermore, it is also rather obvious that the helper data, and the overall choice of the exact bit selection/correction scheme to be used, are highly dependent on the characteristics of each PUF, including the characteristics of the noise found in its responses.

Table 3.2: Comparison of the Operational Requirements for the PUFs Examined in This Work

PUFType	Requires Reboot	Lowest Total Response Time	Reference
SRAM PUF	yes	seconds	[140]
DRAM start-up PUF	yes	seconds	[141]
DRAM retention-based PUF	no	seconds	[142]
Row Hammer PUF	no	seconds	[143]
DRAM access latency-based PUF	no	microseconds	[144]
Flash PUF	no	minutes	[70]

In the subsequent text segments of this work, we will examine in more detail the characteristics of the memory-based PUFs compared in Table 3.2, focusing particularly on the memory-based PUF designs utilised for the purposes of this work, which are presented in Table 3.1. We also note here that these designs – which are presented in Table 3.1 – have been based upon commercial memory components that most often form inherent components of COTS devices, and not on FPGA or ASIC components or devices, in order to examine whether such PUFs can truly be considered for practical applications or not.

3.1.1 The Static Random Access Memory (SRAM) PUF

The SRAM PUF is based upon the start-up values of the un-initialised cells of the relevant SRAM. In particular, one may consider the addresses of these cells as the PUF challenge, while the concatenation of the values of these cells at start-up is the corresponding PUF response. It should be evident that the

ordering of the memory addresses being used in the challenge, as well as the number of addresses being used, will affect the result of the relevant concatenation of cell values and, therefore, the relevant PUF response. Additionally, the exact time at which the relevant start-up values are measured may affect the quality of the measurement, as (some of) the relevant SRAM cells may be utilised (and/or pre-initialised) by the system during the start-up process, in which case their content will not be based on the intrinsic characteristics of the SRAM module – as it will not be random and "unique", and can therefore not be used in the production of a valid SRAM PUF response.

Moreover, as already discussed, the SRAM PUF most often requires a reboot of the host device of the relevant SRAM, as SRAMs are most often used as caches for MicroContoller Units (MCUs) and can therefore not be power-cycled on their own, in order to gain access to the start-up values of their cells, which form the response of each SRAM PUF. In general, SRAM modules tend to have a smaller storage capacity, having fewer memory cells, than DRAM and Flash memory modules. This leads SRAM PUFs to have a smaller size (in terms of memory cells) than DRAM-based PUFs and Flash PUFs, which can also result in the responses of SRAM PUFs having, in total, a lower degree of entropy than the ones of the other two categories of memory-based PUFs examined in this work. Nevertheless, SRAM PUFs can be easily implemented on almost any modern computer system, as SRAMs is almost ubiquitous in such devices, acting as caches or as data buffers, and most often can be easily accessed even by non-privileged software. In this work, we examine SRAM PUFs implemented using the SRAMs modules of a number of distinct COTS devices. In particular, we have implemented SRAM PUFs utilising the SRAMs of Texas Instruments Stellaris LM4F120 LaunchPad evaluation boards (EK-LM4F120XL), of Texas Instruments Tiva C Series TM4C123G LaunchPad evaluation boards (EK-TM4C123GXL), of ST Microelectronics STM32F407 Discovery (STM32F407G-DISC1) boards, of ST Microelectronics STM32 Nucleo-64 NUCLEO-L152RE boards, and of the Qualcomm Atheros QCA9500 wireless communication module of a number of TP-Link Talon AD7200 routers.

Regarding the PUF that utilises the memory of the Qualcomm Atheros QCA9500 wireless communication module of the TP-Link Talon AD7200 router, we need to note that it is not certain that the memory module that is being used as a PUF is an SRAM, as this information is not publicly available. Nevertheless, the characteristics of the PUF responses of this implementation strongly suggest that this implementation is indeed an SRAM PUF. Additionally, the different memory regions of the Qualcomm Atheros QCA9500 FullMAC Institute of Electrical and Electronics Engineers (IEEE) 802.11ad Wi-Fi chip of the TP-Link Talon AD7200 router can be accessed utilising the wil6210 driver [18]. Not all of these memory regions provide start-up values that could be utilised for the implementation of a start-up PUF after a reboot, as the result of the concatenation of such values per memory region – in the way that such memory regions are defined for the wil6210 driver, as shown in Table 3.3 – does not always seem to exhibit a high degree of uniqueness and/or randomness. This has been asserted through the employment of the relevant metrics, which are discussed in Section 3.2. A photo of the TP-Link Talon AD7200 network router is shown in Figure 3.1. Implementing a PUF on this device is an important achievement, as this device is a next-generation tri-band network router, which supports wireless communication at 2.4 GigaHertz (GHz), 5 GHz and 60 GHz. Thus, a PUF implemented on this router could be utilised to secure next-generation wireless communications.

[18] https://wireless.wiki.kernel.org/en/users/drivers/wil6210

Figure 3.1: A photo of the TP-Link Talon AD7200 network router, which supports wireless communication at 2.4 GHz, 5 GHz and 60 GHz.

In particular, we note that certain segments of different memory regions of the Qualcomm Atheros QCA9500 wireless communication module exhibit good PUF characteristics, with the concatenation of their start-up values demonstrating a high degree of robustness, uniqueness, *and* randomness. More specifically, segments of the memory regions denoted in the wil6210 driver as "blob_acg_tbl", "blob_rgf", "blob_fw_code", "blob_fw_data", "blob_fw_peri", "blob_upper", "blob_uc_code", and "blob_uc_data" exhibit good PUF characteristics.

Nevertheless, the values acquired from the relevant memory segments of the "blob_upper" memory region appear to be (mostly) identical to those of the other memory regions, in the order of "blob_fw_code", "blob_fw_data", "blob_fw_peri", as well as "blob_uc_code", and "blob_uc_data". Values between the memory segments of the "blob_upper" memory region that contain values (mostly) identical to those of the "blob_fw_peri" memory segment and to those of the "blob_uc_code" memory segment, as well as values of the memory segment of the "blob_upper" memory region that follows the memory segment of the "blob_upper" memory region that contains values (mostly) identical to those of the "blob_uc_data", seem to be coming from the unnamed memory regions between the "blob_fw_peri" and "blob_uc_code" memory regions and after the "blob_uc_data" memory region, respectively.

However, the largest memory segment that exhibits good PUF characteristics without a break in its continuity is the "blob_uc_code" memory region, the whole (128 KB) of which exhibits a high degree of robustness, uniqueness, *and* randomness. A segment of the "blob_fw_code" memory region also exhibits good PUF characteristics, and has been utilised as an SRAM PUF in [145], but it is only 221660 bits (27707.5 bytes $\approx$ 27.058 KB) long, while the whole "blob_fw_code" memory region is 256 KB long. Therefore, in this work, we examine the PUF characteristics of the "blob_uc_code" memory region instead.

Table 3.3: Memory Layout of the WIL6210 Card for the Qualcomm Atheros QCA9500 Wireless Communication Chip Module

Memory Region Label	Size in Bytes	Description
"blob_rgf"	40 KB	Various Register Files
"blob_AGC_tbl"	4 KB	AGC Table
"blob_rgf_ext"	4 KB	PCI-Express Register File Extension
"blob_mac_rgf_ext"	512 B	MAC Register File Extension
"blob_fw_code"	256 KB	Firmware Code RAM
"blob_fw_data"	32 KB	Firmware Data RAM
"blob_fw_peri"	128 KB	Peripheral Data
"blob_uc_code"	128 KB	Microcode Code RAM
"blob_uc_data"	16 KB	Microcode Data RAM
"blob_upper"	548 KB	Upper Memory Area

Moreover, we need to note that the different memory regions of the Qualcomm Atheros QCA9500 wireless communication module may actually utilise different types of memory modules. The relevant wil6210 driver seems to claim that the "wil6210 cards have on-board Flash memory for the firmware, the cards come pre-flashed and no firmware download is required" [19]. However, the PUF characteristics of the relevant memory do not seem to support the claim that Flash memory is being used to store the firmware, as the relevant characteristics indicate a behaviour rather similar to that of SRAM PUFs.

Regarding the Texas Instruments devices, i.e., the Texas Instruments Stellaris LM4F120 LaunchPad evaluation board (EK-LM4F120XL) and the Texas Instruments Tiva C Series TM4C123G LaunchPad evaluation board (EK-TM4C123GXL), which are shown in Figure 3.2, SRAM PUFs implemented using the relevant SRAMs have been proven in the relevant literature to exhibit good PUF characteristics [146]. We have utilised these devices to test the effects of very low temperatures on SRAM PUFs, by investigating the relevant data remanence phenomena that can lead to successful attacks against these PUFs [3, 4]. Regarding the ST Microelectronics STM32F407 Discovery (STM32F407G-DISC1) and the ST Microelectronics STM32 Nucleo-64 NUCLEO-L152RE boards, which are shown in Figure 3.3, we have implemented SRAM PUFs utilising their intrinsic SRAMs, in order to test the resilience of such PUF to radiation [147]. By testing the resilience of SRAM PUFs to low temperatures and to radiation, we were able to assess whether they can be utilised in IoT applications that potentially involve adverse operating

[19] https://android.googlesource.com/kernel/common/+/2be7d22f062535de59babdb4b5e9de9ff31e817e

conditions, e.g., in space missions and other relevant applications. The quality of the responses of the aforementioned SRAM PUFs under the adverse conditions described is examined in detail in Section 4, while some of their potential security applications in the context of the IoT are discussed in Section 5, where relevant cryptographic protocols based on these PUFs are proposed and examined.

Figure 3.2: A photo of the Texas Instruments Stellaris LM4F120 LaunchPad evaluation board (EK-LM4F120XL) (on the left), and the Texas Instruments Tiva C Series TM4C123G LaunchPad evaluation board (EK-TM4C123GXL) (on the right).

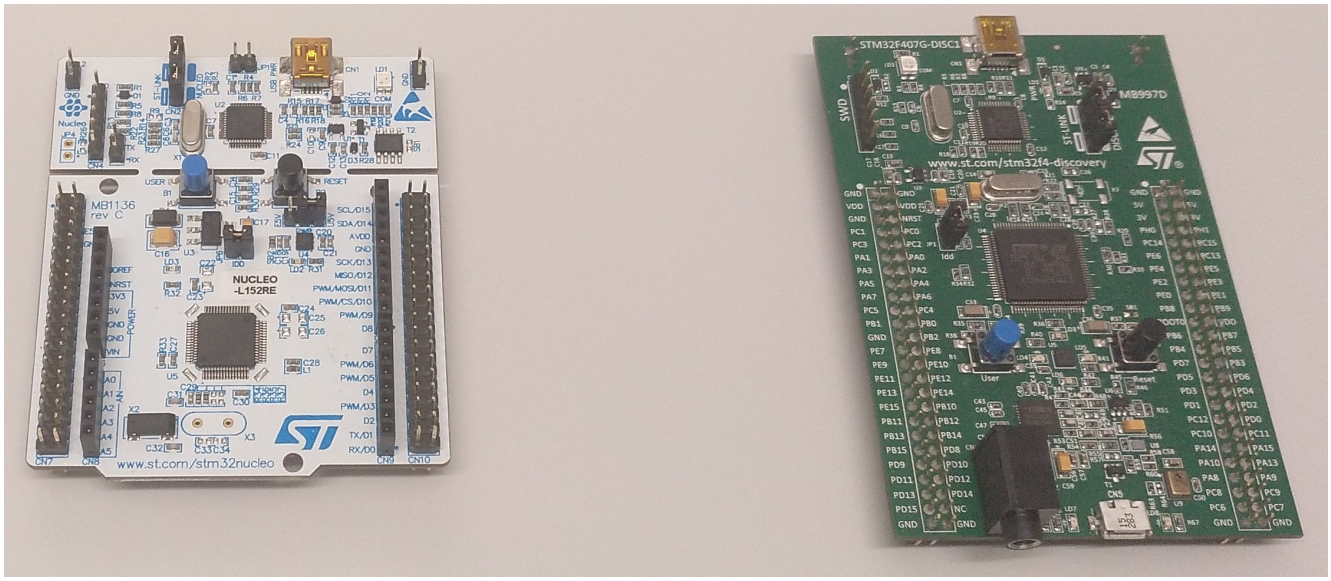


Figure 3.3: A photo of the ST Microelectronics STM32 Nucleo-64 NUCLEO-L152RE board (on the left), and the ST Microelectronics STM32F407 Discovery (STM32F407G-DISC1) board (on the right).

In general, the SRAM PUF, whose response is the concatenation of the start-up values of the cells of the SRAM module (and therefore a reboot is required for its operation), can be easily utilised for security applications. For example, such an SRAM PUF can be used in the context of a secure key storage/generation scheme in the following way:

1. When the device reboots, the bootloader queries the SRAM for its start-up values, which form the raw SRAM PUF response. In order to avoid data remanence, an adequate amount of time must have passed between the device being powered off and on.

2. Then, helper data for error correction are acquired either from the host device itself or from another device.

3. The raw PUF response and the helper data are combined, using a fuzzy extractor scheme, to generate a secret key.

4. Finally, the bootloader overwrites the SRAM cells with a specific logical pattern, which erases the raw SRAM PUF response from the SRAM module.

It is also worth noting that the helper data and the fuzzy extractor scheme used for the implementation of a PUF-based secure key storage/generation scheme, are almost always considered as publicly available information, with the only secrets being the response of the PUF and the resulting key. It is important to note that for the purposes of secure key agreement, an enrollment phase would also be required, during which one of the parties would gain access to the secret of the PUF response, select a key, and produce the relevant helper data. Then, every time the same key would need to be used by another party, this (second) party would need to access, during a reconstruction phase, the same PUF response and the relevant helper data, as described above, in order to reconstruct/reproduce the key selected by the first party during the enrollment phase. Hence, the PUF device should optimally be transferred from the first party to the second one between the enrollment and the reconstruction phases. Nevertheless, the scheme can also be implemented in a secure way, as long as access to the secret PUF response can only be gained by the legitimate parties and in a secure manner. Additionally, while helper data corresponding to a particular entity can be used in order to create different secure keys using responses from different PUF instances (or different responses from the same PUF), different helper data (corresponding to different entities) will result in different keys being produced when the same PUF response is used. Thus, a PUF can produce personalised keys and other cryptographic tokens, which can be utilised for the implementation of advanced cryptographic applications [20].

As SRAM PUFs are, most often, based on SRAMs that are inherent system components, they, most often, constitute intrinsic PUFs. Additionally, their operation does not require the utilisation of computationally heavy software, which further enhances their ability to act as lightweight security mechanisms. Moreover, depending on the size of the SRAM, SRAM PUFs may be able to generate multiple cryptographic tokens, e.g., keys, of varying sizes. Each of the SRAM PUFs examined in this work provides several KiloBytes (KB) as its response, which can be easily utilised to produce multiple 1024-bit or even 4096-bit keys. Even the main disadvantage of a reboot being required, in order for the SRAM PUF to provide its response, can also be seen as a security advantage, as it further prevents an attacker from gaining access to the SRAM PUF response.

[20] Essentially, a memory-based PUF response can serve in a manner similar to the endorsement key of a TPM, which is a cryptographic key confined to the TPM that serves as the security anchor of such a TPM. For example, in the described key storage/generation scheme, the SRAM PUF response can remain secret even to the actual user entity of the cryptographic primitive, as this entity needs to only provide the relevant helper data in order for the PUF to produce the corresponding key, without a need for the actual PUF response to be revealed. In this way, the PUF response can be utilised as an internal security anchor for the primitive itself, allowing the PUF to provide further, more advanced, services, such as authentication, attestation, secure boot, etc.

External SRAM can also be used for the implementation of an SRAM PUF in order to avoid rebooting the whole system, as internal SRAM is most often serving as a cache or a buffer and, therefore, its utilisation as an SRAM PUF requires the whole system to be rebooted. Additionally, such an external SRAM module can be utilised for the flexible implementation of an Advanced Reconfigurable PUF (AR-PUF), as we will discuss in Section 5. Nevertheless, the utilisation of an external SRAM as a PUF cannot, in general, be considered as truly cost-efficient and, therefore, practical, unless the relevant SRAM forms an inherent component of the overall system, as is the case, for example, with the memory of the Qualcomm Atheros QCA9500 wireless communication module, which is a peripheral connected to the main module of the TP-Link Talon AD7200 router, which plays the role of the host. In such a case, only the relevant module on which the SRAM is found (in this case the Qualcomm Atheros QCA9500 wireless communication module), needs to be rebooted every time the relevant SRAM PUF response needs to be accessed. Regarding the memory of the Qualcomm Atheros QCA9500 wireless communication module of the TP-Link Talon AD7200 router, occasional reboots should not significantly affect the operation of the main module of the router or even hinder the communication capabilities of the device.

In general, all of the SRAM PUFs examined in this work have been implemented using COTS components, and do not require the utilisation of FPGA or ASIC designs. In this way, we are able to test if SRAM PUF designs implemented in actual commercial devices, the SRAMs of which, function as normal memory modules when not being utilised as PUFs, can truly act as adequate security mechanisms in practice. To this end, we examine and prove the suitability of such PUFs to act as adequate security mechanisms even under adverse conditions in Section 4, in order to validate that they can be easily employed in the security protocols and applications proposed and discussed in Section 5. Finally, we also note that the responses of the SRAM PUF can also be combined with the responses of other intrinsic memory-based PUFs that are implemented utilising other memories of the same system, in order to implement a PUF that is not only highly reconfigurable, but can also provide enhanced security solutions [148].

3.1.2 Dynamic Random Access Memory (DRAM) PUFs

A number of different DRAM-based PUFs have been proposed and examined in the relevant literature. In particular, the most prominent examples of PUFs based on DRAM are the DRAM start-up PUF, which is based on the start-up values of the DRAM, the DRAM decay-based PUF, which is based on the data decay characteristics of the DRAM, when its values are not being refreshed, the DRAM Row Hammer PUF which is based on the row hammer effect as well as the data decay characteristics of the unrefreshed DRAM, and the DRAM latency-based PUF, which is based on the latency times allowed for writing to and reading from the DRAM cells.

In this work, we discuss the aforementioned four types of DRAM PUFs, but focus more on DRAM decay-based PUFs as well as on DRAM Row Hammer PUFs, because the DRAM decay-based PUFs allowed for the first run-time implementation of a DRAM-based PUF and both of these types of PUFs exhibit a number of rather interesting characteristics, e.g., they utilise the Jaccard index [149, 150] in order to properly evaluate the quality of their responses, and they exhibit a relatively high temperature dependency. Essentially, the leakage characteristics of the DRAM being utilised for the implementation of DRAM decay-based PUF are highly dependent on time and temperature, and therefore need to be evaluated utilising the Jaccard index, instead of the Hamming distance, as a metric, and ways to make

them robust against ambient temperature changes need to be found. The same also hold true for DRAM Row Hammer PUFs, as these PUFs are based on the combination of the effects of the DRAM row hammer effect and the data decay characteristics of the unrefreshed DRAM that form the basis of the operation mechanism of DRAM decay-based PUFs. This work looks into both the time dependency and the temperature dependency of such PUFs, in order to examine whether such PUFs can be utilised as practical lightweight security mechanisms for the IoT.

Regarding the DRAM decay-based PUF, the data decay characteristics are essentially defining the opposite phenomenon of data retention. Thus, such a PUF can also be characterised as a DRAM retention-based PUF. Indeed, at high temperatures or after an extremely long time with the data refresh functionality disabled, the defining phenomenon for such a PUF becomes the data retention, as the majority of bit values that can decay tend to have already done so. Moreover, it should be evident that the attributes of this PUF are based upon the leakage characteristics of the relevant DRAM cells. Therefore, obviously, also DRAM PUFs based on the data remanence of the DRAM while it is powered off can (and, probably, should) be considered as a sub-category of DRAM decay-based PUFs, albeit of its own. The main difference is that the absence of connection to power may also preclude the existence of (adequate) grounding, which is rather considered as granted while the device is connected to a power outlet. The relative restriction of the power load present may also further weaken the creation of sustainable leakage paths. Nevertheless, further work is required in order to examine the exact behaviour of the same DRAM acting as a PUF based on data remanence, which is exhibited during a power-off, and as a PUF based on data decay, when the data refresh functionality has been disabled, but the memory module is still being provided with power. Only in this way, it will be possible to properly assess the similarities and the differences between the two different ways in which a PUF based on the decay of the values of the DRAM cells can be implemented.

It is important to note that, in contrast to the SRAM PUF, some DRAM-based PUFs can operate at run-time, without the need for a reboot. In particular, a DRAM PUF based on the start-up values of the DRAM will obviously require a reboot of the DRAM module in order to gain access to the un-initialised start-up of the cells of the DRAM, while a DRAM decay-based PUF can operate without the need for a reboot, as the data refresh functionality of the DRAM can be disabled at run-time, sometimes even only for particular segments of the DRAM, and the DRAM latency-based PUF can also operate without the need for a reboot, as the latency times allowed for writing to and reading from the DRAM could also potentially be changed during run-time. Nevertheless, in practice, changing the latency times allowed for writing to and reading from the DRAM will require the utilisation of a dedicated DRAM controller firmware mechanism, in order to be able to function properly at run-time, without significantly hindering the normal operation of the DRAM. Additionally, DRAM Row Hammer PUFs can also be utilised at run-time without the need for a reboot, as their operation is based on the combination of the DRAM row hammer effect, which can be easily induced at run-time, even upon only a selection of DRAM regions, and the data decay caused by disabling the data refresh functionality of the DRAM, which can also be disabled at run-time, sometimes even only for particular segments of the DRAM. Therefore, DRAM decay-based PUFs and DRAM Row Hammer PUFs can potentially operate at run-time, affecting only particular segments of the DRAM for a limited amount of time.

Moreover, the size of current COTS DRAM modules ranges between hundreds of MegaBytes (MB) and tens of GigaBytes (GB), in contrast to COTS SRAM modules, which most often have a size of only a few tens or hundreds of MB. Thus, DRAM-based PUFs can provide, in general, larger responses than the SRAM PUF, which, in turn, can be utilised to produce more or larger cryptographic tokens. Naturally, however, the secure generation and storage of such cryptographic tokens, e.g., keys, is highly dependent upon the inherent entropy of the relevant PUF responses. In general, however, most, if not all, DRAM PUF types should be able to easily produce multiple 1024-bit or even 4096-bit keys. Moreover, DRAMs, like SRAMs are quite often inherent components of modern computer systems and, therefore, the relevant DRAM-based PUFs can typically be considered as intrinsic PUFs. Likewise to SRAM PUFs, DRAM-based PUFs also do not require the use of computationally heavy software for their operation, which reinforces their ability to act as lightweight security mechanisms. Nevertheless, also an external DRAM can be used for the implementation of a DRAM PUF. In this case, if the operation of the relevant DRAM PUF requires a reboot, it may be possible to power-cycle only the external DRAM itself, instead of the whole system. Additionally, such an external DRAM module can be utilised for the flexible implementation of an Advanced Reconfigurable PUF (AR-PUF), as we will discuss in Section 5. However, the utilisation of an external DRAM cannot, in general, be considered as truly cost-efficient and, thus, practical, unless the relevant DRAM can be considered to be part of the overall system or its use provides significant advantages, as is the case for AR-PUFs.

In the following paragraphs, we will discuss in more detail the most prominent types of DRAM-based PUFs, with a particular focus on DRAM decay-based PUFs and DRAM Row Hammer PUFs, as these two types of PUF designs have been chosen to be further examined in this work, due to their inherent characteristics. In particular, these DRAM-based PUFs exhibit a relatively high temperature dependency and, therefore, their resilience to ambient temperature variations is a critical factor to assess their ability to act as practical security mechanisms for the IoT. Moreover, these PUFs allowed for the first run-time implementations of DRAM-based PUFs, which can be considered as an extremely important attribute for the implementation of practical protocols. To this end, we will examine the resilience of such PUFs to ambient temperature conditions in Section 4, and we will describe and discuss the implementation of robust security protocols based on these PUFs in Section 5.

The DRAM Startup-Based PUF

DRAM startup-based PUFs operate in the same way as SRAM PUFs, as they are based on the start-up values of the un-initialised cells of the relevant DRAM module. Similar to SRAM PUFs, the addresses of these cells may be considered as the challenge of this PUF, while the concatenation of the values of these cells at start-up is the corresponding PUF response. Therefore, the ordering of the memory addresses being used in the challenge, as well as the number of addresses being used, will affect the result of the relevant concatenation of cell values and, therefore, the relevant PUF response. Additionally, the exact time at which the relevant start-up values are measured may affect the quality of the measurement, as (some of) the relevant DRAM cells may be utilised (and/or pre-initialised) by the system during the start-up process, in which case their content will not be based on the intrinsic characteristics of the DRAM module – as it will not be random and "unique", and can therefore not be used in the production of a valid DRAM startup-based PUF response.

Additionally, as already discussed, the DRAM startup-based PUFs most often requires a reboot of the host device of the relevant DRAM, as DRAMs are most often inherent components of modern computer systems, serving as the system's RAM, i.e., the main memory unit of the system. Therefore, in theory, these DRAM modules could be power-cycled on their own, in order to gain access to the start-up values of their cells, which form the response of each DRAM startup-based PUF. Nevertheless, in practice, most often, the whole computer system that hosts the relevant DRAM needs to be power-cycled, as no hardware or firmware/software mechanism is usually implemented for power-cycling only the relevant DRAM module. Moreover, as already noted, DRAM modules have a larger storage capacity, having more memory cells, than SRAMs. Therefore, DRAM startup-based PUFs have a larger size (in terms of memory cells) than SRAM PUFs, which indicates that the responses of DRAM startup-based PUFs will generally have a higher degree of entropy than the responses of SRAM PUFs.

In general, a DRAM startup-based PUF can be easily utilised for security applications in the same way as an SRAM PUF, with a reboot again being required. For example, similarly to an SRAM PUF, a DRAM startup-based PUF can be used for the implementation of a secure key storage/generation scheme in the following way:

1. When the device reboots, the bootloader queries the relevant DRAM for its start-up values, which form the raw DRAM startup-based PUF response. In order to avoid data remanence, an adequate amount of time must have passed between the device being powered off and on.

2. Then, helper data for error correction are acquired either from the host device itself or from another device.

3. The raw PUF response and the helper data are combined, using a fuzzy extractor scheme, to generate a secret key.

4. Finally, the bootloader overwrites the DRAM cells with a specific logical pattern, which erases the raw DRAM startup-based PUF response from the DRAM module.

Again, it should be noted that the helper data and the fuzzy extractor scheme used for the implementation of a PUF-based secure key storage/generation scheme, are almost always considered as publicly available information, with the only secrets being the response of the PUF and the resulting key. It is also again important to note that for the purposes of secure key agreement, an enrollment phase would also be required, as described in the section relevant to SRAM PUFs. This PUF also allows for more advanced security applications, in a manner similar to the SRAM PUF.

In conclusion, the DRAM startup-based PUF can be considered as a PUF offering a response of higher entropy than the SRAM PUF. Nevertheless, its use in practice is limited by the fact that, likewise to the SRAM PUF, its operation requires a reboot. Additionally, its use for IoT devices may not always be feasible, depending on whether the un-initialised values of the relevant DRAM can be accessed at start-up [151].

The DRAM Decay-Based PUF

DRAM decay-based PUFs are based on the exploitation of the decay characteristics of DRAM modules, when their values are not being refreshed. In particular, a specific pattern is stored in a selection

of DRAM cells and, then, the DRAM refresh operation is stopped for a certain time period. As a number of cells will fail to maintain the values stored in them, a rather unique response can be obtained. The addresses of the cells used for the implementation of this PUF, as well as the initialisation pattern stored in them and the duration of the time period during which the DRAM refresh operation is stopped, can be considered as the challenge of this PUF, while the concatenation of the values of these cells after the DRAM refresh operation is restored can be considered as the corresponding PUF response. In practice, however, the response of this PUF is also heavily dependent on the ambient temperature, which, therefore, forms another component of its challenge, and its response is heavily biased towards the initialisation pattern, therefore, making it rather more practical to consider the positions of cells that have failed to maintain the values stored in them as the *actual* response of this PUF. In either case, the ordering of the memory addresses being used in the challenge, as well as the number of addresses being used, will affect the relevant PUF response.

Moreover, the effects of the decay of the values, i.e., the charge, stored in the relevant DRAM cells is highly dependent on the way these cells are read and, more specifically, on the way the relevant sense amplifiers lead into the charge stored in each cell being interpreted as one of the two logical values, i.e., whether any relevant effects, e.g., bit "flips" (i.e., changes in the values of the bits stored in the cells), will be noticed depends highly on the charge stored in each cell, which will cause a voltage difference between the bitline connected to that cell and its corresponding complementary bitline, and its relation to the offset voltage of the relevant sense amplifier. Figure 3.4 presents a very simplified schematic of how cells are organised in a modern DRAM, from which it is not difficult to understand why the sense amplifier circuitry, which is used to interpret the charge stored in each cell as a particular logical value, has a significant role on the effect that suspending the refresh operation of the DRAM may have on the logical value read from each cell. Therefore, the offset voltage of each relevant sense amplifier, and the voltage difference created between the bitline connected to each cell and its corresponding complementary bitline, when each cell is read, may also constitute part of the challenge of this PUF. More information on how a modern DRAM operates can be found in [152].

The DRAM decay-based PUF was one of the first attempts to address the issue of intrinsic memory-based PUFs requiring a reboot in order to operate. In particular, DRAM decay-based PUFs allowed for the first run-time implementation of a DRAM-based PUF, as their operation is based on the disabling of the refresh operation of the relevant DRAM module, which can be performed at run-time. In this case, however, it is important to keep refreshing the DRAM addresses being used by the kernel of the OS and critical applications, in order to keep the system functional and responsive and not hinder the general operation and functionality of the overall system that hosts the relevant DRAM. This can be achieved by constantly reading the relevant DRAM addresses, as reading the value of a DRAM cell also refreshes the stored value, due to the way in which the reading operation is performed [86]. Therefore, if DRAM cells are read quickly enough, the logical values stored in them will remain intact. In this way, the data decay/retention characteristics of certain DRAM memory regions can be utilised to implement a DRAM decay-based PUF, while other memory segments are still being refreshed by being repeatedly read, before their values are lost.

Nevertheless, at nominal conditions, DRAM cells tend to lose their values after a rather long time. Thus, most often, the Hamming weight of the responses of the DRAM decay-based PUF tends to rather

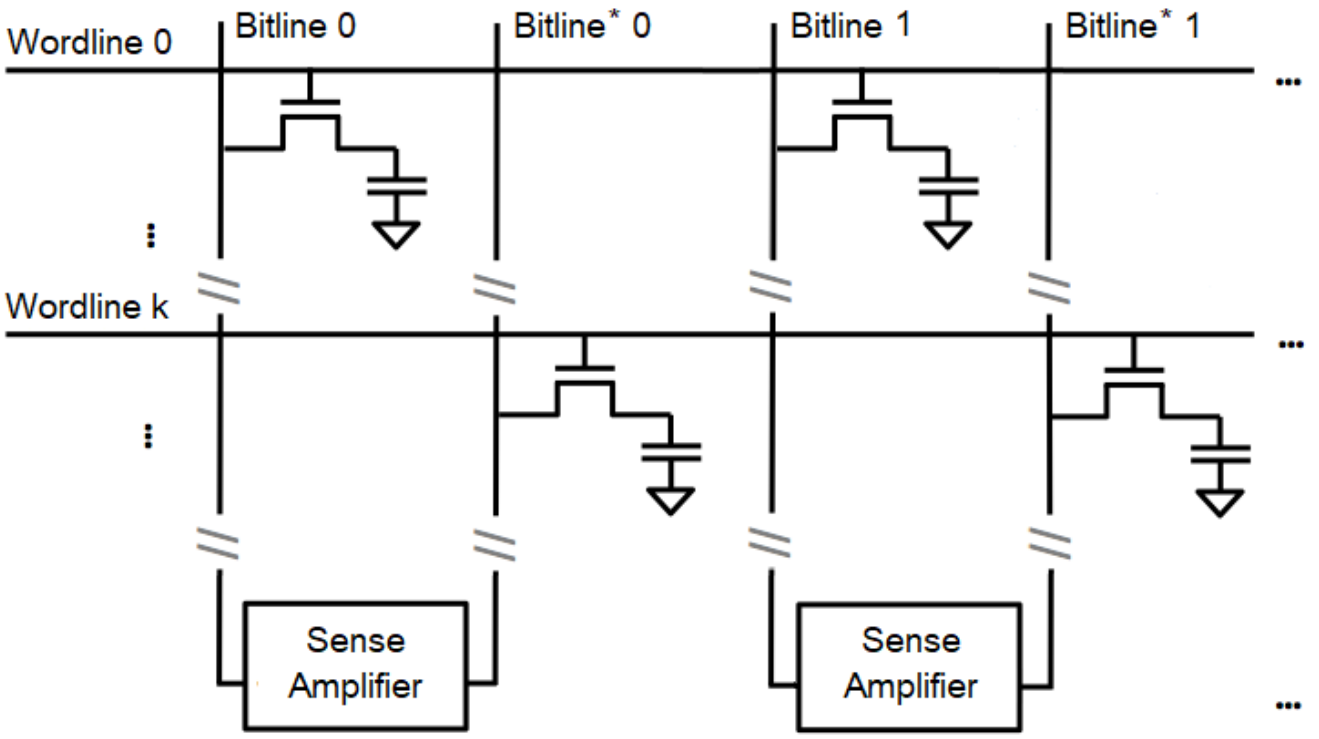


Figure 3.4: A simplified schematic of the organisation of the DRAM cells in a modern DRAM structure.

indicate the bias of the pattern stored in the memory before its refresh operation was disabled. Therefore, the randomness of the overall pattern recorded in the relevant responses cannot be considered as high. Nevertheless, the position of the DRAM cells that will decay after a certain time tends to be rather unique per DRAM instance, and the exact time at which a particular DRAM cell will decay is rather unpredictable, as it is dependent also on the temperature, the initial logical value of the cell (i.e., its initial charge), and the current leakage paths of that particular cell. Therefore, this aspect of the DRAM data decay can provide a rather high degree of entropy, and thus also randomness. In particular, in this case, an attacker would not need to correctly predict the overall content of the PUF response, which is rather similar to the initial pattern written to the memory, but rather the exact cells that have changed their value, i.e., that have "flipped" at a particular time.

Additionally, we also need to note that DRAM cells can act in two distinct ways, with `HIGH` charge signifying either the logical value of '1' or the logical value of '0'. The first category of cells are known as *true cells* and the second one as *anti-cells*. Obviously, when `LOW` charge is stored in true cells, it signifies a logical '0', and when it is stored in anti-cells, it signifies a logical '1'. Therefore, if the initial value of a true cell is a logical '0' before the DRAM refresh operation is disabled, then, with extremely high probability, the value of this cell will not have changed when the refresh operation is enabled again. The same holds true for an anti-cell whose initial value is a logical '1' before the DRAM refresh operation is disabled. Therefore, if the same value is stored in all the DRAM cells before the DRAM refresh operation is disabled, a large amount of the cells will never "flip". Counterintuitively, this fact rather increases the randomness of this PUF, as different positions in different DRAMs will never "flip" and, thus, will never reach the same end state. This increase in the entropy of this PUF, which is based on the positions (i.e., the DRAM addresses) that "flip" (i.e., the logical value stored in them changes), can be easily

understood if one considers that, even after an infinite amount of time, an attacker cannot predict that the value of a particular DRAM cell will have "flipped", as the cell may be such that the value written to it before the refresh operation is disabled corresponds to `LOW` charge being stored in it, i.e., the cell is not charged. Hence, if for a given DRAM the number of true cells is rather similar to that of anti-cells and their distribution rather random, then an attacker cannot but guess the value stored in any particular cell, even after an infinite amount of time and even if the attacker is aware of the bit pattern stored in the DRAM cells before the refresh operation of the DRAM is disabled. This makes it extremely difficult for an attacker to correctly predict the response of such a PUF, without gaining additional information regarding the exact decay/retention characteristics of each individual cell of the relevant DRAM module.

Therefore, a DRAM decay-based PUF can also provide a high level of security in the framework of cryptographic applications. For example, it can be utilised for the implementation of a secure key storage/generation scheme in the following way:

1. First of all, in order to avoid interference with the system, the relevant firmware, and other critical software applications, the DRAM cells that will be used for the implementation of the DRAM decay-based PUF need to be selected in such a way that their addresses are not used by the relevant OS, firmware, or other critical applications.

2. Then a particular pre-selected bit pattern is written to the DRAM cells serving as the DRAM decay-based PUF. This pattern can be such that all the relevant DRAM cells have a particular logical value, e.g., the values of all of them are set to be logical '0' or logical '1', or can also be a rather random pattern, or even a specific bit pattern setting all DRAM cells with an odd address to logical '0' and all with an even address to logical '1' or vice versa. The selection of the pattern should be such that the entropy of the PUF response is maximised, i.e., the maximum [21] number of DRAM cells have "flipped" during the time period that the DRAM refresh operation is suspended, while, at the same time, it still remains hard for an attacker to predict if a certain cell will "flip" or not, not only regarding the time period that the DRAM refresh operation is suspended, but also in general. Therefore, the bit pattern used should not be such that would allow all DRAM cells to "flip" after an infinite amount of time, so that an attacker will not be able to gain adequate information regarding the nature of all the DRAM cells, even if it is possible to gain access to a number of the relevant DRAM decay-based PUF responses. Therefore, it is rather advised that all DRAM cells used for the realisation of a DRAM decay-based PUF be written with the same logical value, or with a rather random bit pattern.

3. Subsequently, the DRAM refresh operation is disabled, in order to allow the charge stored in the DRAM cells to naturally "decay" through leakage. It is important to note that this charge leakage can potentially lead to nearby components, including other DRAM cells getting charged – this

[21] In case the pattern written to the DRAM cells, before the refresh operation is suspended, is known, and the majority of the cells will "flip", then the maximum entropy may correspond to a state where the maximum number of DRAM cells have *not* "flipped" during the time period that the DRAM refresh operation is suspended. In general, the maximum entropy will probably be achieved when half of the DRAM PUF cells will have "flipped" and half will not have "flipped", if an equal number of cells that can "flip" (i.e., true cells with an initial value of '1' and anti-cells with an initial value of '0') and cells that cannot "flip" (i.e., true cells with an initial value of '0' and anti-cells with an initial value of '1') is assumed. Therefore, we aim to maximise the number of "flipped" cells, when such cells are still a minority of the DRAM PUF cells, and to minimise their number when they form the majority of the DRAM PUF cells.

is essentially the physical mechanism behind the DRAM row hammer phenomenon, which we will examine in the following segment. Nevertheless, when natural charge leakage occurs in a particular DRAM cell, it is rather unlikely that this will result in another DRAM cell being charged.

4. After the DRAM refresh operation has been disabled, it is important to preserve the values of the DRAM cells being used by critical applications, the OS, and any relevant firmware. This can be achieved by manually refreshing the values of these cells, before they can fade away due to charge leakage. Due to the way in which modern DRAMs operate, reading the value of a DRAM cell results in its value being refreshed. Therefore, by reading all the relevant cells at a regular period of time less than the minimum time it takes for the weakest DRAM cell to "flip", which is usually considered to be around 32 milliseconds (ms), we can ensure that the values of critically important DRAM cells are maintained.

5. After a certain pre-selected period of time, referred to as the *decay time*, the DRAM refresh operation is restored. Thus, at this point, the manual refresh operation can stop, while the cells used for the realisation of the DRAM decay-based PUF now contain the relevant PUF response.

6. Due to the nature of the response of this PUF, it is rather preferable to use a bit selection scheme, in order to generate the secret key from the relevant PUF response [87]. In this case, the information regarding the bit selection constitutes the relevant helper data, which are acquired either from the host device itself or from another device. In general, as already discussed, if the decay time is rather small, the PUF response will be extremely biased towards the pattern written to the DRAM before the refresh operation was suspended. Therefore, and for reasons of cost, it is rather beneficial to select directly the key bits from the response, rather than try to correct potential errors in all the positions of its bits through the employment of a fuzzy extractor and relevant helper data.

7. Finally, the DRAM cells used for the realisation of the DRAM decay-based PUF will be overwritten with a specific logical pattern, which will erase the PUF response from the DRAM module. Although the same bit pattern can be used as the one written to the DRAM before the refresh operation was suspended, a different bit pattern can also be utilised, in order to disallow an attacker from gaining any information relevant to the key, as some of the bits of the key may be based on cells that did not "flip" and, thus, be identical to the values stored in them by the bit pattern written to the DRAM before the refresh operation was suspended.

It is again important to mention that the helper data used, in the context of a bit selection (or correction) scheme, for the implementation of a PUF-based secure key storage/generation scheme, are almost always considered as publicly available information, with the only secrets being the response of the PUF and the resulting key. The bit pattern used for the initialisation of the DRAM, before the refresh operation is suspended, can also be kept secret, in order to disallow an attacker from potentially gaining any information relevant to the key produced.

As already discussed in the segment regarding the SRAM PUF, for the purposes of a secure key agreement scheme, an enrollment phase would also be required. The DRAM decay-based PUF allows also for more advanced security applications in a manner similar to the start-up PUFs discussed. However, the DRAM decay-based PUF also allows for such applications at run-time, contrary to the start-up PUFs examined.

Moreover, as already mentioned, the refresh operation of a DRAM can also be stopped by powering off the device. In this case, the characteristic utilised for the realisation of the PUF would again be the decay/retention of the values of the DRAM cells, in the form of data remanence exhibited while the device was powered off. In this case, however, the device may also not be adequately grounded, as it will not be connected to a power outlet and, therefore, its data decay/retention characteristics may differ. Additionally, in this case, a reboot is required and care must be taken so that the values of the relevant DRAM cells are accessed at boot-time, before they are overwritten by the system. Nevertheless, in this case, almost all, if not all, DRAM cells can be utilised to provide the relevant DRAM PUF response, as very few, or even none, of the DRAM cells need to be reserved for the system to operate properly.

Again, in this case, the relevant PUF can provide adequate security for the implementation of cryptographic applications. Such a DRAM-based PUF can be employed for the implementation of a secure key storage/generation scheme in the following way:

1. First, a particular pre-selected bit pattern is written to the DRAM cells serving as the DRAM remanence PUF. This pattern can be such that all the relevant DRAM cells have a particular logical value, e.g., the values of all of them are set to be logical '0' or logical '1', or can also be a rather random pattern, or even a specific bit pattern setting all DRAM cells with an odd address to logical '0' and all with an even address to logical '1' or vice versa. The selection of the pattern should be such that the entropy of the PUF response is maximised, i.e., the maximum [22] number of DRAM cells have "flipped" during the time period that the DRAM refresh operation is not working due to the device being powered off, while, at the same time, it still remains hard for an attacker to predict if a certain cell will "flip" or not, not only regarding the time period that the DRAM is powered off, but also in general. Therefore, the bit pattern used should not be such that would allow all DRAM cells to "flip" after an infinite amount of time, so that an attacker will not be able to gain adequate information regarding the nature of all the DRAM cells, even if it is possible to gain access to a number of the relevant DRAM remanence PUF responses. Therefore, it is rather advised that all DRAM cells used for the realisation of a DRAM remanence PUF be written with the same logical value, or with a rather random bit pattern.

2. Subsequently, the DRAM is powered off, in order to allow the charge stored in the DRAM cells to naturally "decay" through leakage.

3. After a certain pre-selected period of time, referred to as the *power-off time*, the DRAM is powered on again. Thus, at this point, the cells used for the realisation of the DRAM remanence PUF now contain the relevant PUF response.

4. Due to the nature of the response of this PUF, it is rather preferable to use a bit selection scheme, in order to generate the secret key from the relevant PUF response. In this case, the information

[22] In case the pattern written to the DRAM cells, before the device is powered off, is known, and the majority of the cells will "flip", then the maximum entropy may correspond to a state where the maximum number of DRAM cells have *not* "flipped" during the time period that the DRAM is powered off. In general, the maximum entropy will probably be achieved when half of the DRAM PUF cells will have "flipped" and half will not have "flipped", if an equal number of cells that can "flip" (i.e., true cells with an initial value of '1' and anti-cells with an initial value of '0') and cells that cannot "flip" (i.e., true cells with an initial value of '0' and anti-cells with an initial value of '1') is assumed. Therefore, we aim to maximise the number of "flipped" cells, when such cells are still a minority of the DRAM PUF cells, and to minimise their number when they form the majority of the DRAM PUF cells.

regarding the bit selection constitutes the relevant helper data, which are acquired either from the host device itself or from another device. In general, if the power-off time is extremely small, the PUF response will be extremely biased towards the pattern written to the DRAM before it was powered off. Therefore, and for reasons of cost, it is rather beneficial to select directly the key bits from the response, rather than try to correct potential errors in all the positions of its bits through the employment of a fuzzy extractor and relevant helper data.

5. Finally, the DRAM cells used for the realisation of the DRAM remanence PUF will be overwritten with a specific logical pattern, which will erase the PUF response from the DRAM module. Although the same bit pattern can be used as the one written to the DRAM before it was powered off, a different bit pattern can also be utilised, in order to prevent an attacker from gaining any information relevant to the key, as some of the bits of the key may be based on cells that did not "flip" and, thus, be identical to the values stored in them by the bit pattern written to the DRAM before it was powered off.

It is again important to mention that the helper data used, in the context of a bit selection (or correction) scheme, for the implementation of a PUF-based secure key storage/generation scheme, are almost always considered as publicly available information, with the only secrets being the response of the PUF and the resulting key. The bit pattern used for the initialisation of the DRAM, before the device is powered off, can also be kept secret, in order to disallow an attacker from potentially gaining any information relevant to the key produced.

As already discussed in the segment regarding the SRAM PUF, for the purposes of a secure key agreement scheme, an enrollment phase would also be required. The DRAM remanence PUF allows also for more advanced security applications in a manner similar to the other PUFs already described. However,the DRAM remanence PUF does not allow for such applications at run-time, contrary to other DRAM decay-based start-up PUFs. Moreover, obviously, if an SRAM exhibits adequate data remanence, it may also be possible to implement an SRAM remanence PUF using the relevant SRAM device.

In general, the DRAM decay-based PUF can be considered as a PUF offering a response of potentially lower entropy than the startup-based PUFs. However, its use is much more practical than the use of startup-based PUFs, as some of its implementations can operate at run-time and do not require a reboot. Nevertheless, its use at run-time is dependent on the ability to manually refresh the DRAM segments used by critical applications, such as the OS kernel, etc., while the DRAM refresh operation has been disabled.

Furthermore, it is important to note here that exactly due to the nature of the responses of the DRAM decay-based PUF, which are heavily biased towards the pattern written to the DRAM before the refresh operation is suspended, regular PUF metrics cannot evaluate the quality of their characteristics correctly. To this end, as we will discuss in Section 3.2, the Jaccard index [149, 150] has been employed in order to properly consider the set of bit "flips" contained in the overall response of this PUF and assess its quality characteristics. Moreover, the responses of the DRAM decay-based PUF are also dependent on the ambient temperature, an effect that we will discuss and examine further in Section 4 and address in Section 5, by proposing a protocol that can provide robust security applications utilising this type of PUF.

In order to examine DRAM PUF designs that may truly be considered as practical, for the purposes of this work, we have employed COTS DRAM modules, which do not require the use of FPGA or ASIC designs. In this way, we are able to examine whether DRAM PUFs designs implemented in actual commercial devices, the DRAM modules of which, function as normal memory when they are not utilised as PUFs, can truly act as adequate security mechanisms in practice.

In general, DRAM PUFs can be easily implemented on almost any modern computer system, as DRAM is very often an inherent component of modern computer systems, serving as the system's RAM, i.e., its main memory unit. Although DRAMs most often can be easily accessed even by non-privileged software, the implementation of a DRAM decay-based PUF requires the usage of privileged software, in order for the DRAM refresh operation to be suspended. In this work, we have utilised DRAM decay-based PUFs implemented using the DRAM modules of two distinct COTS devices. In particular, we have implemented DRAM decay-based PUFs utilising the DRAMs of PandaBoard ES Rev B3 and Intel Galileo Gen 2 boards. Other works [86, 87, 153] have demonstrated that these two boards, shown in Figure 3.5, allow for the practical implementation of lightweight intrinsic DRAM decay-based PUFs that exhibit good PUF characteristics.

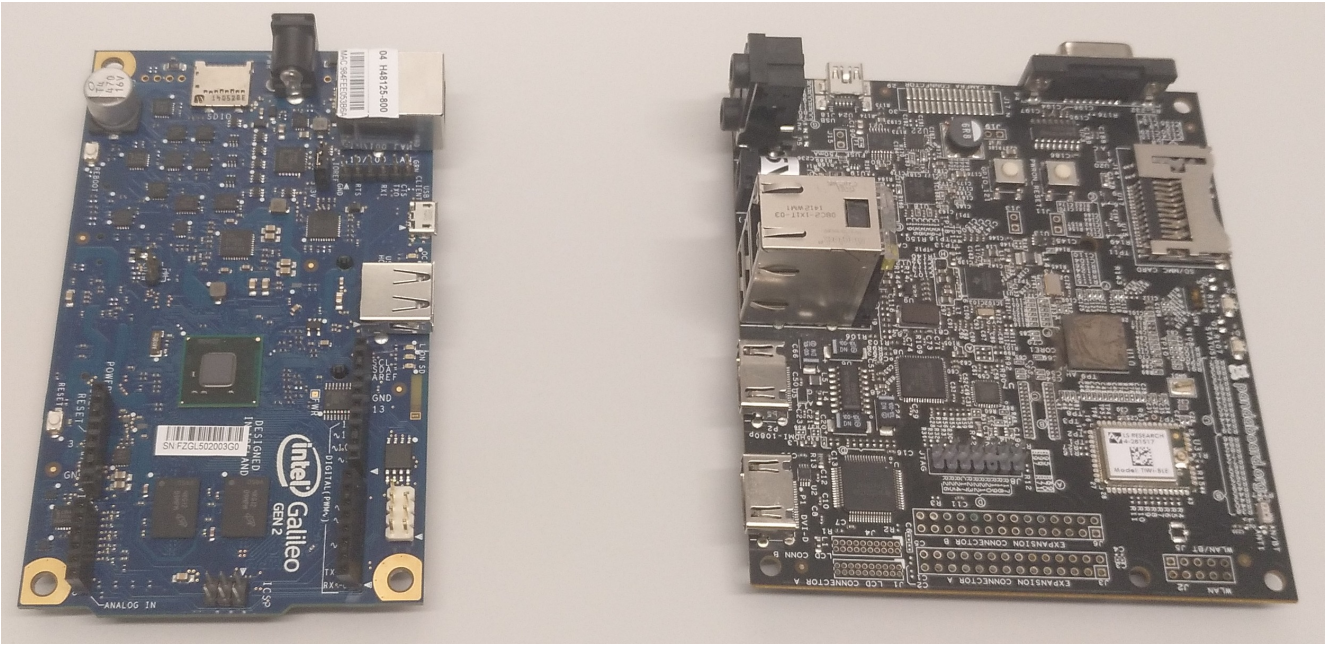


Figure 3.5: A photo of the Intel Galileo Gen 2 board (on the left), and the PandaBoard ES Rev B3 board (on the right).

As the DRAM decay-based PUF responses are highly dependent on both the ambient temperature and the decay time duration, the characteristics of these PUFs, in relation to both of these factors, have been examined in detail in previous works [83, 86, 87, 147, 153], in order to test whether such responses can truly provide reliable security. In order to validate that these PUFs can provide adequate security in the context of the security protocols and applications proposed and discussed in Section 5, we will examine in more detail the effects of adverse environmental conditions, including ambient temperature variations, in Section 4.

It is also worth noting that these PUFs are intrinsic, as they are based on DRAMs that form inherent components of the relevant COTS systems. For example, it is worth noting here that such PUFs can be

implemented even on COTS devices utilising the Android OS [154]. Additionally, the operation of these PUFs does not require the utilisation of computationally heavy software, which further enhances their ability to act as lightweight security mechanisms.

Moreover, it has been proven that these DRAM PUFs are able to generate multiple cryptographic tokens, e.g., keys, of varying sizes. Each of the DRAM PUFs examined in this work provides responses that can be easily utilised in order to produce multiple 1024-bit or even 4096-bit keys, even for a decay time of a minute or less, under nominal conditions [87].

Finally, we also note that the responses of DRAM decay-based PUFs can also be combined with the responses of other intrinsic memory-based PUFs that are implemented utilising other memories of the same system, in order to implement a PUF that is not only highly reconfigurable, but can also provide enhanced security solutions [148].

The DRAM Row Hammer PUF

The DRAM Row Hammer PUF aims to address the issue of the relatively long (decay) time that is required for the DRAM decay-based PUF to produce a response of adequate entropy. Depending on the size of the cryptographic token, e.g., a key, that needs to be produced utilising this response, the relevant decay time for a DRAM decay-based PUF may need to be in the order of tens or hundreds of seconds [148], depending on the size of the DRAM region that is utilised for the implementation of the relevant DRAM decay-based PUF and the ambient temperature during the corresponding PUF operation that leads to the measurement of the relevant PUF response.

The DRAM Row Hammer PUF utilises the DRAM row hammer effect in order to enhance the decay process of the relevant DRAM, while the DRAM refresh operation has been suspended, i.e., during the decay time. This leads into an increased number of bit "flips" in comparison to the DRAM decay-based PUF, enhancing the entropy of the responses of the DRAM Row Hammer PUF and, in this way, allowing for the faster generation of the relevant cryptographic tokens, such as keys.

Hence, DRAM Row Hammer PUFs are based on the exploitation of the decay characteristics of DRAM modules, when their values are not being refreshed and a row hammering operation is applied to them. In particular, a specific pattern is stored in a selection of DRAM cells and, then, the DRAM refresh operation is stopped for a certain time period and these cells are affected by the row hammering of cells in nearby rows. As an increased number of cells, in comparison to the DRAM decay-based PUFs, will fail to maintain the values stored in them, a rather unique response can again be obtained. The addresses of the cells used for the implementation of this PUF, the initialisation pattern stored in cells used for the PUF response (PUF cells), the row hammering pattern used to hammer cells in nearby rows, the type of row hammering being applied, the duration of the time period during which the DRAM refresh operation is stopped and nearby rows are hammered, as well as the ambient temperature and the presence or absence of any relevant caching and ECC operations that may affect the row hammering, can be considered as the challenge of this PUF, while the concatenation of the values of the PUF cells, after the DRAM refresh operation is restored and the row hammering operation is stopped, could be considered as the corresponding PUF response. In practice, however, the response of this PUF is heavily biased towards the initialisation pattern of the relevant cells, therefore, making it rather more practical to consider the positions of cells that have failed to maintain the values stored in them as the *actual*

response of this PUF. Nevertheless, in either case, the ordering of the memory addresses being used in the challenge, as well as the number of addresses being used, will affect the relevant PUF response.

Moreover, as already mentioned, the effects of the decay of the values, i.e., the charge, stored in the relevant PUF cells is highly dependent on the way these cells are read and, more specifically, on the way the relevant sense amplifiers lead into the charge stored in each cell being interpreted as one of the two logical values, i.e., whether any relevant effects, e.g., bit "flips", will be noticed depends highly on the charge stored in each cell, which will cause a voltage difference between the bitline connected to that cell and its corresponding complementary bitline, and its relation to the offset voltage of the relevant sense amplifier. Making use of Figure 3.4, which shows how cells are organised in a modern DRAM, it is not difficult to understand why the sense amplifier circuitry, which is used to interpret the charge stored in each cell as a particular logical value, has a significant role on the effects that suspending the refresh operation of the DRAM and row hammering cells in rows nearby to PUF cells may have on the logical value read from each such cell. Therefore, the offset voltage of each relevant sense amplifier, and the voltage difference created between the bitline connected to each cell and its corresponding complementary bitline, when each cell is read, may also constitute part of the challenge of this PUF. More information on how a modern DRAM operates can be found in [152].

The row hammer effect is based on the disturbance of rows of DRAM cells due to the rapidly repeated accessing (hammering) of nearby rows. In this case, disturbance errors occur due to the charge leakage of the rows that are hammered, which affects the leakage characteristics of nearby rows that are not being accessed, due to charge coupling between the DRAM cells of the rows being hammered, which can be referred to as the *hammer rows*, and the cells of nearby rows that are not being accessed while the hammer rows are being hammered, but which are affected by the row hammer effect, and which, in the context of a DRAM Row Hammer PUF can be referred to as the PUF or victim rows.

In recent years, large-scale integration and higher clock frequencies being used in DRAMs have brought into the spotlight the significance of the row hammer effect for the security of contemporary DRAM implementations [155, 156, 157]. Due to large-scale integration, DRAM components that are physically close to each other start to exhibit charge coupling effects, which, in the case of row hammering, can be exploited to "flip" the values of unaccessed DRAM cells by quickly and repeatedly accessing cells in nearby DRAM rows. High clock frequencies can amplify the effects of row hammering, as they allow for more frequent accesses to the DRAM rows that are being hammered and, thus, intensify the rate at which these rows are being activated.

In general, the row hammer effect induces errors in the values of DRAM cells when uncached DRAM rows are rapidly and repeatedly accessed, e.g., by being written, in an operation referred to as row hammering. The very frequent activation of the DRAM rows that are being hammered, in conjunction with the charge coupling between nearby DRAM components, may lead to a significant increase in the charge leakage of unaccessed cells in DRAM rows close to those being hammered, which can quickly result in a change of the logical value of these cells, if they were initially in their charged state.

Therefore, the natural decay of DRAM cells that is exploited in order to implement a DRAM decay-based PUF can be enhanced by the row hammer effect, in the context of the DRAM Row Hammer PUF, to achieve a larger number of bit "flips" in a shorter amount of time and, in this way, increase the entropy and the randomness of the relevant PUF responses. In this case, it must be ensured that

the relevant application causing the row hammering remains active during the time period at which the DRAM refresh operation has been suspended and the relevant DRAM rows are being hammered, i.e., the time period during which the PUF response is produced, and, thus, that the DRAM addresses relevant to this application keep being refreshed during this period of time and are not affected by the row hammering. Additionally, of course, the DRAM addresses being used by the kernel of the OS and critical applications need to also be refreshed during that period of time and remain unaffected by the row hammering, in order to keep the system functional and responsive and not hinder the general operation and functionality of the overall system that hosts the relevant DRAM.

As the DRAM Row Hammer PUF is essentially an improved adaptation of the DRAM decay-based PUF, the DRAM Row Hammer PUF rather has similar characteristics to the DRAM decay-based PUF. For example, for the same reasons as for the DRAM decay-based PUF, the Jaccard index [149, 150] has been employed in order to properly consider the set of bit "flips" contained in each of the responses of the DRAM Row Hammer PUF and assess its quality characteristics, as we will discuss in Section 3.2. Additionally, the responses of the DRAM Row Hammer PUF are also dependent on the ambient temperature, an effect that we will discuss and examine further in Section 4 and address in Section 5, by proposing a protocol that can provide robust security applications utilising this type of PUF. Moreover, the overall distribution, location, and behaviour of true cells and anti-cells need to be taken into account also in the case of the DRAM Row Hammer PUF, in the same way that they are considered for the implementation of the DRAM decay-based PUF.

Moreover, in the case of the DRAM Row Hammer PUF, an attacker would obviously not only need to know how the disabling of the DRAM refresh operation will affect the values stored in the DRAM cells that are relevant for the production of the PUF response, but also how the row hammer effect will also affect the values stored in these cells. To this end, we can distinguish two different cases of row hammering: single-sided and double-sided row hammering. As hammering a DRAM row will most likely affect its two adjacent rows, we can distinguish between single-sided row hammering, where one hammer row is used to affect its two adjacent rows, and double-sided row hammering, where two (hammer) rows adjacent to the same victim row are hammered, in order to increase the chance of bit "flips" [82, 143, 158]. Of course, these two hammer rows may also affect each of the two adjacent (victim) rows that are adjacent to only one, and not to both, of them.

Additionally, similar to the DRAM decay-based PUF, the pattern written to the PUF rows before the PUF operation significantly affects the number of bit "flips" that occur in the PUF rows while the DRAM refresh operation is suspended. However, in the case of the DRAM Row Hammer PUF, the pattern used for row hammering can also affect the number of bit "flips" in the PUF rows [82, 143, 158]. It should be evident, therefore, that it is extremely difficult for an attacker to correctly predict the response of such a PUF, without gaining additional information not only regarding the exact decay/retention characteristics of each individual cell of the relevant DRAM module that is being used as a PUF cell, but also regarding the way different types of row hammering would affect each such cell.

Therefore, a DRAM Row Hammer PUF can also provide a high level of security in the context of cryptographic applications. For example, it can be utilised for the implementation of a secure key storage/generation scheme in the following way:

1. First of all, in order to avoid interference with the system, the relevant firmware, and other critical software applications, the DRAM cells that will be used for the implementation of the DRAM Row Hammer PUF need to be selected in such a way that their addresses are not used by the relevant OS, firmware, or any other critical applications.

2. Then, a particular pre-selected bit pattern is written to the DRAM cells in the PUF rows (or victim rows) of the DRAM region that is reserved for the implementation of the DRAM Row Hammer PUF. This pattern can be such that all the relevant DRAM cells have a particular logical value, e.g., the values of all of them are set to be logical '0' or logical '1', or can also be a rather random pattern, or even a specific bit pattern setting all the relevant DRAM cells with an odd address to logical '0' and all with an even address to logical '1' or vice versa. Additionally, in the case of the DRAM Row Hammer PUF, a particular pre-selected bit pattern needs to be used for the hammering of the DRAM rows selected to be used as hammer rows. Again, this pattern can be such that all the relevant DRAM cells have a particular logical value, e.g., the values of all of them are set to be logical '0' or logical '1', or can also be a rather random pattern, or even a specific bit pattern setting all the relevant DRAM cells with an odd address to logical '0' and all with an even address to logical '1' or vice versa. The selection of both of these patterns should be such that the entropy of the PUF response is maximised, i.e., the maximum [23] number of DRAM cells in PUF rows have "flipped" during the time period that the DRAM refresh operation is suspended and the relevant DRAM rows are being hammered, i.e., the time period during which the PUF response is produced, while, at the same time, it still remains hard for an attacker to predict if a certain cell will "flip" or not, not only regarding the time period during which the PUF response is produced, but also in general. Therefore, the bit pattern used should not be such that would allow all DRAM cells to "flip" after an infinite amount of time, so that an attacker will not be able to gain adequate information regarding the nature of all the DRAM cells, even if it is possible to gain access to a number of the relevant DRAM Row Hammer PUF responses. It has been shown in the relevant literature [82, 143, 158] that, for the PandaBoard ES Rev B3 board used for our implementation of the DRAM Row Hammer PUF, the maximum number of bit "flips" can be achieved when the hammer rows are hammered with the bit pattern "01010101" ("`0x55`", in hexadecimal) and the PUF rows are written, before the time period during which the PUF response is produced, with the bit pattern "10101010" ("`0xAA`", in hexadecimal). Nevertheless, this combination of patterns may lead to all the DRAM cells in the PUF rows being able to "flip", which should be avoided. Therefore, a rather mixed pattern should probably be adopted, depending on the attacker model being assumed. Finally, the type of row hammering that will be applied to the relevant hammer

[23] In case the pattern written to the DRAM cells, before the refresh operation is suspended, is known, and the majority of the cells will "flip", then the maximum entropy may correspond to a state where the maximum number of DRAM cells have *not* "flipped" during the time period that the DRAM refresh operation is suspended and the relevant DRAM rows are being hammered, i.e., the time period during which the PUF response is produced. In general, the maximum entropy will probably be achieved when half of the DRAM PUF cells will have "flipped" and half will not have "flipped", if an equal number of cells that can "flip" (i.e., true cells with an initial value of '1' and anti-cells with an initial value of '0') and cells that cannot "flip" (i.e., true cells with an initial value of '0' and anti-cells with an initial value of '1') is assumed. Therefore, we aim to maximise the number of "flipped" cells, when such cells are still a minority of the DRAM PUF cells, and to minimise their number when they form the majority of the DRAM PUF cells.

rows of the DRAM needs to be selected, with double-sided row hammering allowing for fewer PUF rows, but, at the same time, yielding more bit "flips".

3. Subsequently, the DRAM refresh operation is disabled, in order to allow the charge stored in the DRAM cells to naturally "decay" through leakage, and the cells in the hammer rows start being hammered. Optimally, in order to maximise the effects of the row hammering operation, caching and any DRAM ECC functionality are disabled.

4. After the DRAM refresh operation has been disabled, it is important to preserve the values of the DRAM cells being used by critical applications, the OS, any relevant firmware, as well as those storing the row hammering application and its parameters. This can be achieved by manually refreshing the values of these cells, before they can fade away due to charge leakage. Due to the way in which modern DRAMs operate, reading the value of a DRAM cell results in its value being refreshed. Therefore, by reading all the relevant cells at a regular period of time less than the minimum time it takes for the weakest DRAM cell to "flip", which is usually considered to be around 32 ms, we can ensure that the values of critically important DRAM cells are maintained. It is important to also ensure that the values of DRAM cells storing important information will remain unaffected by the row hammering operation.

5. After a certain pre-selected period of time, referred to as the *row hammer time*, the row hammering operation is stopped and the DRAM refresh operation is restored. Thus, at this point, the manual refresh operation can stop, while the cells used for the realisation of the DRAM Row Hammer PUF, i.e., the cells of the PUF rows, now contain the relevant PUF response.

6. Due to the nature of the response of this PUF, it is rather preferable to use a bit selection scheme, in order to generate the secret key from the relevant PUF response, in a way similar to how a key can be generated from the DRAM decay-based PUF using a bit selection scheme [87]. In this case, the information regarding the bit selection constitutes the relevant helper data, which are acquired either from the host device itself or from another device. In general, if the row hammer time is rather small, the PUF response may be biased towards the pattern written to the DRAM before the refresh operation was suspended and the row hammering was started. Therefore, and for reasons of cost, it is rather beneficial to select directly the key bits from the response, rather than try to correct potential errors in all the positions of its bits through the employment of a fuzzy extractor and relevant helper data.

7. Finally, the DRAM cells used for the realisation of the DRAM Row Hammer PUF, i.e., the cells of both the hammer and the PUF rows, will be overwritten with a specific logical pattern, which will erase the PUF response and the bit pattern used for the row hammering from the DRAM module. Although the same bit patterns could be used as the one utilised for the row hammering operation, in the case of cells in hammer rows, and the one written to the cells on the PUF rows of the DRAM before the refresh operation was suspended and the row hammering operation was started, in the case of cells in PUF rows, a different bit pattern should be utilised, in order to disallow an attacker from gaining any information relevant to the key, as some of the bits of the key may be based on cells in PUF rows that did not "flip" and, thus, be identical to the values stored in them by

the bit pattern written to these DRAM rows before the refresh operation was suspended and the row hammering operation was started, and as the bit pattern used for row hammering has an effect on which cell values will "flip" in the PUF rows and, thus, also potentially on the key itself.

It is again important to mention that the helper data used, in the context of a bit selection (or correction) scheme, for the implementation of a PUF-based secure key storage/generation scheme, are almost always considered as publicly available information, with the only secrets being the response of the PUF and the resulting key. The bit patterns used for the initialisation of the PUF rows and the row hammering can also be kept secret, in order to disallow an attacker from potentially gaining any information relevant to the key produced.

As already discussed in the segment regarding the other memory-based PUFs, for the purposes of a secure key agreement scheme, an enrollment phase would also be required. The DRAM Row Hammer PUF allows also for more advanced security applications in a manner similar to the other memory-based PUFs discussed. However, like the DRAM decay-based PUF, the DRAM Row Hammer PUF also allows for such applications at run-time, contrary to the start-up PUFs examined.

In general, the DRAM Row Hammer PUF can be considered as a PUF offering a response of higher entropy than the DRAM decay-based PUFs, for the same period of time during which the PUF response is produced. Additionally, its use is much more practical than the use of startup-based PUFs, as some of its implementations can operate at run-time and do not require a reboot. Nevertheless, its use at run-time is dependent on the ability to manually refresh the DRAM segments used by critical applications, such as the OS kernel, etc., while the DRAM refresh operation has been disabled, and to keep these memory segments unaffected by the row hammering operation.

Moreover, as the responses of the DRAM Row Hammer PUF may be highly biased towards the pattern written to the PUF rows of the DRAM before the refresh operation is suspended and the row hammering operation is started, regular PUF metrics may not be able to evaluate the quality of their characteristics correctly. To this end, as we will discuss in Section 3.2, the Jaccard index [149, 150] has been employed in order to properly consider the set of bit "flips" contained in this PUF's response and assess its quality characteristics. Furthermore, the responses of the DRAM Row Hammer PUF are also dependent on the ambient temperature, an effect that we will discuss and examine further in Section 4 and address in Section 5, by proposing a protocol that can provide robust security applications utilising this type of PUF.

DRAM PUFs can, in general, be easily implemented on almost any modern computer system, as DRAM is very often an inherent component of modern computer systems, serving as the system's RAM, i.e., its main memory unit. Although DRAMs most often can be easily accessed even by non-privileged software, the implementation of a DRAM Row Hammer PUF requires the usage of privileged software, in order for the DRAM refresh operation to be suspended. Additionally, while the row hammering operation can be run by unprivileged software that has write access to the DRAM, privileged code may be needed in order to disable any relevant caching and ECC operations, in order to maximise the effects of the row hammering operation.

In order to examine DRAM PUF designs that may truly be considered as practical, for the purposes of this work, we have employed COTS DRAM modules, which do not require the use of FPGA or ASIC designs. In this way, we are able to examine whether DRAM PUFs designs implemented in actual com-

mercial devices, the DRAM modules of which, function as normal memory when they are not utilised as PUFs, can truly act as adequate security mechanisms in practice. In particular, in this work, we have utilised DRAM Row Hammer PUFs implemented using the DRAM modules of PandaBoard ES Rev B3 boards. Other works [82, 143, 158] have demonstrated that this board, which is shown on the right side of Figure 3.5, allows for the practical implementation of a lightweight intrinsic DRAM Row Hammer PUF that exhibits good PUF characteristics.

As the DRAM Row Hammer PUF responses are highly dependent on both the ambient temperature and the decay time duration, the characteristics of this PUF, in relation to both of these factors, have been examined in detail in previous works [82, 143, 158], in order to test whether such responses can truly provide reliable security. In order to validate that these PUFs can provide adequate security in the context of the security protocols and applications proposed and discussed in Section 5, we will examine in more detail the effects of adverse environmental conditions, including ambient temperature variations, in Section 4.

It is also worth noting that these PUFs are intrinsic, as they are based on DRAMs that form inherent components of the relevant COTS systems. Additionally, the operation of these PUFs does not require the utilisation of computationally heavy software, which further enhances their ability to act as lightweight security mechanisms.

Moreover, it has been proven that these DRAM PUFs are able to generate multiple cryptographic tokens, e.g., keys, of varying sizes. Each of the DRAM PUFs examined in this work provides responses that can be easily utilised in order to produce multiple 1024-bit or even 4096-bit keys, even for a decay time of a minute or less, under nominal conditions [143].

Finally, we also note that the responses of DRAM Row Hammer PUFs can also be combined with the responses of other intrinsic memory-based PUFs that are implemented utilising other memories of the same system, in order to implement a PUF that is not only highly reconfigurable, but can also provide enhanced security solutions [148].

The DRAM Latency-Based PUF

DRAM latency-based PUFs exploit the write and read latency times required by DRAMs to write logical values to DRAM cells and read them from them, respectively. By lowering either one of these latency times, or even both of them, below their nominal values, the logical values read from the cells of a DRAM module may form a rather unique bit-string, which can be utilised as the response of this PUF. In particular, reducing the write latency below its nominal value may lead into some DRAM cells not being written correctly, while reducing the read latency below its nominal value may lead into some DRAM cells not being read correctly. In either case, when the reduction of the value of either latency time starts to affect a significant number of DRAM cells, the concatenation of the logical values read out from all the DRAM cells will appear as a rather random bit-string, which will be rather unique per DRAM instance. Of course, for any effects relevant to the write latency to be noticeable, the value being written to each cell, while the write latency time has been reduced, needs to be different from the value that the cell already contained. Similarly, the effects of the reduction of the read latency time are highly dependent on the way DRAM cells are read and, more specifically, on the way the relevant sense amplifiers lead into the charge stored in each cell being interpreted as one of the two logical values, i.e., whether any relevant

effects will be noticed depends highly on the charge stored in each cell, which will cause a voltage difference between the bitline connected to that cell and its corresponding complementary bitline, and its relation to the offset voltage of the relevant sense amplifier. As already mentioned, Figure 3.4 shows a very simplified schematic of how cells are organised in a modern DRAM, from which it is easy to understand why the sense amplifier circuitry, which is used to interpret the charge stored in each cell as a particular logical value, has a significant role on the effect that reducing the read latency time of the DRAM may have on the logical value read from each cell. Therefore, not only the values of the write and read latency times of the DRAM and the addresses of the cells used for the implementation of this PUF, but also the initial logical value stored as a particular amount of charge in each relevant DRAM cell, the logical value that may be then written to each relevant DRAM cell, the offset voltage of each relevant sense amplifier, and the voltage difference created between the bitline connected to each cell and its corresponding complementary bitline, when each cell is read, constitute the challenge of this PUF. The concatenation of the values of the relevant DRAM cells used for the implementation of this PUF can be considered as the corresponding PUF response. Of course, it should be noted that also the ordering of the memory addresses being used in the challenge, as well as the number of addresses being used, will affect the result of the relevant concatenation of cell values and, therefore, the relevant PUF response.

Therefore, DRAM latency-based PUF not only does not require a reboot, but it can also provide a response within a few milliseconds or even microseconds [88, 144, 159]. However, of course, in order to be implemented, the DRAM latency-based PUF requires access to the privileged firmware code that controls the DRAM latency times. Depending on how much the latency times are reduced, the response of the DRAM latency-based PUF may be significantly biased towards a particular logical value or a specific bit pattern. For example, if only the write latency time is manipulated and is reduced well below its nominal value, the response of this PUF will be strongly biased towards the values that the relevant cells had *before* a write operation was attempted, while if only the write latency time is manipulated, but it is only slightly reduced below its nominal value, the response of this PUF will be strongly biased towards the bit pattern that was written to the relevant cells. On the other hand, particular values of the write latency time can lead into a highly random response for this PUF, containing an even number of bits with the logical value of '1' and of bits with the logical of '0', again depending on the values that the relevant DRAM cells contained before they were written to and the bit pattern that was written to them. Therefore, it is not always easy to assess whether the most proper metrics in order to evaluate the quality characteristics of the responses of this PUF would be based on the Hamming distance of its responses or on the Jaccard index of them. Nevertheless, at the same time, it should be evident that it is extremely difficult for an attacker to predict correctly the responses of this PUF, as adequate knowledge is required not only regarding the values of the relevant DRAM latency times, but also about the initial value of the relevant cells, as well as the effects that the chosen DRAM latency times will have on write and read operations that may occur with regards not only to the logical value stored in each relevant cell, but also to the actual charge stored in it.

Moreover, in contrast to the DRAM decay-based PUF and the DRAM Row Hammer PUF, the responses of the DRAM latency-based PUF do not appear to be highly dependent on the ambient temperature [88, 144], thereby providing this PUF type's responses with increased stability and robustness. Thus, the DRAM latency-based PUF can provide a high level of security in the context of cryptographic

applications. For example, it can be utilised for the implementation of a secure key storage/generation scheme in the following way:

1. First of all, it needs to be decided if only the write latency time, only the read latency time, or both latency times of the DRAM will be decreased and how much. It should also be rather obvious that while either the write latency, or the read latency, or both of them, are reduced, the DRAM cannot be used correctly. Therefore, the system, as well as any other firmware and applications, will need to halt their operation, as far as this is dependent on correctly accessing the DRAM while this PUF is producing its response. However, in most systems, this should not be a problem, as this PUF type requires only a few milliseconds, or even microseconds, in order to produce its response. Additionally, as the refresh operation of the DRAM is not disrupted, there is no need to manually maintain the values of the DRAM cells or to attempt to recover them. In either case, however, the DRAM cells selected for the implementation of this PUF should not contain values that are relevant to the system or any other active application or firmware.

2. After the appropriate set of cells has been selected, the relevant DRAM latency times are reduced. In case the write latency is reduced, a particular bit pattern needs to be selected in order to then be written to the relevant DRAM cells. In any case, the values of the relevant DRAM cells before the DRAM latency times are reduced need to be known. In the best case, a particular pre-selected bit pattern can be written to the relevant cells before the DRAM latency times are reduced, such that the entropy of the response of the DRAM decay-based PUF is maximised. Again, depending on whether the write latency time is reduced and on how much it is reduced, a new attempt to write a value to a particular cell may or may not be successful. In case this value is the same as the one that the cell had before this attempt was made, the cell will obviously keep having that value – this operation could be utilised to potentially mislead an attacker. In case, however, this value is different from the value that the cell had before this attempt, the new write operation may or may not be successful depending on both the value of the write latency time and the actual charge stored in that cell. Therefore, a new value may be written to that cell after a number of attempted write operations, i.e., a cell may fail to be written for a certain number of times. Therefore, the number of times a write operation is attempted could also form part of the challenge for this PUF. Thus, if the write latency time is reduced, it should be attempted a particular number of times to write a specific pre-selected bit pattern, or even a number of different bit patterns, to the relevant DRAM cells. Notwithstanding whether the values of the relevant cells have been changed, if the read latency time is reduced, their values may or may not be read correctly, again resulting potentially in a rather random (and unique per DRAM) bit-string that forms the response of this PUF. Nevertheless, even if only the write latency time has been reduced, the response of the DRAM latency-based PUF will need to be acquired by a read operation performed on the relevant DRAM cells. In this case, its quality characteristics will be based only on the reduction of the value of the write latency, the initial charge stored in the relevant cells before the write latency time was reduced, the number of times a write operation was attempted on these cells, and the bit patterns utilised for these write operation attempts. Finally, it is worth mentioning here that, as the refresh operation of the DRAM has not been disabled, it could potentially interfere with the operation of the DRAM latency-based PUF by changing the amount of charge –but not the logical value – stored

in the relevant cells. Therefore, this parameter needs to also be appropriately considered and could potentially form part of the challenge of this PUF, especially with regards to the number of write operations that are attempted in the case where the write latency time has been reduced and how quickly these can be performed.

3. As soon as the response of the DRAM latency-based PUF is acquired, the DRAM latency times that have been reduced should be set back to their nominal values, in order to allow for the system and its applications to continue functioning normally.

4. Regarding the generation of a secure key, we need to distinguish two cases for this PUF, depending on how biased its responses are and towards which bit patterns. If the response of this PUF is significantly biased towards the initial values that the relevant cells had before the relevant DRAM latency times were reduced or towards the bit pattern(s) used for the attempted write operations, in case the write latency time has been reduced, then, it would make sense to use a bit selection scheme to generate the secret key from the relevant PUF response in order to utilise cells that can provide enough entropy and randomness, even if an attacker has gained some knowledge on the initial values of the relevant cells and/or the bit pattern(s) used for the operation of this PUF. In this case, the information regarding the bit selection constitutes the relevant helper data, which are acquired either from the host device itself or from another device. If the response of this PUF is rather random and unbiased, then, a fuzzy extractor scheme can be utilised to generate the secret key. In this case, the relevant helper data that will be used for error correction will also need to be acquired either from the host device itself or from another device. Additionally, even if the response of this PUF is rather unbiased towards the initial values of the relevant cells and the aforementioned bit patters, but it is not very random, being rather biased towards one of the two logical values, then, again, it is rather sensible to use a bit selection scheme to generate the secret key from the relevant PUF response, as, for reasons of cost, it is rather beneficial to select directly the key bits from the response, rather than try to correct potential errors in the overall response through the employment of a fuzzy extractor and relevant helper data. In this case, again, the information regarding the bit selection constitutes the relevant helper data, which are acquired either from the host device itself or from another device.

5. Finally, the DRAM cells used for the realisation of the DRAM latency-based PUF should be overwritten with a specific bit pattern, which will erase the PUF response from the DRAM module, and which should ideally be different from the initial values that these cells had before the DRAM latency times were reduced and from the bit pattern(s) that were possibly used for potential attempted write operations on the relevant DRAM cells, in order to prevent an attacker from gaining any knowledge on these sets of bit values, i.e., any information that may be relevant to the generated key, as some of the bits of the key may be based on these values.

It is once again important to mention that the helper data used, in the context of a bit selection (or correction) scheme, for the implementation of a PUF-based secure key storage/generation scheme, are almost always considered as publicly available information, with the only secrets being the response of the PUF and the resulting key. The initial values that these cells had before the DRAM latency times were reduced and the bit pattern(s) that were possibly used for potential attempted write operations on the

relevant DRAM cells can also be kept secret, in order to disallow an attacker from potentially gaining any information relevant to the key produced.

As already discussed in the segment regarding the other memory-based PUFs, for the purposes of a secure key agreement scheme, an enrollment phase would also be required. The DRAM latency-based PUF allows also for more advanced security applications in a manner similar to the other memory-based PUFs discussed. However, like the DRAM decay-based PUF and the DRAM Row Hammer PUF, the DRAM latency-based PUF also allows for such applications at run-time, contrary to the start-up PUFs examined.

In general, the DRAM latency-based PUF offers a response of adjustable entropy within a relatively short amount of time. Its use is very practical, as most, if not all, of its implementations can operate at run-time and do not require a reboot. Nevertheless, its use at run-time is dependent on the capability of the system and critical applications to be halted, or at least to be disallowed of access to the relevant DRAM module, during the time period required for the operation of this PUF. Additionally, in contrast to the DRAM decay-based PUF and the DRAM Row Hammer PUF, the responses of the DRAM latency-based PUF do not appear to be highly dependent on the ambient temperature [88, 144], thereby providing responses of high stability and robustness.

Moreover, as the responses of the DRAM latency-based PUF may or may not be biased towards particular patterns and/or one of the logical values, depending on which DRAM latency times have been reduced and by how much, on the initial values that the relevant cells have before these DRAM latency times are reduced, and on the bit pattern(s) that were possibly used for potential attempted write operations on these DRAM cells, the most proper metrics in order to evaluate the quality characteristics of the responses of this PUF have to be decided accordingly, with metrics based on the Hamming distance of the relevant responses being used when the responses of this PUF will be used in the context of a fuzzy extractor scheme, and with metrics based on the Jaccard index of such responses being used when the relevant responses will be utilised to produce a cryptographic token, e.g., a key, through a bit selection scheme.

DRAM PUFs can, in general, be easily implemented on almost any modern computer system, as DRAM is very often an inherent component of modern computer systems, serving as the system's RAM, i.e., its main memory unit. Although DRAMs most often can be easily accessed even by non-privileged software, the implementation of a DRAM latency-based PUF requires the usage of privileged code in order to access and modify the DRAM firmware that controls the DRAM latency times. Writing to and reading from the DRAM can, of course, be performed by unprivileged application code.

Likewise to the rest of the memory-based PUFs presented in this work, the DRAM latency-based PUF can also be implemented in COTS DRAM modules, which do not require the use of FPGA or ASIC designs [88, 144, 159], and can, therefore, truly be considered as practical DRAM PUFs. These PUFs have been implemented on actual commercial devices, the DRAM modules of which, function as normal memory when they are not utilised as PUFs, and, thus, can truly act as adequate security mechanisms in practice. It is also worth noting that these PUFs are intrinsic, as they are based on DRAMs that form inherent components of the relevant COTS systems. Additionally, the operation of these PUFs does not require the utilisation of computationally heavy software, which further enhances their ability to act as lightweight security mechanisms.

Moreover, as the DRAM latency-based PUF can provide responses of adjustable entropy, multiple cryptographic tokens, e.g., keys, of varying sizes can be generated from each instance of this PUF. These responses can, then, also be combined with the responses of other intrinsic memory-based PUFs that are implemented utilising other memories of the same system, in order to implement a PUF that is not only highly reconfigurable, but can also provide enhanced security solutions [148].

3.1.3 Flash-Memory-Based PUFs

Flash-memory-based PUFs are based on the disturbance of cells of Flash memory cells due to rapidly repeated operations, e.g., programming, reading, etc., of nearby cells [70]. As Flash memory is organised in pages, the aforementioned operations are performed at the level of a page in order to disturb the cells of nearby pages. In particular, by repeatedly programming a Flash memory page, after a whole block of pages has been erased, or by repeatedly reading from such a page, nearby cells can be affected resulting in changes in their logical values, i.e., in bit "flips". In our work, we focus on Flash-memory-based PUFs implemented on NAND Flash memories using programming disturbances (and, thus, we will proceed to examine in more detail only this type of Flash-memory-based PUF), as this method has been proven to be effective in order to construct Flash-memory-based PUFs in COTS devices, and as Flash-memory-based PUFs implemented on NAND Flash memories using reading disturbances have been shown to provide responses of lower entropy, as they do not create as many disturbances as Flash-memory-based PUFs implemented on NAND Flash memories using programming disturbances do, for the same period of time required in order to produce the relevant responses. In general, it has also been noted that NAND Flash memories are cheaper than NOR Flash memories, with NAND Flash memories being used for data storage, and NOR ones for code storage and execution [160].

As Flash memories tend to also be inherently present in most systems that utilise a non-volatile memory, Flash-memory-based PUFs most often constitute intrinsic PUFs. Additionally, their operation does not require the utilisation of computationally heavy software, which further enhances their ability to act as lightweight security mechanisms. Moreover, depending on the size of the Flash memory, Flash-memory-based PUFs can be used to generate multiple cryptographic tokens, e.g., keys, of varying sizes. In general, as Flash memories tend to have a larger storage capacity than SRAMs, but a smaller one than DRAMs, Flash-memory-based PUFs also tend to have a larger size than SRAM-based ones and a smaller one than DRAM-based ones. This can potentially result in the responses of Flash-memory-based PUFs having, in total, a higher degree of entropy than the responses of SRAM PUFs and a lower one than the responses of DRAM PUFs. We also need to note that Flash memories tend to have a relatively large size, a property that makes their use as PUFs quite flexible, as different memory segments of varying sizes can be utilised each time as a PUF, depending on the size and amount of cryptographic tokens, e.g., keys, that need to be generated. In particular, each of the Flash-memory-based PUFs examined in this work provides several KB as its response, which can be easily utilised in order to produce multiple 1024-bit or even 4096-bit keys.

The Flash-memory-based PUF examined in this work essentially employs the relevant disturbances caused to unaccessed pages by the programming of nearby Flash memory pages, in a manner rather similar to the way in which the DRAM Row Hammer PUF utilises the row hammer effect, where unaccessed rows are affected by the rapidly repeated writing operation that is applied to nearby rows. Therefore,

in a fashion that resembles the DRAM Row Hammer PUF, the addresses of the pages (of NAND Flash memory cells) used for the implementation of this PUF, the initialisation pattern stored in the cells of the pages used for the PUF response (PUF cells and PUF pages, respectively), the pattern used to program cells in nearby pages in order to disturb PUF pages, the type of program "hammering" being applied, the duration of the time period during which the relevant Flash memory pages are being constantly programmed, i.e., the number of programming rounds that take place, as well as the presence or absence of any relevant caching and ECC operations that may affect the programming of Flash pages, can be considered as the challenge of this PUF, while the concatenation of the values of the PUF cells, after the relevant programming operation of the nearby pages is stopped, could be considered as the corresponding PUF response. In general, therefore, this Flash-memory-based PUF may be able to produce multiple CRPs, much like the DRAM decay-based PUF, the DRAM Row Hammer PUF, and the DRAM latency-based PUF, and in contrast to the SRAM PUF and the DRAM startup-based PUF, which are known to produce only a single such pair. However, if the response of this Flash-memory-based PUF is biased towards the initialisation pattern of the relevant cells, it may be more practical to consider the positions of cells that have failed to maintain the values stored in them as the actual response of this PUF. Nevertheless, in practice, if an adequate number of programming rounds has been applied to nearby pages, there are enough bit "flips" in the PUF pages of the Flash memory and, thus, the concatenation of the values of all the PUF cells can be considered as the *true* response of this PUF. In either case, the ordering of the memory addresses being used in the challenge, as well as the number of addresses being used, will also affect the relevant PUF response.

In general, we need to also note that this PUF requires the erasure of a block of Flash memory in order to operate correctly. In NAND Flash memory, the erasure of a block of pages, sets the values of all cells in these pages to the logical value of '1'. Additionally, in this type of memory, after a cell has been programmed to the logical value of '0', the whole relevant block of pages needs to be erased, in order to change the value of this cell (and of all the other cells in this page block) to a logical '1'. Therefore, the bit patterns that can be used to repeatedly program the pages of a block of NAND Flash memory in order to affect other pages of that block are rather limited. Moreover, while the operation of this PUF does not require a reboot, the relevant block(s) of Flash memory used in the implementation of this Flash-memory-based PUF, if not even the whole Flash memory, cannot be used by the system or any applications.

Despite these limitations, a Flash-memory-based PUFs implemented on a NAND Flash memory using programming disturbances can provide a high level of security and allow for the implementation of practical cryptographic applications. For example, it can be used to realise a secure key storage/generation scheme in the following way:

1. First of all, in order to avoid interference with the system, the relevant firmware, and other critical software applications, the Flash memory block(s) that will be used for the implementation of this PUF need to be selected in such a way that the addresses of the Flash memory cells contained in them are not used by the relevant OS, firmware, or any other critical applications. It is also important to ensure that the values of Flash memory cells storing important information will remain unaffected by the overall operation of this PUF, e.g., the rapidly repeated programming of the relevant Flash pages.

2. Then, the selected Flash memory block(s) need to be erased, so that the Flash-memory-based PUFs operation can be started. A particular pre-selected bit pattern can be written to the cells of the pages that will be utilised for the PUF response (PUF cells and PUF pages, respectively). If no pattern is written, then, all of these cells will have the logical value of '1'. Additionally, and more importantly, a particular pre-selected bit pattern needs to be used for the rapidly repeated programming of the cells of the pages nearby to the PUF pages, which will be used in order to disturb the PUF cells. If more than one bit patterns are used for the aforementioned rapidly repeated programming of the relevant pages, then the limitations stated previously also need to be taken into account, as a cell's logical value cannot be programmed from being '0' into being '1' without the whole relevant block of Flash memory pages being erased. The selection of the bit pattern that is written to the PUF cells and the bit pattern(s) that are used for the rapidly repeated programming should be such that the entropy of the PUF response is maximised, i.e., the maximum [24] number of Flash memory cells in PUF pages have "flipped" during the time period that the programming disturbances are occurring, i.e., the time period during which the PUF response is produced, while, at the same time, it still remains hard for an attacker to predict if a certain cell will "flip" or not, not only regarding the time period during which the PUF response is produced, but also in general. Therefore, the bit patterns used should not be such that would allow all PUF cells to "flip" after an infinite amount of time, so that an attacker will not be able to gain adequate information regarding the nature of all the PUF cells, even if it is possible to gain access to a number of the relevant Flash-memory-based PUFs responses. Finally, the type of program "hammering", i.e., the rapidly repeated programming, that will be applied to the relevant pages of the Flash memory needs to be selected. As program "hammering" a Flash memory page will most likely affect its two adjacent pages, we can distinguish between single-sided program "hammering", where one Flash memory page, which is rapidly and repeatedly programmed, is used to affect its two adjacent pages, and double-sided program "hammering", where two pages that are adjacent to the same PUF page are rapidly and repeatedly programmed, in order to increase the chance of bit "flips". Of course, these two pages may also affect each of the two adjacent (PUF) pages that are adjacent to only one, and not to both, of them. Of course, double-sided program "hammering" will allow for fewer PUF pages per Flash memory block, but, at the same time, it will yield more bit "flips".

3. Subsequently, the cells in the relevant Flash memory pages start being rapidly and repeatedly programmed with the relevant bit patterns. Optimally, in order to maximise the effects of the programming operation, any relevant caching and ECC operations have been disabled.

4. After a certain pre-selected period of time, which, for example, could be measured in rounds of programming of the relevant Flash memory block(s), the rapidly repeated programming operation

[24] In case the pattern written to the PUF cells, before the rapidly repeated programming of the other relevant cells, is known, and the majority of the cells will "flip", then the maximum entropy may correspond to a state where the maximum number of PUF cells have *not* "flipped" during the time period that the programming disturbances are occurring, i.e., the time period during which the PUF response is produced. In general, the maximum entropy will probably be achieved when half of the PUF cells will have "flipped" and half will not have "flipped", if an equal number of cells that can "flip" and cells that cannot "flip" is assumed. Therefore, we aim to maximise the number of "flipped" cells, when such cells are still a minority of the PUF cells, and to minimise their number when they form the majority of the PUF cells.

is stopped. Thus, at this point, the cells used for the realisation of this Flash-memory-based PUFs, i.e., the cells of the PUF pages, now contain the relevant PUF response.

5. Regarding the generation of a secure key, we need to distinguish two cases for this PUF, depending on how biased its responses are towards the bit pattern that is written to the PUF cells after they have been erased. If the response of this PUF is significantly biased towards the values that the PUF cells have before the program "hammering" occurs, then, it would make sense to use a bit selection scheme to generate the secret key from the relevant PUF response in order to utilise cells that can provide enough entropy and randomness, even if an attacker has gained some knowledge on the bit pattern that is written to the PUF cells after they have been erased. In this case, the information regarding the bit selection constitutes the relevant helper data, which are acquired either from the host device itself or from another device. If the response of this PUF is rather random and unbiased, then, a fuzzy extractor scheme can be utilised to generate the secret key. In this case, the relevant helper data that will be used for error correction will also need to be acquired either from the host device itself or from another device. Additionally, even if the response of this PUF is rather unbiased towards the bit pattern that is written to the PUF cells after they have been erased, but it is not very random, being rather biased towards one of the two logical values, then, again, it is rather sensible to use a bit selection scheme to generate the secret key from the relevant PUF response, as, for reasons of cost, it is rather beneficial to select directly the key bits from the response, rather than try to correct potential errors in the overall response through the employment of a fuzzy extractor and relevant helper data. In this case, again, the information regarding the bit selection constitutes the relevant helper data, which are acquired either from the host device itself or from another device.

6. Finally, the cells used for the realisation of this Flash-memory-based PUFs, i.e., the cells of both the PUF pages and the pages that were rapidly and repeatedly programmed, will be overwritten with a specific logical pattern, which will erase the PUF response and the bit pattern(s) used for the rapidly repeated programming from the Flash memory. Although the same bit patterns could be used as one of those utilised for the programming operation, in the case of cells that were rapidly and repeatedly programmed, and the one written to the cells on the PUF pages of the Flash memory after they were erased, in the case of PUF cells, a different bit pattern should be utilised, in order to disallow an attacker from gaining any information relevant to the key, as some of the bits of the key may be based on cells in PUF pages that did not "flip" and, thus, be identical to the values stored in them by the bit pattern written to these cells after they were erased, and as the bit pattern used for the rapidly repeated programming also has an effect on which cells will "flip" in the PUF pages and, thus, also potentially on the key itself.

It is once again important to mention that the helper data used, in the context of a bit selection (or correction) scheme, for the implementation of a PUF-based secure key storage/generation scheme, are almost always considered as publicly available information, with the only secrets being the response of the PUF and the resulting key. The bit patterns used for the initialisation of the PUF pages and the rapidly repeated programming can also be kept secret, in order to disallow an attacker from potentially gaining any information relevant to the key produced.

As already discussed in the segment regarding the other memory-based PUFs, for the purposes of a secure key agreement scheme, an enrollment phase would also be required. The Flash-memory-based PUFs implemented on a NAND Flash memory using programming disturbances can also allow for more advanced security applications in a manner similar to the other memory-based PUFs discussed. Additionally, like the DRAM decay-based PUF, the DRAM Row Hammer PUF, and the DRAM latency-based PUF, this Flash-memory-based PUF also allows for such applications at run-time, contrary to the start-up PUFs previously examined. Nevertheless, depending on the number of blocks utilised for the implementation of this PUF, the operation of the system and other critical applications could potentially be affected. In either case, however, the responses of this Flash-memory-based PUFs do not appear to be highly dependent on the ambient temperature [161, 162], thereby providing responses of high stability and robustness.

Moreover, as the responses of the examined Flash-memory-based PUF may or may not be biased towards particular patterns and/or one of the logical values, depending on which bit pattern has been written to the PUF cells after they have been erased, on which bit pattern(s) have been used for the rapidly repeated programming of the relevant Flash memory cells, and on how many programming rounds have taken place, the most proper metrics in order to evaluate the quality characteristics of the responses of this PUF have to be decided accordingly, with metrics based on the Hamming distance of the relevant responses being used when the responses of this PUF will be used in the context of a fuzzy extractor scheme, and with metrics based on the Jaccard index of such responses being used when the relevant responses will be utilised to produce a cryptographic token, e.g., a key, through a bit selection scheme.

In general, a Flash-memory-based PUF allows for the implementation of a PUF in systems whose main memory is not based on DRAM, which is a volatile memory, but rather on non-volatile memory, i.e., Flash memory. Such systems are rather common for particular use cases, e.g., in the case of embedded systems being utilised in the IoT. Additionally, the implementation of this PUF does not require the use of privileged code, as erasing, programming and reading the Flash memory cells can, of course, be performed by unprivileged application code.

Likewise to the rest of the memory-based PUFs presented in this work, the Flash-memory-based PUFs implemented on a NAND Flash memory using programming disturbances can also be realised using COTS components [70, 161]. However, most often, the experimental setup presented in the relevant literature requires the use of an FPGA or an ASIC, in order to access the COTS Flash memory module. In this work, we consider the use of COTS NAND Flash memory found inherently on relevant COTS evaluation devices, which do not require the utilisation of FPGA or ASIC designs, for the implementation of Flash-memory-based PUFs, and can, therefore, truly be considered as Flash-memory-based PUFs. As these PUFs have been implemented on actual commercial devices, the Flash memory modules of which, function as normal memory when they are not utilised as PUFs, they can truly act as adequate security mechanisms in practice. It is also worth noting that these PUFs are intrinsic, as they are based on Flash memories that form inherent components of the relevant COTS systems. Additionally, the operation of these PUFs does not require the utilisation of computationally heavy software, which further enhances their ability to act as lightweight security mechanisms.

Furthermore, external Flash memory can also be used for the implementation of a Flash-memory-based PUF in order to avoid disturbing the operation of the system, as internal Flash memory may be used as the main memory of the system. Additionally, such an external Flash memory module can be utilised for the flexible implementation of an Advanced Reconfigurable PUF (AR-PUF), as we will discuss in Section 5. Nevertheless, the utilisation of an external Flash memory as a PUF cannot, in general, be considered as truly cost-efficient and, therefore, practical, unless the relevant Flash memory forms an inherent component of the overall system. For this reason, in this work, we have considered Flash-memory-based PUF designs based both on internal and external Flash memory modules. In particular, we have implemented intrinsic Flash-memory-based PUFs based on programming disturbances utilising the NAND Flash memories of ST Microelectronics STM32F407 Discovery (STM32F407G-DISC1) and ST Microelectronics STM32 Nucleo-64 NUCLEO-L152RE boards, while a Waveshare NandFlash Board (A), which is a removable external Flash memory module, has also been utilised for the implementation of this type of PUF, being controlled using an ST Microelectronics STM32F429I Discovery (STM32F429I-DISC1) board, to which the Flash board has been connected using a Waveshare Open429Z-D Standard, an STM32F4 expansion/development board for the ST Microelectronics STM32F429I Discovery (STM32F429I-DISC1) board.

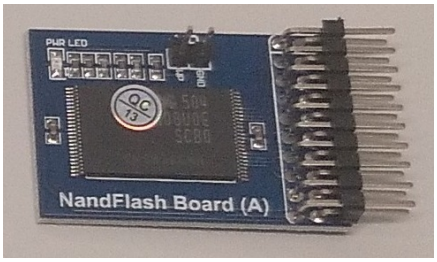


Figure 3.6: A photo of the Waveshare NandFlash Board (A).

Regarding the ST Microelectronics STM32F407 Discovery (STM32F407G-DISC1) and the ST Microelectronics STM32 Nucleo-64 NUCLEO-L152RE boards, which are shown in Figure 3.3, we have implemented Flash-memory-based PUFs based on programming disturbances using their intrinsic NAND Flash memories, in order to test the resilience of such PUF to radiation [147]. Regarding the Waveshare NandFlash Board (A), which is shown in Figure 3.6, we have implemented an external Flash-memory-based PUFs based on programming disturbances, and tested its resilience to ambient temperature and supply voltage variations. By testing the resilience of Flash-memory-based PUFs implemented on NAND Flash memories using programming disturbances, to ambient temperature and supply voltage variations, and to radiation, we were able to assess whether such PUFs can be utilised in IoT applications that potentially involve adverse operating conditions, e.g., in space missions and other relevant applications. The quality of the responses of the aforementioned Flash-memory-based PUFs under the adverse conditions described is examined in detail in Section 4, while some of their potential security applications in the context of the IoT are discussed in Section 5, where relevant cryptographic protocols based on these PUFs are proposed and examined.

Finally, we note that, as the examined Flash-memory-based PUFs can provide responses of adjustable entropy depending on the rounds of rapidly repeated programming being applied to the relevant cells of its pages, multiple cryptographic tokens, e.g., keys, of varying sizes can be generated from each instance

of this PUF. These responses can, then, also be combined with the responses of other intrinsic memory-based PUFs that are implemented utilising other memories of the same system, in order to implement a PUF that is not only highly reconfigurable, but can also provide enhanced security solutions [148].

3.2 Quality Metrics for the Responses of the Memory-Based PUF Implementations Presented

We note that all of the memory-based PUFs examined in the previous subsection can, in general, be considered as lightweight, cost-efficient, flexible, and practical security solutions for the IoT. However, the quality of their responses needs to be individually assessed, as it depends on the particular characteristics of each such PUF. To this end, a number of relevant metrics have been proposed over the years in order to assess the quality of the responses of memory-based PUFs. However, this variety of metrics being employed in order to evaluate the security of different memory-based PUFs may significantly hinder the ability to easily and adequately compare and assess these PUFs. This variety stems from the novelty of these implementations and their unique characteristics.

For example, for DRAM-based PUFs, not only conventional metrics, such as the Hamming weight [74, 75, 79, 85, 86, 87, 130, 141, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173], the intra-device and inter-device Hamming distances [79, 80, 85, 86, 130, 141, 163, 164, 165, 167, 172, 173, 174], the Shannon entropy [74, 86, 87, 166, 172, 175], and the min-entropy [74, 80, 85, 141, 165, 167], have been extensively employed, but also a number of less well-known metrics, such as the intra-device and inter-device Jaccard indices [82, 86, 87, 88, 143, 166, 175], the Sokal–Michener (simple matching) coefficient [154], the overlap distance [74, 75, 168, 169, 170, 171], as well as other quantitative and statistical performance metrics [154]. An overview of metrics for binary data, such as distances and similarities between binary strings, which can potentially be utilised in order to assess and evaluate the quality of the responses of memory-based PUFs, in order to estimate the level of security that such PUFs can provide, can be found in [176].

Moreover, a number of other metrics, such as the number of stable cells [48, 79, 85, 141, 169, 170, 171, 173, 174], the probability of uniqueness [75, 168, 169, 170, 171], the chance of mismatch [175], the bit error rate [88, 166], and the number of CRPs or keys being produced [48, 76, 82, 85, 86, 87, 88, 130, 141, 163, 166, 168, 169, 170, 171, 173, 175, 177], have also been employed in order to assess the security that a DRAM-based PUF can provide.

Finally, we also need to mention the National Institute of Standards and Technology (NIST) statistical suite of tests, which serves as a sui generis suite of metrics for DRAM-based PUFs and implementations of True Random Number Generators (TRNGs) that are based on these PUFs [79, 141, 172, 173], as well as the generation time as a metric to assess security [86, 88, 163, 173]. Moreover, the correlation among the responses of the Flash-memory-based PUF has also been used as a metric in order to assess their quality characteristics [70, 89]. It is therefore clear that it is currently not easy to compare different memory-based PUFs and their implementations, as a common single set of metrics has not been employed in the relevant literature.

Nevertheless, in this work, we will focus upon, examine in detail, and utilise only the most well-known of these metrics, in order to provide an easy way to assess the quality of the responses of the PUFs examined in this work. In particular, we will discuss the following metrics: the Hamming weight,

the binary entropy, the intra-device and inter-device Hamming distances, and the intra-device and inter-device Jaccard indices.

3.2.1 Hamming Weight

In order to estimate the entropy and the randomness of a PUF response in an extremely rough way, most often the Hamming weight is employed as a metric. The Hamming weight, in general, is equal to the number of positions in a string that have a value other than '0'. Therefore, for a binary PUF response, the Hamming weight will be equal to the number of bit positions that will have a logical value of '1'. We need to note here that the metrics examined in this work may correspond to different sets of bits and bit characteristics, depending on the memory-based PUF they are used for. For example, the Hamming weight of a DRAM decay-based PUF whose cells were initially all set to a logical value of '0' will correspond to the number of bit "flips" that have occurred during the operation of this PUF, while the Hamming weight of a DRAM startup-based PUF will correspond to the number of cells whose start-up value is equal to a logical '1'.

In general, quite often, the *fractional* Hamming weight, i.e., the number of bit positions that have a logical value of '1' is divided by the total number of bit positions, in order to estimate the percentage of each PUF response that has one logical value or the other. An average of multiple fractional Hamming weight measurements taken for a particular PUF response will, of course, provide an estimation of the probability of a bit of the response to have the logical value of '1'. The ideal value for the fractional Hamming weight metric would be 0.5 revealing an equal number of cells with the logical value of '1' and of cells with the logical value of '0' in the relevant PUF response.

3.2.2 Binary Entropy

It can easily, therefore, be concluded that the reason why the Hamming weight can provide a rough estimation of the entropy and the randomness of a PUF response is that the relevant binary (Shannon) entropy is heavily dependent on the probability that each logical value has on appearing in the relevant PUF response. In particular, the relevant binary entropy function is as follows:

$$H_b(X) \;=\; -\,Pr[X_i=1]\;\; log_2(Pr[X_i=1]) \;-\; Pr[X_i=0]\;\; log_2(Pr[X_i=0])\;\;, \tag{3.1}$$

where $Pr[X_i=1]$ is the probability of a bit of the response to have the logical value of '1', and $Pr[X_i=0]$ the probability of a bit of the response to have the logical value of '0'. Therefore, by examining the relevant Hamming weight, it is easy to also get an estimation of the binary entropy of the relevant PUF entropy. The binary entropy of the relevant PUF response, of course, provides a rather adequate measurement of its randomness. However, the measurement of the relevant entropy cannot always be provided by the aforementioned function, especially in case the relevant response is heavily biased towards a particular logical value and/or bit pattern. For example, in the case of DRAM decay-based PUFs and DRAM Row Hammer PUFs, their entropy is rather based on the positions of the relevant bit "flips" [82, 83, 86, 87, 143, 158]. In this case, assuming the set S_{bf} that contains the positions of the

bit “flips”, and which has a cardinality $C_{bf} = |S_{bf}|$, while the total number of bits of the relevant PUF response is denoted by N, the probability of such a set of bit “flips” being observed is:

$$P(S_{bf}) = \frac{1}{\binom{N}{k}} , \tag{3.2}$$

and the Shannon entropy of the overall response is provided by the following function:

$$H(S_{bf}) = log_2\binom{N}{k} . \tag{3.3}$$

In this case, the Shannon entropy of the relevant PUF response *per bit* is provided by the following function:

$$H_b(S_{bf}) = \frac{log_2\binom{N}{k}}{N} , \tag{3.4}$$

i.e., by dividing the value of the overall entropy by the total number of bit positions in the relevant response, we can get an estimation of the average entropy per bit of the response, i.e., the *fractional* entropy value. In general, we note that if a highly biased response is biased towards a particular bit pattern that is known, such as the initialisation pattern of the memory addresses that will be utilised to produce the PUF response in the case of the DRAM decay-based PUF and the DRAM Row Hammer PUF, then only the positions of the response bits that are different from that bit pattern should be used to form the relevant set, i.e., a set equivalent to the S_{bf} set examined in the case of the DRAM decay-based PUF and the DRAM Row Hammer PUF, while if no such pattern exists, but the PUF response is heavily biased towards a particular logical value, then only the positions of the response bits that have the other logical value should be used to form the relevant set. Thus, we can easily note that the binary entropy function shown in Equation (3.1) is rather applicable to PUFs to which a fuzzy extractor scheme [25] is rather more applicable in order to generate a cryptographic token, e.g., a key, from them and denotes on its own the fractional entropy value, i.e., the average value of entropy per bit of each response of the relevant PUF. On the contrary, for PUFs for which a bit selection scheme would be more appropriate in order to generate a cryptographic token, e.g., a key, from them, Equation (3.4), instead of Equation (3.1), provides the correct measurement of the entropy of each relevant PUF response. Finally, the ideal value of the fractional entropy metric would be, in both cases, 1, revealing full entropy in each bit of the relevant PUF response.

[25] As already discussed, a fuzzy extractor scheme can remove noise from a PUF response, in order to stabilise it and transform it into a usable cryptographic token, e.g., a key. In order to achieve secure key agreement, a one-time enrollment is required, during which entity *A* gains access to the CRPs of the PUF of entity *B*, and a fuzzy extractor scheme is selected. Then, *A* can choose a key and encode it with a PUF response through the fuzzy extractor scheme, to produce what is known as helper data. When *A* wants to share this key with *B*, it sends to *B* the helper data and the challenge *c* corresponding to the PUF response used. *B* can then reconstruct the key chosen by *A*, by decoding the helper data with the PUF response corresponding to *c* through the fuzzy extractor scheme. Thus, a PUF can be used to retrieve and share unique keys from hardware.

3.2.3 Intra-Device and Inter-Device Hamming Distances

One of the most common metrics used to evaluate the quality of memory-based PUF responses is the Hamming distance, which is the number of bit positions that are different in two bit-strings. The Hamming distance between two PUF responses can be utilised as a metric to assess the robustness and the uniqueness of the PUF responses, depending on whether these responses correspond to the same challenge and originate from the same PUF instance or not, respectively. In particular, the *intra-device* Hamming distance reveals the robustness of the PUF responses produced by a particular device [26] for the same challenge, while the *inter-device* Hamming distance reveals how unique the PUF responses (corresponding to the same challenge) from two different devices are. Of course, the Hamming distances between responses originating from the same PUF device, but corresponding to different challenges, can be used as a metric that will reveal the number of usable CRPs that this PUF instance can truly provide. Additionally, the Hamming distances between PUF responses originating from different devices and corresponding to different challenges can provide an estimation of how many CRPs this PUF type can generate in general and how unique these may be among different PUF instances, which may also affect the entropy and randomness estimation for this type of PUF devices, as it determines with what probability an attacker will be able to predict the responses of a certain PUF instance from the responses of another PUF instance of the same PUF type.

In this work, we will mostly use the *fractional Hamming distance*, which is the number of bit positions that are different in two bit-strings divided by the number of the total bit positions in the longest of the two bit-strings. In this way, by using the *fractional intra-device Hamming distance*, we can compute the percentage of bit positions that are different in two PUF responses originating from the same device and, therefore, estimate the robustness of these responses, while, by using the *fractional inter-device Hamming distance*, we can compute the percentage of bit positions that are different in two PUF responses originating from different devices and, therefore, estimate the uniqueness of such responses. The ideal value for the fractional intra-device Hamming distance metric would be 0, revealing that the examined PUF responses are the same and, therefore, the relevant PUF exhibits complete robustness in its responses for the relevant challenge. However, as we have already noted, in practice, most memory-based PUF responses exhibit a certain amount of noise, i.e., unstable bit values, which is addressed through a bit selection or correction scheme. Therefore, in practice, any value of the fractional intra-device Hamming distance metric in the range between 0 and 0.1 can most often be considered as ideal. For the fractional inter-device Hamming distance metric, the value of 0.5 is considered as ideal, as it indicates that the two PUF responses compared are totally different from each other and, therefore, the relevant PUFs exhibit genuine uniqueness in their responses for the relevant challenge.

3.2.4 Intra-Device and Inter-Device Jaccard Index

One of the most recently introduced metrics for the assessment of the quality characteristics of memory-based PUF responses has been the Jaccard index [149, 150], which has been utilised in order to assess the robustness and the uniqueness of highly biased PUF responses, such as the responses of the DRAM decay-based PUF and the DRAM Row Hammer PUF. In general, if a highly biased response is biased

[26] The term “device” here is used to refer to a PUF instance.

towards a particular bit pattern that is known, such as the initialisation pattern of the memory addresses that will be utilised to produce the PUF response in the case of the DRAM decay-based PUF and the DRAM Row Hammer PUF, then only the positions of the response bits that are different from that bit pattern should be used to form a set, while if no such pattern exists, but the PUF response is heavily biased towards a particular logical value, then only the positions of the response bits that have the other logical value should be used to form a set. In either case, metrics that are based on the Hamming distance cannot be used, as the relevant PUF responses will be, in each particular case, highly similar to each other, as they will contain large segments of identical values, which will either essentially form segments of the relevant bit pattern, or contain the logical value, towards which each relevant response is biased. In either case, however, the Jaccard index can be employed as a metric based on the sets of bit positions that have been formed as previously explained. In particular, we can utilise the Jaccard index as a metric to measure the similarity of the relevant bit sets between different responses and, in this way, get a good measurement of the similarity of the corresponding PUF responses, in order to assess and evaluate their quality characteristics.

In particular, assuming a set S_{bp1} that is such a set of bit positions as discussed in the previous paragraph, for a particular PUF instance A of a particular PUF type, and for a particular challenge c, and another set S_{bp2} that is another such set of bit positions, for a particular PUF instance B of the same PUF type, and for the same challenge c, then, the relevant Jaccard index can be calculated using the following function:

$$J(S_{bp1}, S_{bp2}) = \frac{|S_{bp1} \cap S_{bp2}|}{|S_{bp1} \cup S_{bp2}|} \ . \tag{3.5}$$

Depending on the origin of the relevant PUF responses, we can again distinguish between the *intra-device* Jaccard index metric, which will reveal the robustness of the PUF responses produced by a particular device [27] for the same challenge, and the *inter-device* Jaccard index, which will reveal how unique the PUF responses (corresponding to the same challenge) from two different devices are. Of course, the Jaccard indices between bit sets from responses originating from the same PUF device, but corresponding to different challenges, can be used as a metric that will reveal the number of usable CRPs that this PUF instance can truly provide. Moreover, the Jaccard indices between bit sets from PUF responses originating from different devices and corresponding to different challenges can provide an estimation of how many CRPs this PUF type can generate in general and how unique these may be among different PUF instances, which may also affect the entropy and randomness estimation for this type of PUF devices, as it determines with what probability an attacker will be able to predict the responses of a certain PUF instance from the responses of another PUF instance of the same PUF type. It should be noted that all metrics based on the Jaccard index are already fractional, with their values ranging from 0 to 1. The ideal value for the intra-device Jaccard index metric would be 1, revealing that the examined sets of bits from the relevant PUF responses are the same and, therefore, the relevant PUF exhibits complete robustness in its responses for the relevant challenge. However, as we have already noted, in practice, most memory-based PUF responses exhibit a certain amount of noise, i.e., unstable bit values, which is addressed through a bit selection or correction scheme. Therefore, in practice, any value of the intra-device Jaccard index metric over the value of 0.9 can most often be considered as ideal. For the

[27] The term “device” here is used to refer to a PUF instance.

inter-device Jaccard index metric, the value of 0 is considered as ideal, as it indicates that the two sets of bits compared are totally different from each other and, therefore, the relevant PUFs exhibit genuine uniqueness in their responses for the relevant challenge. Finally, we note that when we refer to the Jaccard index for a pair of any two PUF responses in this work, we refer to the Jaccard index for the sets of bit positions that are different in each of these two PUF responses from the bit pattern that was initially written to the relevant memory cells utilised for the production of these PUF responses. Although a different initial bit pattern could be written to the memory cells utilised for each of these two responses, in order for the resulting Jaccard index value to be meaningful, most often it is assumed that the same initial bit pattern has been written to the relevant memory cells.

3.3 On the Relation Between Security and Cost

Engineering is a concept that connects theoretical science concepts to real-life technological applications, as engineering is essentially the process of applying scientific knowledge to the construction and development of (often rather useful) items and applications. In particular, in the field of computer science, "software engineering", "hardware engineering" and the composite term "software and hardware co-engineering" seem to be rather popular, referring to the use of scientific knowledge in order to develop, respectively, software, hardware, or even hardware and software together. Nevertheless, most often, scientific works tend to overlook the practical side of engineering, which is, however, of extremely high importance.

In particular, regarding the concept of "security engineering", i.e., constructing and developing security applications, protocols, and schemes, based on the relevant scientific knowledge, quite often, the relevant scientific works put forward a number of rather unrealistic assumptions. For example, as we have discussed regarding PUFs, a number of works assumed all PUFs, including memory-based ones, to be unclonable and tamper-evident. As we have noted, while this assumption would significantly enhance the security properties of PUFs, it is a rather unrealistic one. Nevertheless, such unrealistic assumptions may stem from relevant high expectations regarding the security of the relevant mechanisms, protocols, and schemes. Most scientific works seem to share and put forward a belief that a security mechanism, protocol, or scheme, is either totally insecure or completely secure. However, in practice, this is very rarely, if ever, the case. In reality, every security concept, mechanism, protocol, and scheme, is vulnerable to some attack(s). Even if one examines the case of the one-time pad encryption technique [28], which is provably impossible to break, a secret key needs to be pre-shared and kept in the memory, until it is utilised. Although this key will be used only once, in order for this encryption technique to be considered as secure, one needs to assume that the entities involved in the sharing of the secret key are legitimate and acting in good faith to each other, as well as that the key can be stored in a secure manner. Even though these assumptions are rather reasonable, yet, in practice, even these assumptions cannot always be held as being true. In general, for (symmetric) security applications, a secret, e.g., a key, needs to somehow be established and then verified among two or more parties. Therefore, an attacker may be able to gain knowledge of it, while it is being shared, stored, or accessed for verification. Hence, we can easily conclude that, at least in practice, *perfect* security, which would be equivalent to a security scheme without any assumptions needed for it to be considered as secure, cannot exist. In particular,

[28] In this technique, a message is encrypted using a secret key of the same length as the message, which is used only once (hence, the name "one-time pad"), with each bit of the message being encrypted using the corresponding bit of the key.

we also allude to the observations of Canetti and Fischlin regarding the impossibility of achieving secure cryptographic protocols without any setup assumptions [178].

Therefore, in reality, the level of security provided by a particular mechanism, scheme, or protocol, is strongly linked to the relevant assumptions being made regarding the requirements for such a mechanism, scheme, or protocol, to operate itself in a secure manner. For example, for most security protocols, all entities participating in them are assumed to be legitimate and to not act in a malevolent manner, and the communication channels among these entities are most often assumed to be authenticated, or even secure. Thus, we need to consider security as a rather *relative* term, which is strongly linked not only to the costs required for manufacturing and operating the relevant security mechanism, scheme, or protocol, but also to the costs of performing a successful attack against this security mechanism, scheme, or protocol, and to the potential gains and/or damages of such a successful attack [179]. Therefore, and most importantly, a successful attack – one that has, most often, been proposed, presented and/or examined in the relevant literature – against a particular security mechanism, scheme, or protocol, does not preclude on its own such a mechanism, scheme, or protocol from acting as a security solution in practice. In real life, it needs to be assessed how such a successful attack affects the balance among the security level expected to be provided by this mechanism, scheme, or protocol, the costs associated with it, the costs associated with the relevant attack as well as with other attacks, and the expected gains and/or damages from such attacks. Hence, in this work, we claim that the fact that most memory-based PUFs are not unclonable and/or tamper-evident does *not* preclude them from serving as adequate security mechanisms for the IoT in practice.

In general, we note that the security of cost-efficient devices has started to become an ever-increasing concern. Due to their low cost and high flexibility, these devices are widely employed in a variety of (large-scale) applications, ranging from road traffic control to weather forecasting. Recently, however, a large number of such devices have been exploited in the context of large-scale attacks. At the same time, however, the construction and operation of most, if not all, types of security mechanisms, other than memory-based PUFs, in resource-constrained devices would incur additional manufacturing and operational costs. Such costs, however, would significantly hinder the further adoption and employment of such devices in practical applications. Therefore, only lightweight and cost-efficient security mechanisms, such as memory-based PUFs can be utilised in order to secure such devices and their communication. In this case, such security mechanisms are not expected to be immune to all types of potential attacks, but rather to provide an adequate level of security with regards to their costs, the cost of performing a successful attack against them, and the gains and/or damages expected to occur from such an attack.

In practice, this balance between security and cost is described through the concept of the *acceptable* risk, as it is recognised that *perfect* security cannot exist and, therefore, a certain level of risk is always associated with any security mechanism or combination thereof employed. By defining such a level of risk considered as *acceptable*, the relevant level of security that is considered as *adequate* can also be defined, which leads to trying to achieve the lowest level of cost possible while maintaining the level of security that has been set as *adequate* for each particular use case [179]. The level of risk considered as acceptable is highly dependent on the costs of performing a successful attack, and on the potential gains and/or damages incurred by such an attack. If successful attacks are particularly hard to perform,

requiring the use of advanced equipment and expert knowledge, and affect only a small number of instances of the relevant devices (and their security mechanisms), then even a relatively high level of risk may be considered as acceptable [179]. In general, attacks against hardware security components and/or primitives, such as the memory-based PUF examined in this work, tend to have a high cost, which may not always be justified by the potential gains and benefits for the attacker. Therefore, the current aim of security is rather to make attacks economically infeasible than to actually make them completely impossible, based on the relevant level of acceptable risk [180].

In particular, we note that most, if not all, memory-based PUFs are lightweight, as they are based on inherent components of their host systems and do not require computationally heavy software in order to operate. Additionally, such PUFs are also flexible, as they, most often, can provide a relatively high number of cryptographic tokens, e.g., keys, of varying sizes and can be utilised in a highly scalable manner. In general, the construction and operation of these PUFs can be considered as cost-efficient, and these PUFs can be utilised in practical use cases, especially in the framework of IoT applications, as a significant number of IoT devices do not inherently incorporate another (potentially, dedicated) security mechanism. It is, therefore, in this context that the role of memory-based PUFs as security mechanisms, both for the IoT and in general, should be assessed. In addition, the relevant attacks that aim to clone or tamper with such PUFs are highly sophisticated, involving high-end equipment and a high degree of expertise. Therefore, such attacks cannot *truly* be considered as practical. Moreover, the potential gains and/or damages from compromising particular PUF instances cannot be considered as particularly high, especially in the context of IoT applications. Furthermore, such attacks can be prevented, as long as it can be assumed that an attacker lacks physical access to such PUFs or that their operation requires the use of privileged software. On the contrary, attacks based on environmental variations, which are rather easy to implement and which do not require a high degree of expertise, can *truly* affect the balance between security and cost in the case of memory-based PUFs. For this reason, we will examine in detail how environmental variations can affect the quality of the responses of such PUFs in Section 4.

Moreover, we also note that, since each secret used for (symmetric) security applications needs to be shared, stored, and accessed for verification, the potential success of cloning and tampering attacks against memory-based PUFs means that also the secrets used in the context of other security mechanisms, schemes, and protocols, can also not truly be considered as secure, as such secrets, most often, need to be stored in the same memories that are utilised for the construction of the relevant memory-based PUFs. Furthermore, even if such secrets were stored in other components, e.g., in TPMs or other security structures, which were assumed to be secure, these secrets would still need to be accessed, at least internally at the level of the relevant device, or even of the relevant security structure, itself. In this case, the buses carrying these secrets could be probed and the secrets be revealed, in the same manner that the response of a memory-based PUF could be stolen by probing the relevant memory buses. Additionally, in any case, an attacker can potentially damage the relevant security structure, regardless of whether it is a PUF of any type, a TPM, or any other security component, or even completely destroy the relevant device, in order to cause a Denial of Service (DoS) attack. Therefore, even security components that offer a relatively high level of security, albeit, at a relatively high cost, cannot guarantee perfect security against all attacks possible, or even provide a higher level of security than the memory-based PUFs

examined in this works, especially in the context of IoT applications, which, most often, utilise cost-efficient, resource-constrained devices.

We can, therefore, easily conclude that the adoption of high-end (and high-cost) security solutions in the context of IoT applications does not seem to be, in general, practical, in the same manner that the adoption of bulletproof glasses for the windows of most houses is also not considered as practical. Additionally, the inherent advantage of intrinsic memory-based PUFs being lightweight and cost-efficient, allows them to be considered as an adequate security mechanism for the IoT, in the same manner that the inherent advantage of glass being transparent allows it to be considered as an adequate and practical isolation mechanism for a wide range of use cases, which range from common houses and vehicles to bottles and drinking vessels, the latter of which, i.e., the drinking vessels, are themselves commonly known as *glasses*. Therefore, while glass is rather more fragile than bricks and mortar [29], it is rather widely used in practice due to its advantage of being transparent, in the same way that the fact that memory-based PUFs are lightweight, cost-efficient, and flexible security primitives, can also allow them to potentially act as adequate security mechanisms in a wide range of practical applications, as long as they can provide an *acceptable* level of security in relation to the relevant risks [30].

In order to assess what may constitute an adequate level of security for the memory-based PUFs examined in this work, we need to examine what may constitute acceptable assumptions for their operation as security mechanisms in the context of IoT applications. To this end, we also acknowledge that the formal security analysis of memory-based PUFs remains an open research topic, which has not yet been extensively addressed, but which has attracted the attention of recent works, such as [181, 182, 183]. In general, however, we note that, by using hardware components that already form part of the relevant

[29] In this context, and alluding to the previous paragraph that suggested that almost all, if not all, of the most well-known and most commonly used security mechanisms cannot prevent all potential attacks, at least on their own, we observe that such is the case also with both glass, and brick and mortar, as neither of them can, for example, fully protect from the effects of a nuclear weapon attack. Nevertheless, as such an attack is usually considered as rather improbable, although the relevant tensions between world powers are currently rising and a significant number of people live in urban areas that can be considered as potential targets of such attacks in the event of a nuclear war, the number of nuclear war shelters that have been built is rather low, and the number of domestic houses with walls that would protect from a relevant attack remains rather insignificant, with an extremely low number of people even considering the potential merits of walls of such a protective nature. In the same way, the number of devices that offer an extremely high level of protection against physical and other relevant attacks remains rather low, with an extremely low number of people considering the potential merits of physically protecting such devices with mechanisms and structures that would constitute adequate measures and countermeasures to completely preclude the success of physical attacks, as such protection is considered neither practical nor necessary. Furthermore, it is rather worth mentioning that even the physical mechanisms and structures that will allow the relevant world leaders to authorise the use of nuclear weapons, i.e., the instances of what is commonly known as a nuclear "briefcase" or a nuclear "football", are not truly kept protected from all possible attacks, as this is rather impractical, as the world leaders tend to travel around the world, unnecessary, as they are rather *adequately* protected considering the relevant (rather low) level of acceptable risk, and, at the very end, impossible – if the relevant physical structure (and its nearby physical location) is, for example, stricken with several projectiles and/or missiles of adequate energy, there is no way to protect it from destruction, notwithstanding whether the relevant objects are projectiles, guided missiles, or a shower of meteorites. As PUFs were originally considered in the context of nuclear weapon identification, the aforementioned attacks are rather relevant within the framework of world politics, global geopolitical strategy, and military doctrines, especially with regards to the concept of Mutual(ly) Assured Destruction (MAD) that seems to have so far prevented world superpowers from launching pre-emptive nuclear attacks, i.e., surprise first strikes, against each other.

[30] A rather more relevant and fitting example would be the use of DNA as a biometric. While DNA-based methods lead to much better identification and authentication results, they are more costly, more time-consuming, and less practical than fingerprint matching or face recognition techniques and, therefore, are far less commonly adopted in practice than fingerprint matching and face recognition, which have been widely adopted due to their relatively more time-efficient and cost-effective nature.

computer system, in order to secure it, we reduce the security issue to a software modification/addition problem, which has a relatively low-cost and low-overhead solution. In general, we note that the relevant literature has proven that attacks against memory-based PUFs are successful only when an attacker has physical access to the corresponding device or to a relevant side channel, or can directly access the relevant memory [1, 2, 3, 4, 5, 6, 7, 24, 76, 85, 130, 141, 163, 164, 165, 167, 173]. In general, as long as the entropy and the randomness of the responses of such PUFs are adequately high, brute force attacks, dictionary attacks, as well as guessing attacks, against them cannot succeed, even if such attacks are based on expert manufacturing knowledge. Moreover, if an attacker tries to remove or replace the relevant PUF, this would probably be immediately noticed as the relevant PUF most often constitutes also an inherent memory of the system and/or the relevant PUF response would be different. Therefore, the only way for an attacker to be successful is to gain immediate access to the relevant CRPs of such PUFs. Therefore, in order to prevent a successful attack for such PUFs, we need to prevent an attacker from gaining physical or logical access to them, or to relevant side channels, to the extent necessary to prevent such an attacker from gaining any meaningful information regarding the relevant CRPs, which constitute the relevant secret, and/or modifying, tampering with, or otherwise affecting them. To this end, we also need to assume that an attacker cannot gain access to the relevant CRPs during the one-time enrollment phase of the relevant protocols, which constitutes their setup phase, during which the secret is being shared. While these assumptions can be considered as rather impractical for devices that are left unattended for extended periods of time, such as IoT devices, they, nevertheless, constitute the only way to prevent an attack against memory-based PUFs, as they are essentially equivalent to an assumption that an attacker does not have a way to gain access to the relevant secret, i.e., to the relevant CRPs of these PUFs. Additionally, such assumptions seem to be necessary in all cases in which a cryptographic token, such as a key, is stored in memory. Therefore, in general, we believe that the PUFs examined in this work can be considered as secure when these assumptions hold, i.e., when a relevant threat and/or attack model is adopted for their assessment.

However, as, *in practice*, preventing malevolent parties from accessing the relevant memory modules cannot always be guaranteed, we need to devise schemes and protocols that can accommodate to this [31].

[31] The author is a rather avid supporter of practical security. However, at the same time, he is well aware of the fact that perfect security cannot exist either in theory or in practice. In particular, he is familiar with the fact that micro-probing and reverse engineering can lead into most secrets of a physical device being revealed, notwithstanding whether these secrets are stored or contained in the relevant device. Additionally, he is also aware that devices that are vulnerable to such physical attacks, e.g., smart cards, not only continue to be utilised in practical applications, but are also still considered as secure *enough* to be used in security applications, for example, in the context of services and transactions relevant to the banking sector. Moreover, the author also understands that certain types of attacks cannot always be prevented, such as DoS attacks. Nevertheless, the author was rather surprised to discover that the amazon.de online store website, which employs a certificate whose signature has been hashed using the SHA-256 hash algorithm and encrypted using the RSA-2048 encryption algorithm, allows a user/customer with an account in good standing to use any German bank account to pay for an order, by providing only the relevant account owner's name and the account's International Bank Account Number (IBAN), without any further validation that the owner of that account, who may be a different person, totally unrelated to the person placing the order, approves of the relevant order being paid from that account. For purely scientific reasons, this was tested by placing an order of more than 500 euros with the amazon.de online store, which was normally fulfilled and delivered. The relevant amount was paid back to the relevant person, who was completely unaware of the relevant (unauthorised) transaction. In this way, it was rather proven that the relevant security scheme utilised by this online store website was not in any way able to prevent the relevant fraudulent transaction. Even though the relevant bank account credentials were probably transferred online in a rather secure manner, the very transaction itself was completely unauthorised and a clear instance of fraud, as the credentials of the victim, which were acquired from the front side of the victim's Electronic Cash (EC) bank card and should have not been

To this end, we note again that most of the relevant attacks that can potentially be successful *in practice* against a memory-based PUF with good quality characteristics require relatively high expertise and costs, as well as high-end equipment, and may affect only a single PUF instance at a time. Nevertheless, we also observe that attacks against memory-based PUFs that are based on the dependency of such PUFs upon some environmental factor can be implemented easily and at a low cost. In either case, we note that the use of a dedicated memory module for the implementation of the relevant PUF, or the denial of access to the relevant module for unprivileged software code, can potentially prevent such attacks, but could prevent the relevant PUF from being a lightweight and, thus, also a practical security mechanism, as the relevant memory module may not be able to function as both a hardware-based security primitive and a normal memory in the context of its host system. In general, the addition of any hardware-based or software-based countermeasure against the relevant attacks could potentially prevent these PUFs from acting as lightweight and, thus, practical security mechanisms. We, therefore, examine in detail how the variations of different environmental factors, including ambient temperature variations, can affect these PUFs in Section 4, and we will propose schemes and protocols that can accommodate to the potential dependencies of these PUFs to environmental factors, or even to successful attacks against them, in Section 5. In this way, not only we can provide insights into what could constitute a level of acceptable risk for the memory-based PUFs examined in this work, but also assist in raising this level, in order to facilitate the adoption of such PUFs in practical security applications.

able to be utilised for such a fraudulent transaction, provided *in fact* enough information for the fraudulent transaction to take place. We, therefore, note that *in practice* systems and services that are considered, in general, as secure may not always provide a high, or even an *acceptable*, level of security.

Partially Based Upon Peer-Reviewed Content Published In

- W. Xiong, A. Schaller, N. A. Anagnostopoulos, M. U. Saleem, S. Gabmeyer, S. Katzenbeisser & J. Szefer, "Run-Time Accessible DRAM PUFs in Commodity Devices", Proceedings of the 18th International Conference on Cryptographic Hardware and Embedded Systems (CHES 2016), vol. 9813 of "Lecture Notes in Computer Science (LNCS)", pp. 432-453, Springer, 2016. DOI: 10.1007/978-3-662-53140-2_21
- A. Schaller, W. Xiong, N. A. Anagnostopoulos, M. U. Saleem, S. Gabmeyer, S. Katzenbeisser & J. Szefer, "Intrinsic Rowhammer PUFs: Leveraging the Rowhammer Effect for Improved Security", Proceedings of the 2017 IEEE International Symposium on Hardware Oriented Security and Trust (HOST 2017), IEEE, 2017. DOI: 10.1109/HST.2017.7951729
- N. A. Anagnostopoulos, T. Arul, M. Rosenstihl, A. Schaller, S. Gabmeyer & S. Katzenbeisser, "Low-Temperature Data Remanence Attacks Against Intrinsic SRAM PUFs", Proceedings of the 21st Euromicro Conference on Digital System Design 2018 (DSD 2018), pp. 581-585, IEEE, 2018. DOI: 10.1109/DSD.2018.00102
- N. A. Anagnostopoulos, T. Arul, Y. Fan, J. Lotichius, C. Hatzfeld, F. Fernandes, R. Sharma, F. Tehranipoor & S. Katzenbeisser, "Securing IoT Devices Using Robust DRAM PUFs", Proceedings of the 2018 Global Information Infrastructure and Networking Symposium (GIIS 2018), IEEE, 2018. DOI: 10.1109/GIIS.2018.8635789
- A. Schaller, W. Xiong, N. A. Anagnostopoulos, M. U. Saleem, S. Gabmeyer, B. Škorić, S. Katzenbeisser & J. Szefer, "Decay-Based DRAM PUFs in Commodity Devices", IEEE Transactions on Dependable and Secure Computing (TDSC), vol. 16, iss. 3, pp. 462-475, IEEE, 2018. DOI: 10.1109/TDSC.2018.2822298
- N. A. Anagnostopoulos, T. Arul, Y. Fan, C. Hatzfeld, A. Schaller, W. Xiong, M. Jain, M. U. Saleem, J. Lotichius, S. Gabmeyer, J. Szefer & S. Katzenbeisser, "Intrinsic Run-Time Row Hammer PUFs: Leveraging the Row Hammer Effect for Run-Time Cryptography and Improved Security", Cryptography, vol.2, iss. 3, MDPI, 2018. DOI: 10.3390/cryptography2030013
- W. Xiong, N. A. Anagnostopoulos, A. Schaller, S. Katzenbeisser & J. Szefer, "Spying on Temperature Using DRAM", Proceedings of the 22nd Design, Automation & Test in Europe Conference & Exhibition (DATE 2019), pp. 13-18, IEEE, 2019. DOI: 10.23919/DATE.2019.8714882
- N. A. Anagnostopoulos, Y. Fan, T. Arul, R. Sarangdhar & S. Katzenbeisser, "Lightweight Security Solutions for IoT Implementations in Space", Proceedings of the 2019 IEEE Topical Workshop on Internet of Space (TWIOS 2019), IEEE, 2019. DOI: 10.1109/TWIOS.2019.8771257
- N. A. Anagnostopoulos, T. Arul, M. Rosenstihl, A. Schaller, S. Gabmeyer & S. Katzenbeisser, "Attacking SRAM PUFs Using Very-Low-Temperature Data Remanence", Microprocessors and Microsystems, vol. 71, art. 102864, Elsevier, 2019. DOI: 10.1016/j.micpro.2019.102864
- N. A. Anagnostopoulos, S. Ahmad, T. Arul, D. Steinmetzer, M. Hollick & S. Katzenbeisser, "Low-Cost Security for Next-Generation IoT Networks", ACM Transactions on Internet Technology, vol. 20, iss. 3, art. 30, ACM, 2020. DOI: 10.1145/3406280

In this section, we examine the robustness of the responses of memory-based PUFs to variations of their external environment. In particular, we evaluate memory-based PUFs under nominal conditions, ambient temperature changes, supply voltage variations, and in the presence of (ionising) radiation. In this way, we are able to assess the performance of such PUFs in adverse environmental conditions, in order to determine their suitability for securing IoT application segments, which may include smart homes, smart grids, and space modules and applications.

In general, we note that the potential dependency of memory-based PUFs on external environmental factors not only could prevent them from providing security to certain IoT application areas, such as space applications, but it could even be used for the successful implementation of attacks both against such PUFs themselves [3, 4] as well as against other relevant devices, networks, and applications [184]. In particular, we note that such attacks are easy to implement and rather hard to mitigate, as the external environment of a device is rather hard to control in practical applications. For this reason, it is important to identify ways in which such potential dependencies can be alleviated.

The effects of external environmental factors, e.g., temperature and voltage variations, on memory-based PUFs have also been examined in other works [74, 79, 80, 82, 85, 86, 87, 88, 113, 130, 163, 173, 185, 186], however none of these works has provided a truly comprehensive study of such effects. Moreover, although some of these works identified potential dependencies of memory-based PUFs on environmental factors, they did not really explore these dependencies in depth. In this way, these works have failed to thoroughly address the effect that such dependencies may have on the ability of memory-based PUFs to serve as practical security mechanisms. In this section, we concisely examine the effects of environmental factors on a selection of well-known memory-based PUFs, with a focus on identifying potential dependencies of memory-based PUFs on such environmental factors, in order to then try to address such dependencies through the construction of relevant protocols and schemes in Section 5.

Moreover, we also observe that minor discrepancies in the form of noise found in PUF responses can be considered as insignificant within the context of the relevant security protocols, as most, if not all, of them incorporate an inherent error correction mechanism for PUF responses, most often in the form of a fuzzy extractor scheme. In this context, in this work, we will try to identify if specific environmental factors have a rather significant, i.e., not just a minor, but rather a major, effect on the responses of memory-based PUFs. Additionally, it is important to state that other rather intrinsic factors, such as aging [80, 85, 86, 130, 163, 173, 174, 187], may also have an effect on the responses of memory-based PUFs, but such factors are rather outside the scope of this work, as they are rather inherent and, thus, do not appear to be easy to control externally. Therefore, although such factors may have a profound effect on memory-based PUFs, it may be rather hard to utilise them in order to manipulate the responses of a PUF, in a manner that is easy to implement or that is cost-efficient. In general, we also note that it may be possible to easily mitigate such aging effects [188]. Finally, we also do not examine, in this work, other inherent factors that may affect the responses of memory-based PUFs, such as spatial correlation, but we do observe that, in the relevant literature [189, 190], it has been noted that the values of the responses of memory-based PUFs do not appear to be highly correlated in terms of spatial correlation.

4.1 Memory-Based PUFs Under Nominal Conditions

In this subsection, we will try to define what can be considered as nominal operating conditions for memory-based PUFs and present some results regarding the responses of such PUFs operating in such nominal conditions. To this end, we note that, under nominal conditions, there will obviously be an absence of (ionising) radiation, at least of a significant level of it, as low levels of (ionising) radiation are rather present everywhere on our planet. Regarding the supply voltage, under nominal conditions, it should be also nominal, i.e., the device hosting the relevant PUF should be powered with the regular voltage needed for its operation, which most often is 3.3 or 5 Volts (V), but can sometimes also be 12 V or higher. The memory component itself that serves as a PUF is most often powered at 1.5 or 1.8 Volts (V), or even at 1.2 V if it is a low-power device [82, 86, 87]. Regarding the temperature, we need to distinguish between the ambient temperature, which may vary between -10°C and 40°C, and the operating temperature of the device, which most often is between 40°C and 80°C. Nevertheless, we do assume that the nominal ambient temperature varies only between 15°C and 25°C, which agrees with most definitions for *standard* and/or *room* temperature.

In general, we note that other works have discussed in depth the ability of memory-based PUFs, such as SRAM PUFs [113, 145, 146, 185, 186], DRAM-based PUFs [79, 82, 85, 86, 88, 143, 144, 158, 159], and Flash-memory-based PUFs [161, 162], to provide adequate security under nominal conditions. In this work, we will examine the ability of these memory-based PUFs types to provide high-quality responses that can be used in order to provide an *acceptable* level of security, by utilising the relevant quality metrics that have been discussed in Section 3.2.

In particular, we will examine the quality characteristics of SRAM PUFs, DRAM decay-based PUFs, DRAMs Row Hammer PUFs, and NAND Flash memory programming-disturbance-based PUFs under nominal conditions. In this way, we will be able to examine whether each of the three different types of widely used memory (SRAM, DRAM, and Flash memory) can allow for the implementation of PUFs that provide an *acceptable* level of security under nominal conditions.

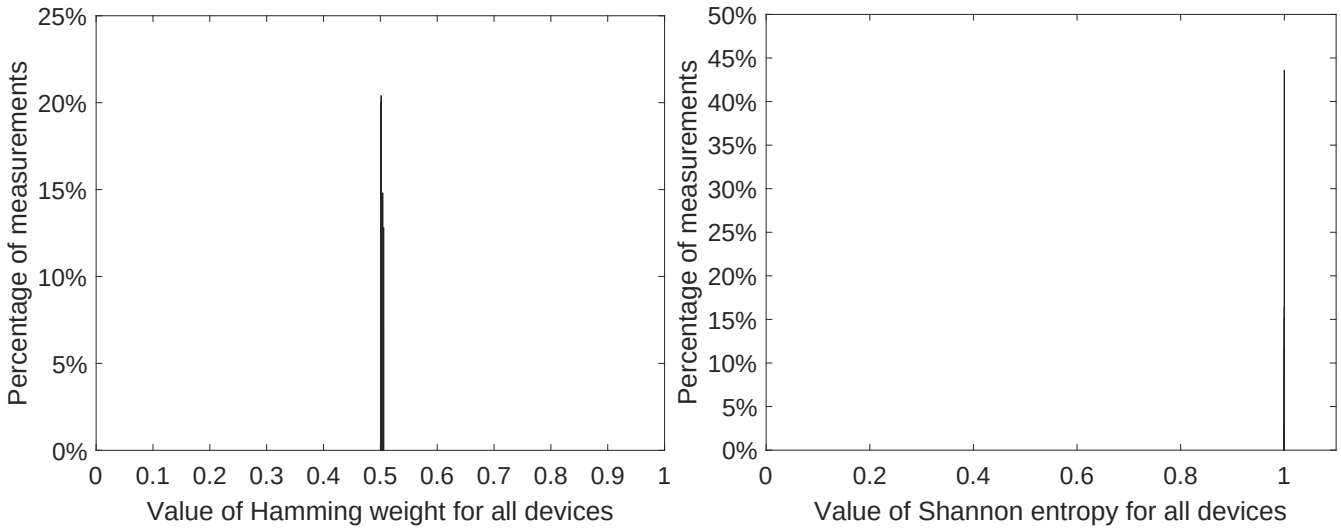
4.1.1 SRAM PUFs Under Nominal Conditions

The quality characteristics of the SRAM PUF have been examined under nominal conditions in a number of works. In general, SRAM PUFs have been shown to exhibit extremely good PUF characteristics, providing responses of high robustness, entropy and randomness, and uniqueness. In particular, it has been shown that SRAM PUFs implemented on the on-die SRAM module of a Texas Instruments Stellaris LM4F120 LaunchPad evaluation board (EK-LM4F120XL) exhibit good PUF characteristics [3, 4, 146]. SRAM PUFs utilising the on-die SRAM module of a Texas Instruments Tiva C Series TM4C123G LaunchPad evaluation board (EK-TM4C123GXL) also exhibit good PUF characteristics. Moreover, SRAM PUFs implemented using the "blob_fw_code" memory region of the Qualcomm Atheros QCA9500 wireless communication module of the TP-Link Talon AD7200 router exhibit extremely good PUF characteristics [145]. In this work, we will examine the quality characteristics of SRAM PUFs implemented on the "blob_uc_code" memory region of the Qualcomm Atheros QCA9500 wireless communication module of the TP-Link Talon AD7200 router. For this purpose, we utilise the relevant metrics discussed in Section 3.2, i.e., the Hamming weight, the binary (Shannon) entropy, and the intra-device and inter-device Hamming distances. Each of the SRAM PUFs implemented on the "blob_uc_code" memory region can,

at most, provide a PUF response of 128 KB, and our results are based on responses of this size. We have used five different devices, taking 50 PUF response readings from each of them, in order to evaluate the quality of the characteristics of the implemented PUF. All measurements have been performed at room temperature.

Evaluation of the Entropy of the Examined SRAM PUF Under Nominal Conditions

We compute the binary (Shannon) entropy of the relevant PUF responses, as well as their fractional Hamming weight, in order to gain information on their entropy and, therefore, also insights into their randomness. In particular, Figure 4.1a presents the fractional Hamming weight values for all the individual measurements. As it can be easily seen, all of the measurements have a fractional Hamming weight extremely close to the ideal value of 0.5, with the minimum fractional Hamming weight observed being 0.50061 and the maximum 0.50614. The mean fractional Hamming weight is 0.50290. Figure 4.1b shows the binary (Shannon) entropy values for all the individual measurements. As it can be easily seen, all of the measurements have a Shannon entropy extremely close to the ideal value of 1, with the minimum Shannon entropy observed being 0.99989 and the maximum 1. The mean Shannon entropy value is 0.99997. We can, thus, conclude that the examined PUF exhibits near-perfect entropy and, therefore, also a high degree of randomness, under nominal conditions.



(a) Fractional Hamming weight

(b) Binary (Shannon) entropy

Figure 4.1: Evaluation of the entropy of the responses of the examined SRAM PUF under nominal conditions.

Evaluation of the Robustness of the Examined SRAM PUF Under Nominal Conditions

In order to measure the robustness of the PUF responses produced by the examined PUF, we compute the fractional intra-device Hamming distance for all the measurements originating from the same device, for all the devices being tested. In this way, we can know how different are the PUF responses measured from the same device at different times and, in this way, measure their robustness. As already mentioned, PUF responses tend to be noisy, which is the reason why a (reverse) fuzzy extractor algorithm

is required in order to stabilise them and, at the same time, allow for the production of a cryptographic token that is based on them.

Figure 4.2 presents the fractional intra-device Hamming distance values for all the measurements originating from the same device, for all the devices being tested. As it can be easily seen, all of the measurement pairs originating from the same device have a fractional intra-device Hamming distance that is rather close to the ideal values of this metric, which lie in the range between 0 and 0.1, with the minimum fractional intra-device Hamming distance observed being 0.03028 and the maximum 0.12213. The mean fractional intra-device Hamming distance is 0.03906. These results suggest that PUF responses taken from the same device vary, on average, less than 4% and, at most, ≈12.22%, which allows us to conclude that the examined PUF is very robust under nominal conditions. Finally, this level of noise can be easily corrected using an error correction code, in the context of a fuzzy extractor scheme [16, 76, 136, 137, 138, 139].

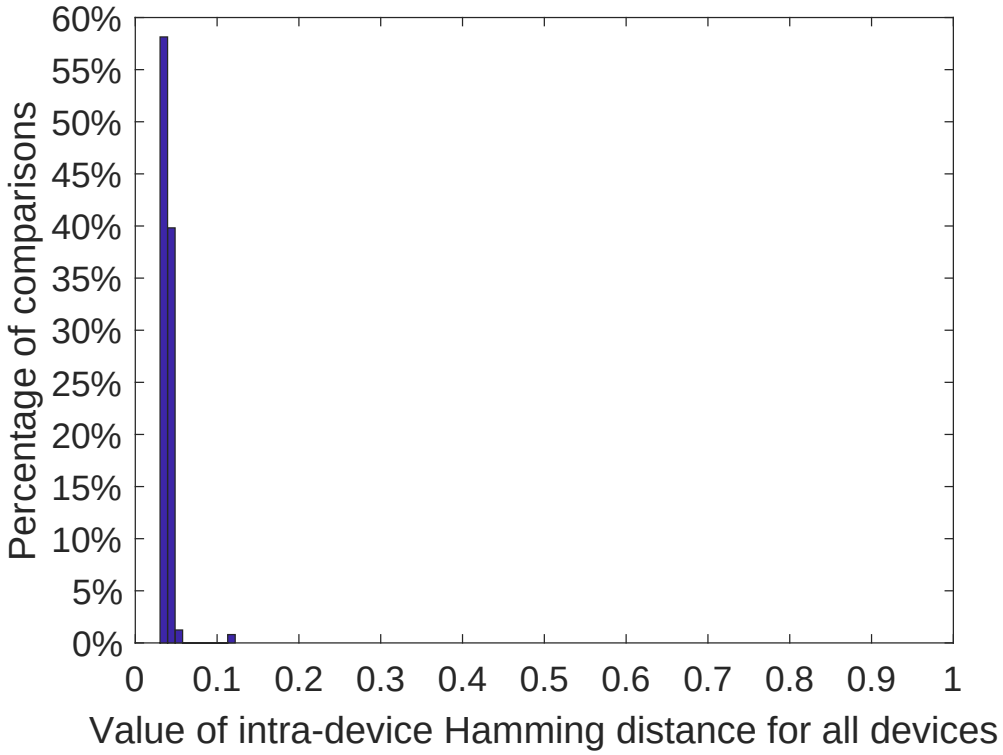


Figure 4.2: Evaluation of the robustness of the responses of the examined SRAM PUF under nominal conditions.

Evaluation of the Uniqueness of the Examined SRAM PUF Under Nominal Conditions

In order to test the uniqueness of the responses produced by the examined PUF, we compute the fractional inter-device Hamming distance between all the pairs of responses, so that the two PUF responses being compared originate from different devices, for all the devices being tested. In this way, we can know how different are the PUF responses measured from different devices and, therefore, also estimate their uniqueness.

Figure 4.3 presents the fractional inter-device Hamming distance values for all the measurements originating from two different devices, for all the devices being tested. As it can be seen, all of the mea-

surement pairs have a fractional inter-device Hamming distance close to the ideal value of 0.5, with the minimum fractional inter-device Hamming distance observed being 0.48787 and the maximum 0.50024. The mean fractional inter-device Hamming distance is 0.49524. Therefore, the PUF responses produced by the different devices of this SRAM PUF appear to be highly unique, under nominal conditions.

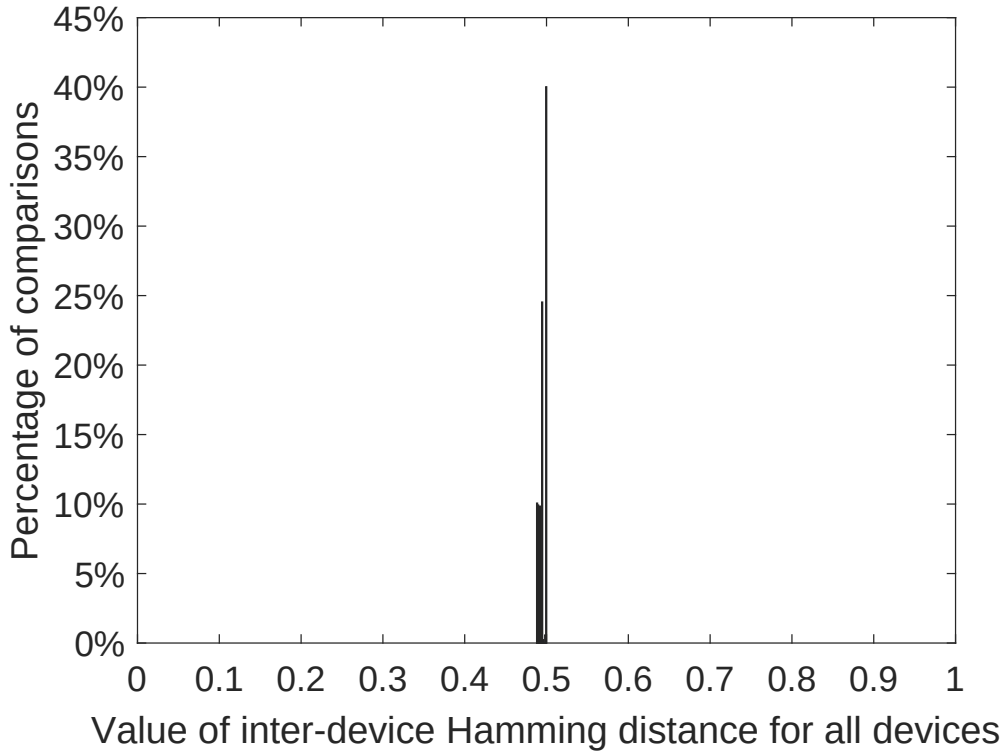


Figure 4.3: Evaluation of the uniqueness of the responses of the examined SRAM PUF under nominal conditions.

4.1.2 DRAM-Based PUFs Under Nominal Conditions

The ability of DRAM-based PUFs to provide responses of high quality that can be utilised to provide an *acceptable* level of security has been examined in a number of relevant works. In general, DRAM-based PUFs have been shown to be able to provide adequate security under nominal conditions [79, 80, 82, 83, 85, 86, 87, 88, 143, 144, 147, 153, 159]. Nevertheless, in this work, we focus on DRAM decay-based PUFs and DRAMs Row Hammer PUFs, for reasons that have already been discussed in Section 3.1.

In particular, we note that, under nominal conditions, our run-time implementation of the DRAM decay-based PUF on the PandaBoard ES Rev B3 can provide, within 60 second (s), more than 20 KB of cells whose values have "flipped" in a memory region of 32 MB, while our run-time implementation of the DRAM decay-based PUF on the Intel Galileo Gen 2 board can provide more than 1.7 KB of cells whose values have "flipped" within the same time in a memory region of the same size [86]. In general, we observe that, within 300s (i.e., 5 minutes), ≈2% of the memory cells being utilised as a run-time DRAM decay-based PUF have "flipped", when a PandaBoard ES Rev B3 is used for the implementation of this PUF [86, 153], and 0.5% to 1% of the memory cells being utilised as a run-time DRAM decay-based PUF, when an Intel Galileo Gen 2 board is used [83, 86, 147, 153]. Therefore, as the relevant works prove, DRAM decay-based PUF can provide an *acceptable* level of security under nominal conditions, as

the number of bit "flips" observed is rather adequate even if the DRAM refresh operation is suspended for a rather short period of time.

Regarding the DRAMs Row Hammer PUFs, which has been implemented on the PandaBoard ES Rev B3, the relevant works [82, 143] demonstrate that, depending on the type of row hammering being employed and the initial values of the relevant PUF and hammer rows, 0.5% to 4% of the PUF cells, i.e., the memory cells that provide the DRAMs Row Hammer PUFs response, have "flipped", i.e., their value has changed during the operation of this PUF, when, under nominal conditions, and for a time period of 120s, the DRAM refresh operation has been suspended and the cells of the relevant hammer rows have been rapidly and repeatedly accessed, i.e., they have been "hammered". For a time period of 60s and under nominal conditions, the amount of the relevant PUF cells that have "flipped" ranges between 0.05% and 1%, again depending on the type of row hammering being employed and the initial values of the relevant PUF and hammer rows [82, 143]. In comparison, for a time period of 60s, during which only the DRAM refresh operation has been suspended, the amount of the cells that have "flipped", in the case of the run-time DRAM decay-based PUF implemented on the PandaBoard ES Rev B3, is only ≈0.06% [86] under nominal conditions. Therefore, it is easy to conclude that also this PUF can provide adequate security under nominal conditions. In order to validate this conclusion, we will briefly examine the quality characteristics of DRAMs Row Hammer PUFs implementations on the PandaBoard ES Rev B3 under nominal conditions, utilising the relevant metrics discussed in Section 3.2, i.e., not only the number of bit "flips" present in the responses of this PUF, but also their binary entropy, and their intra-device and inter-device Jaccard indices. In our evaluation, we use four different devices, and have collected 20 responses for each combination of the parameters shown in Table 4.1. The type of row hammering is denoted by `RH type`, the total size of the PUF rows is denoted by `PUF size`, the initial pattern of the PUF rows is denoted by `PUF row IV`, the pattern written to the hammer rows – which is also their initial pattern – is denoted by `hammer row IV`, the time period during which the DRAM refresh operation is suspended and the hammer rows are being "hammered" is denoted by `RH time`, and, finally, `Cache` denotes the state of the relevant cache. Although this PUF could provide, at most, ≈1 GB, we collected, in this work, responses with a size of only 128 KB, so that the characteristics of this PUF and of the SRAM PUF examined in the relevant previous segment would remain somehow comparable. The same memory addresses were utilised in all the devices used for the implementation of this PUF, with regards to hammer and PUF cells and rows. All measurements have been performed at

Table 4.1: Parameters Used for Evaluation of the Row Hammer PUF Characteristics, and Their Corresponding Set of Values. Cases with `PUF row IV` = "`0x55`" are not examined in depth, as we have verified that they lead to no bit "flips" for our PandaBoard implementations. Finally, both the firmware and the kernel module implementation have been tested using this configuration.

Parameter	Evaluated Values
`RH type`	single-sided (`SSRH`), double-sided (`DSRH`)
`PUF size`	128 KB
`PUF row IV`	"`0x00`", "`0xAA`", "`0xFF`"
`hammer row IV`	"`0x00`", "`0x55`", "`0xAA`", "`0xFF`"
`RH time`	60s, 120s
`Cache`	'enabled', 'disabled'

room temperature. Two implementations have been considered; one utilising the relevant firmware at boot-time and one utilising the (Linux) OS kernel, through a relevant kernel module, at run-time.

Regarding the Entropy of the Responses of the DRAM Row Hammer PUF Under Nominal Conditions

Our results regarding the number of bit "flips" observed in the responses of our DRAM Row Hammer PUF implementations are shown in Figure 4.4 and Figure 4.5, for the firmware and the kernel module implementation, respectively. The *fractional number of bit "flips"* shown in these Figures denotes the total number of PUF cells that have "flipped", i.e., have changed their logical values, during the operation of the DRAM Row Hammer PUF, divided by the total number of PUF cells.

As Figures 4.4 and 4.5 show, the average number of bit "flips" being observed in the DRAM Row Hammer PUF responses is rather dependent on the `RH time`, the `hammer row IV` and the `PUF row IV`. In particular, setting `RH time`= 120s leads to ≈400% more bit "flips" than when setting

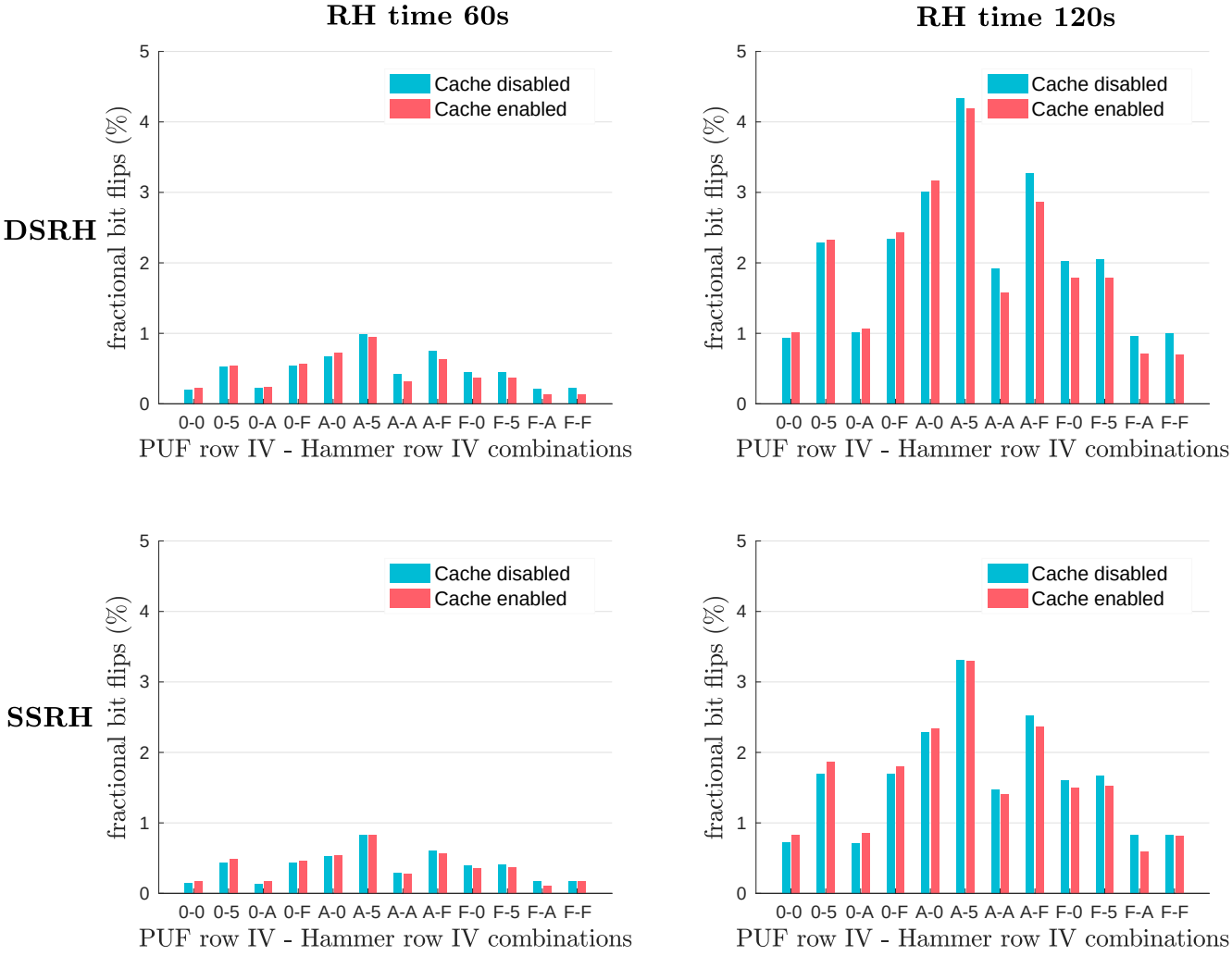


Figure 4.4: Average fractional number of bit "flips" observed in the responses of the firmware implementation of the Row Hammer PUF, for different combinations of `hammer row IV` and `PUF row IV` under nominal conditions. Configuration used: `PUF size`= 128KB, (`RH time`= 60s/`RH time`= 120s), (`RH type`=SSRH/`RH type`=DSRH), and (Cache disabled/Cache enabled).

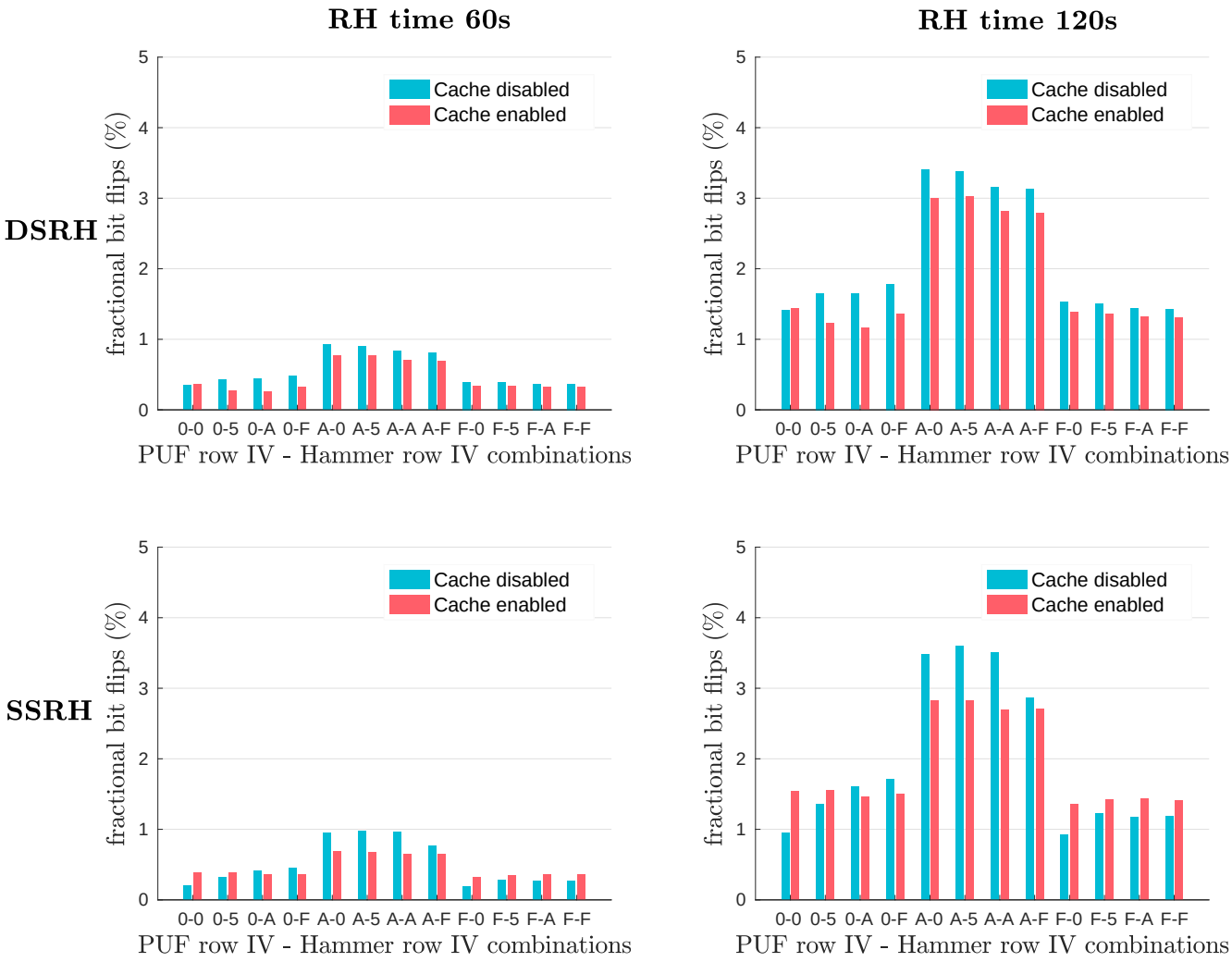


Figure 4.5: Average fractional number of bit "flips" observed in the responses of the kernel module implementation of the Row Hammer PUF, for different combinations of `hammer row IV` and `PUF row IV` under nominal conditions. Configuration used: `PUF size`= 128KB, (`RH time`= 60s/`RH time`= 120s), (`RH type`=SSRH/`RH type`=DSRH), and (Cache disabled/Cache enabled).

`RH time`= 60s. We also note that the largest average number of bit "flips" and, therefore, the highest entropy occur when `hammer row IV` = "`0xAA`" and `PUF row IV` = "`0x55`".

On the contrary, the `RH type` and the cache state do not seem to affect significantly the number of bit "flips" being observed in the relevant PUF responses. In particular, the use of `DSRH` results in slightly more bit "flips" observed than the use of `SSRH`. However, the difference in the number of bit "flips" produced with the two methods does not appear to be significant. Nevertheless, `SSRH` requires ≈55% less memory and involves fewer memory accesses in comparison to `DSRH`. Furthermore, the relevant cache being enabled even seems to be increasing the number of bit "flips" observed for some combinations of `hammer row IV` and `PUF row IV`, while, for most cases, disabling the cache operation leads to an increase in bit "flips", as we were expecting.

Finally, the firmware implementation seems to provide more bit "flips" in comparison to the kernel module implementation, for all cases. This difference is due to the fact that the firmware implementation accesses directly the DRAM, while the kernel module implementation uses memory-mapped registers to

access it, a fact that leads to fewer DRAM accesses being achieved by the kernel module implementation for the same `RH time` and, therefore, the hammer rows being hammered less often, causing fewer bit "flips" in the PUF rows.

Based on the results shown in Figures 4.4 and 4.5, we can easily assume that for `hammer row IV` = "`0xAA`" and `PUF row IV` = "`0x55`", the minimum fractional number of bit "flips" observed in the responses of our DRAM Row Hammer PUF implementations will be at least 0.5% for `RH time` = 60s, and 2.5% for `RH time` = 120s. Based on Equation (3.4), the lowest bound for the Shannon entropy of the relevant PUF response *per bit*, for `RH time` = 60s, is:

$$H_b = \frac{log_2\left(\begin{array}{c}1048576\\ \frac{0.5}{100}\times 1048576\end{array}\right)}{1048576} \approx 0.045 \ , \tag{4.1}$$

while, for `RH time` = 120s, it is:

$$H_b = \frac{log_2\left(\begin{array}{c}1048576\\ \frac{2.5}{100}\times 1048576\end{array}\right)}{1048576} \approx 0.169 \ . \tag{4.2}$$

Therefore, given the vast amount of available cells, the DRAM Row Hammer PUF responses show sufficient entropy, under nominal conditions, to derive cryptographic keys. For example, the derivation of a 1024-bit key, given $H_b = 0.045$, requires ≈2.8 KB, while, given $H_b = 0.169$, it requires ≈757 Byte (B). Thus, as `PUF size` = 128 KB, this PUF can create at least 45 1024-bit keys at `RH time` = 60s and 173 such keys at `RH time` = 120s.

Regarding the Robustness and the Uniqueness of the Responses of the DRAM Row Hammer PUF Under Nominal Conditions

We also consider the intra-device Jaccard index, denoted by J_{intra}, and the inter-device Jaccard index, denoted by J_{inter}, in order to assess the robustness and the uniqueness, respectively, of the responses of our DRAM Row Hammer PUF implementations. As Figure 4.6 shows, the Row Hammer PUF responses of both the firmware and the kernel module implementation exhibit a high degree of robustness and uniqueness, as, for both cases, the J_{intra} values are close to 1 and the J_{inter} values close to 0. We need to note that our results consider all cases for the different parameter values shown in Table 4.1, and not just the different cases for `hammer row IV` = "`0xAA`" and `PUF row IV` = "`0x55`".

As the values of J_{intra} and J_{inter} are not overlapping in any case, we can conclude that all of the DRAM Row Hammer PUF instances can be uniquely identified in a robust manner. Nevertheless, we note that in Figure 4.6, the J_{intra} values for `RH time` = 120s are closer to 1 than the relevant values for `RH time` = 60s, and the J_{inter} values for `RH time` = 60s closer to 0 than the relevant values for `RH time` = 120s. Such a result should be expected, due to the larger number of bit "flips" observed at `RH time` = 120s in comparison to `RH time` = 60s.

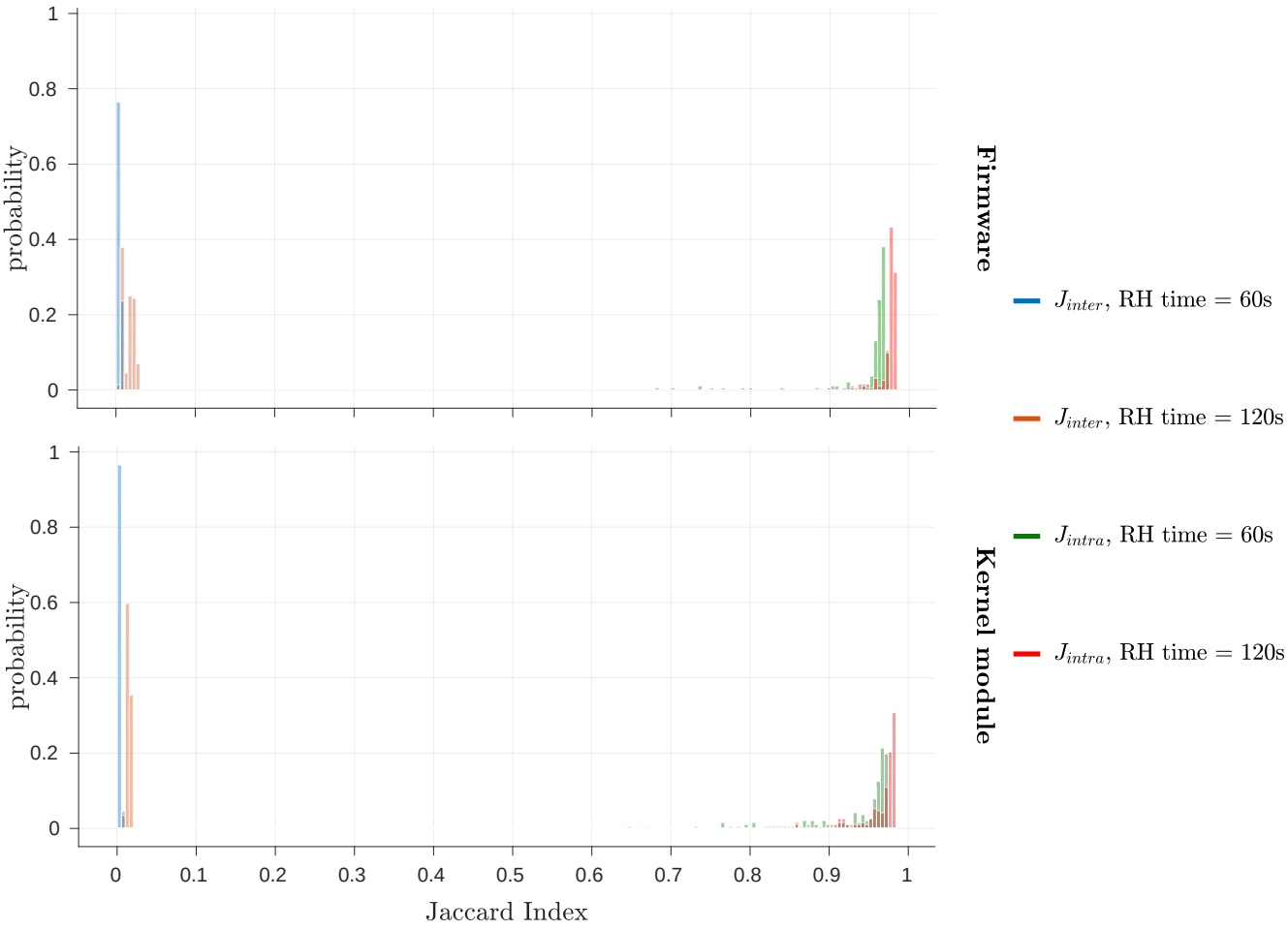


Figure 4.6: Histogram of J_{inter} and J_{intra} values for the firmware and kernel module implementations, under nominal conditions, using 20 measurements for each case of different combinations of `RH type`, `PUF row IV`, `hammer row IV`, Cache state, and (`RH time` = 60s/`RH time` = 120s), for `PUF size` = 128KB, according to Table 4.1.

Furthermore, Figure 4.7 and Figure 4.8 present in more detail the J_{intra} and J_{inter} values for all cases considered, for the firmware and the kernel module implementation, respectively. These two Figures also clearly indicate that for both implementations, and for both `Cache` states, all the DRAM Row Hammer PUF instances can be robustly and uniquely identified. Additionally, we note that, in most cases, a few outliers exist for J_{intra} values, which could potentially be ignored.

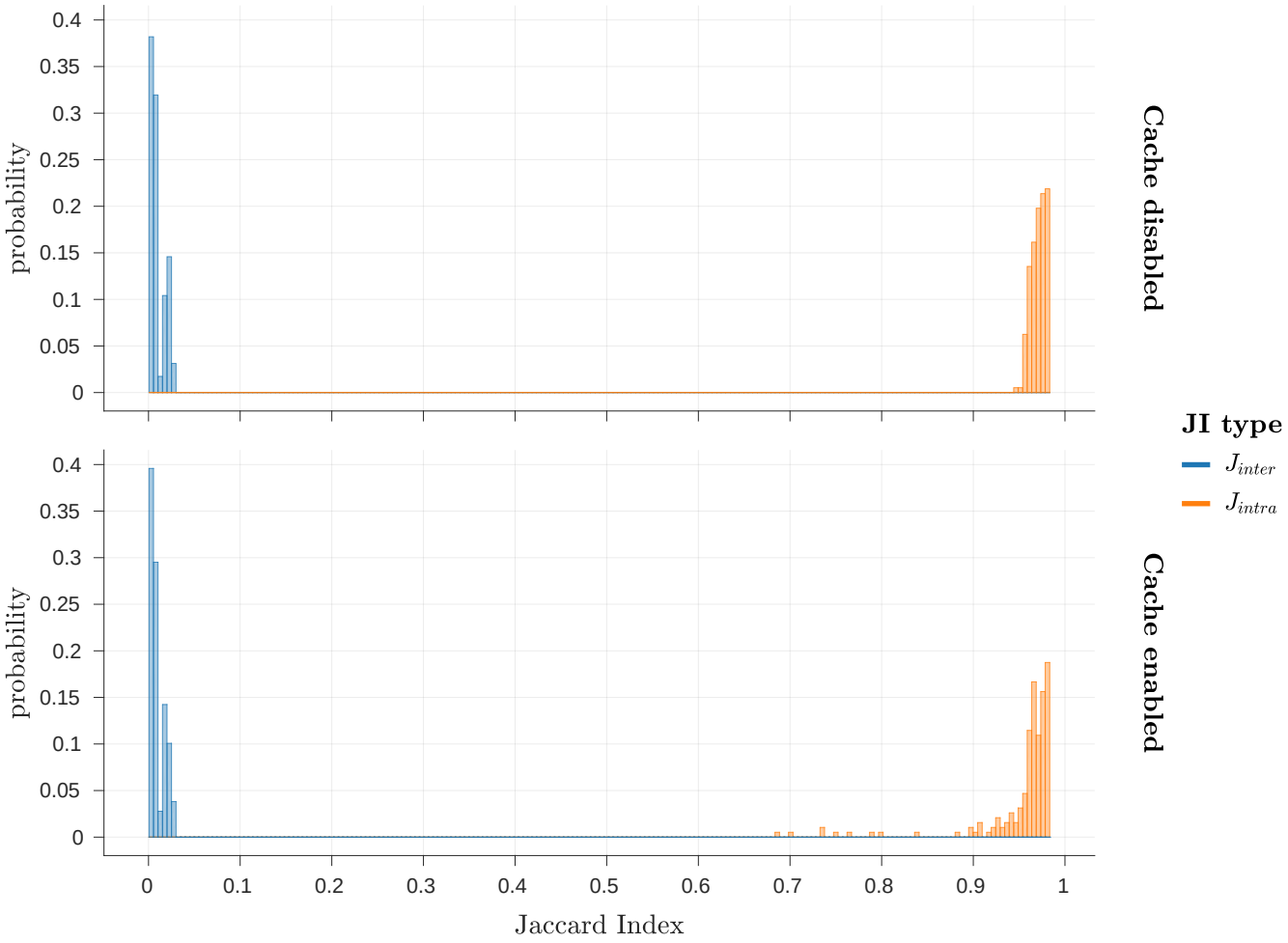


Figure 4.7: Histogram of J_{inter} and J_{intra} values for the firmware implementation, under nominal conditions, using 20 measurements for each case of different combinations of `RH type`, `PUF row IV`, `hammer row IV`, `RH time`, and (`Cache` = 'enabled'/`Cache` = 'disabled'), for `PUF size` = 128KB, according to Table 4.1.

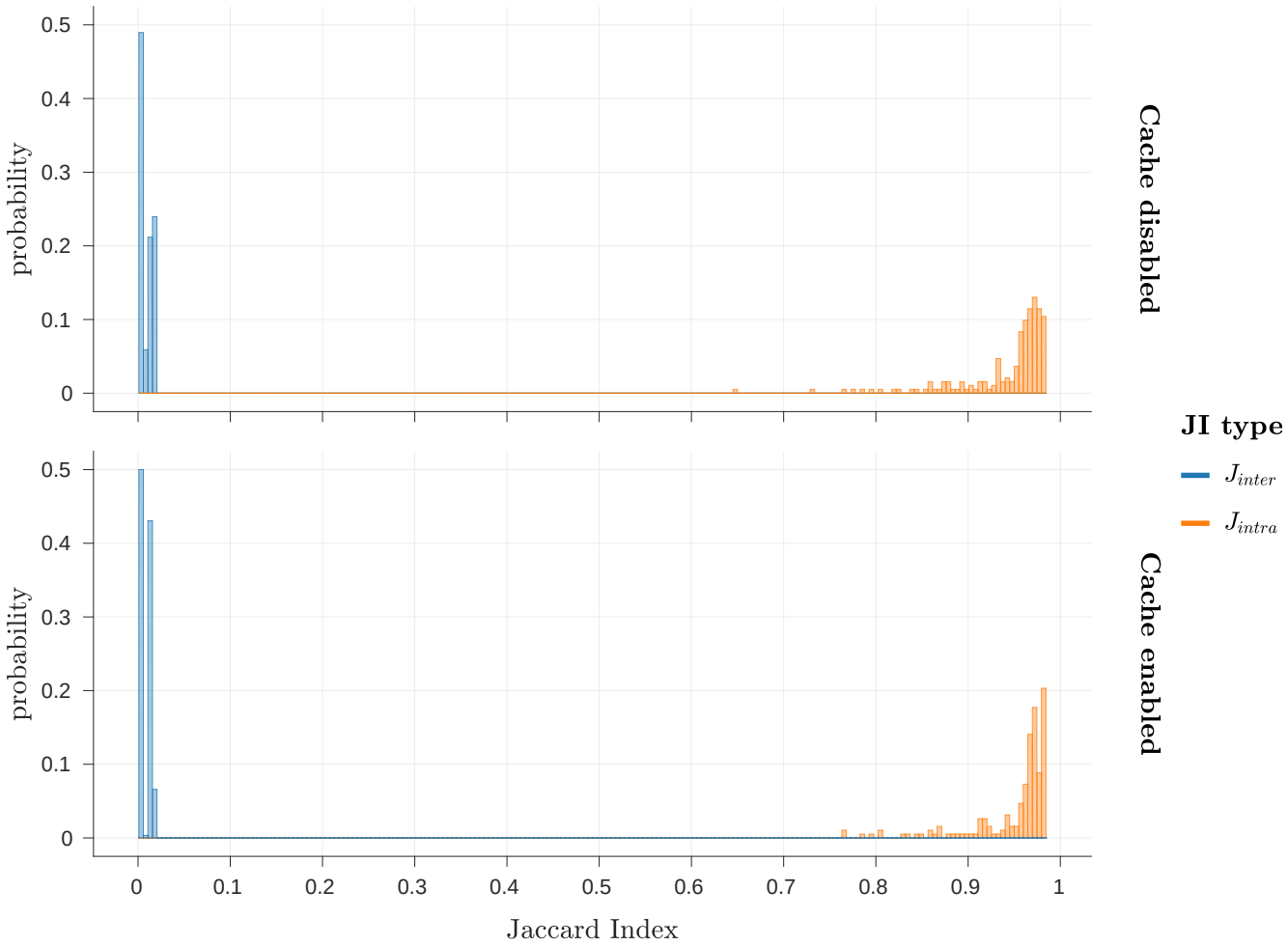


Figure 4.8: Histogram of J_{inter} and J_{intra} values for the kernel module implementation, under nominal conditions, using 20 measurements for each case of different combinations of `RH type`, `PUF row IV`, `hammer row IV`, `RH time`, and (`Cache` = 'enabled'/`Cache` = 'disabled'), for `PUF size` = 128KB, according to Table 4.1.

In order to address the nature of these outliers, we present Figure 4.9 and Figure 4.11, which show the distributions of J_{inter} and J_{intra} values, respectively, grouped by `hammer row IV`, for different `PUF row IV`, `Cache` states, and `RH type`, for the firmware implementation, and Figure 4.10 and Figure 4.12, which show the distributions of J_{inter} and J_{intra} values, respectively, grouped by `hammer row IV`, for different `PUF row IV`, `Cache` states, and `RH type`, for the kernel module implementation. In these Figures, the values of J_{inter} and J_{intra} for different `RH time` are grouped in a single distribution.

As one can see in Figure 4.9, the lowest J_{inter} values for the firmware implementation seem to occur for `hammer row IV` = "`0xAA`", in all cases. However, Figure 4.10 indicates that J_{inter} values for the kernel module implementation seem to be similar for all `hammer row IV` values, in all cases. Furthermore, Figure 4.11 indicates that all J_{intra} values for the firmware implementation are close to 1, apart from some values for `RH type` = `SSRH`, with the cache operation enabled. On the contrary, Figure 4.12 shows that J_{intra} values for the kernel module implementation are more noisy in all cases, with the highest J_{intra} values for this implementation occuring for `hammer row IV` = "`0xAA`", in all cases.

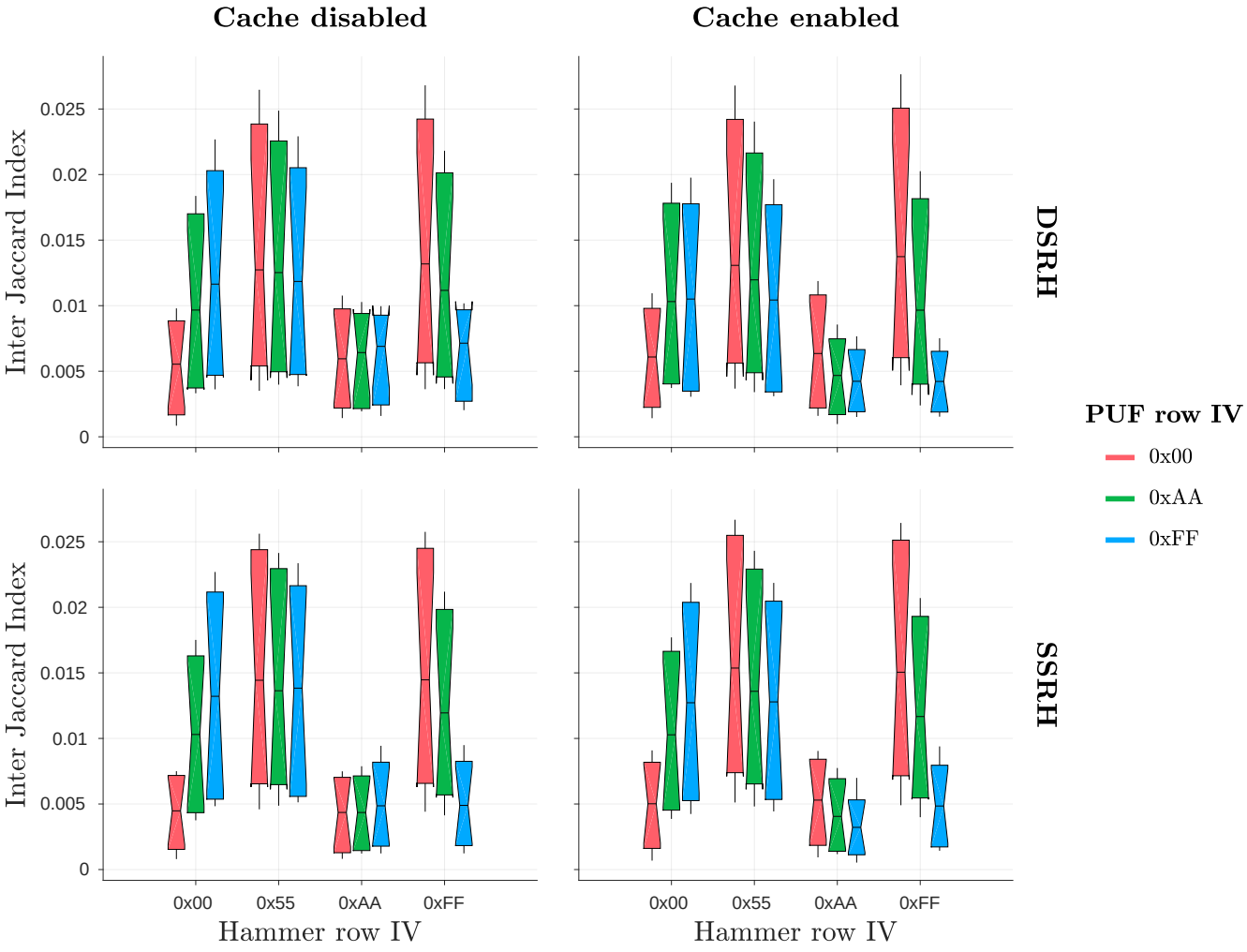


Figure 4.9: Distributions of J_{inter} values, grouped by `hammer row IV`, for different `PUF row IV`, Cache states and `RH type`, for the firmware implementation under nominal conditions. J_{inter} values for `RH time` $=$ 60s and `RH time` $=$ 120s are grouped in a single distribution, per case.

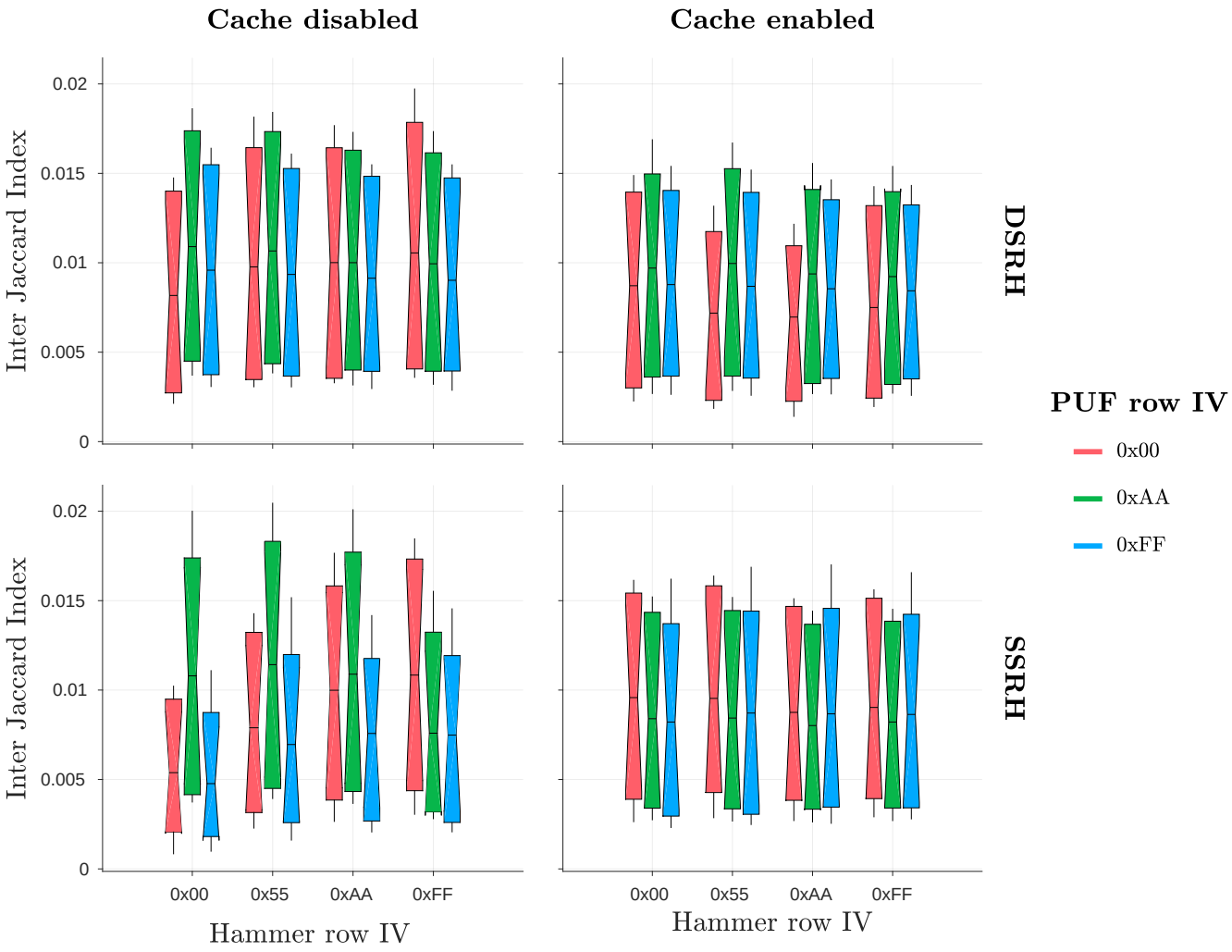


Figure 4.10: Distributions of J_{inter} values, grouped by `hammer row IV`, for different `PUF row IV`, Cache states and `RH type`, for the kernel module implementation under nominal conditions. J_{inter} values for `RH time` = 60s and `RH time` = 120s are grouped in a single distribution, per case.

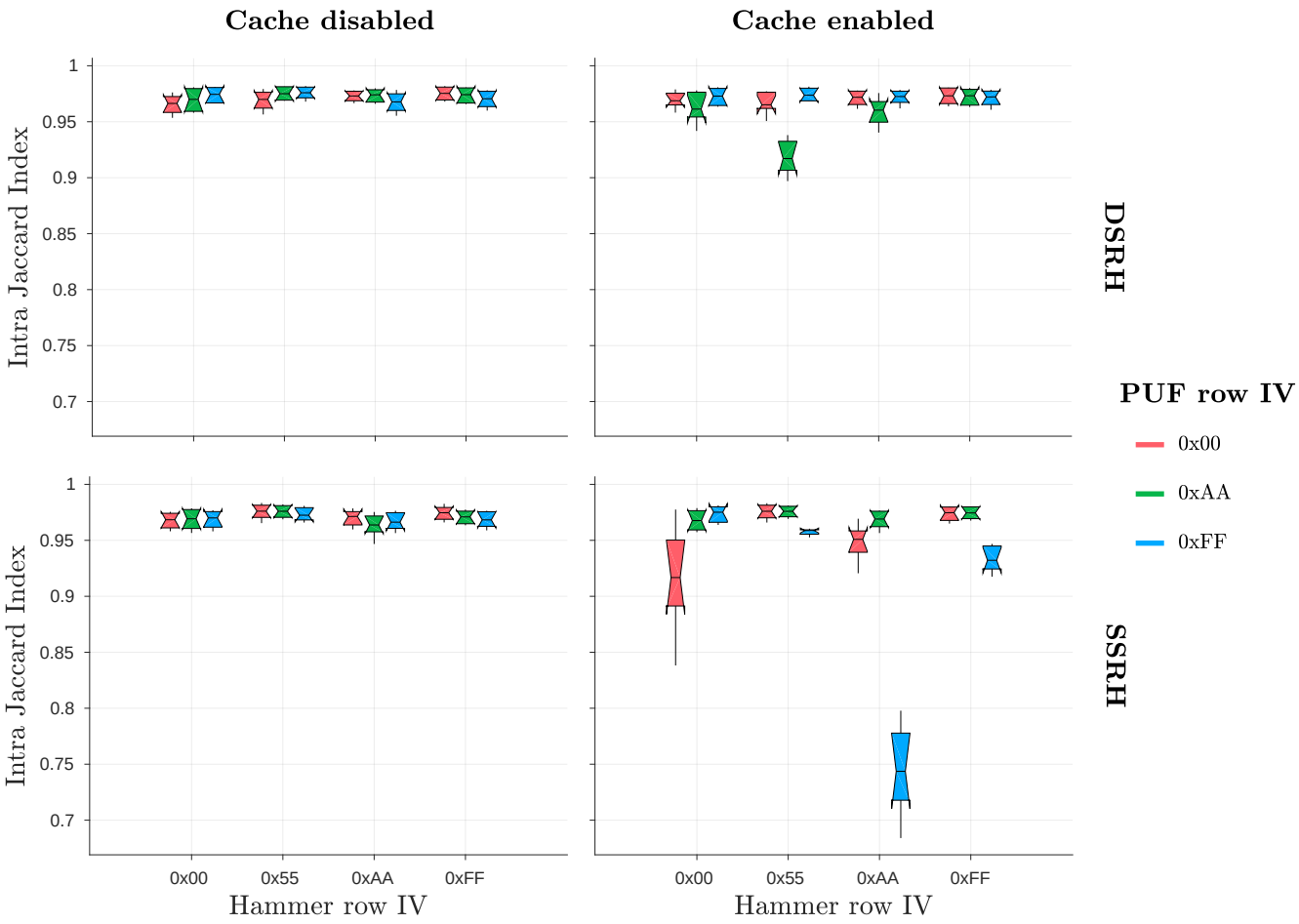


Figure 4.11: Distributions of J_{intra} values, grouped by `hammer row IV`, for different `PUF row IV`, Cache states and `RH type`, for the firmware implementation under nominal conditions. J_{intra} values for `RH time` $=$ 60s and `RH time` $=$ 120s are grouped in a single distribution, per case.

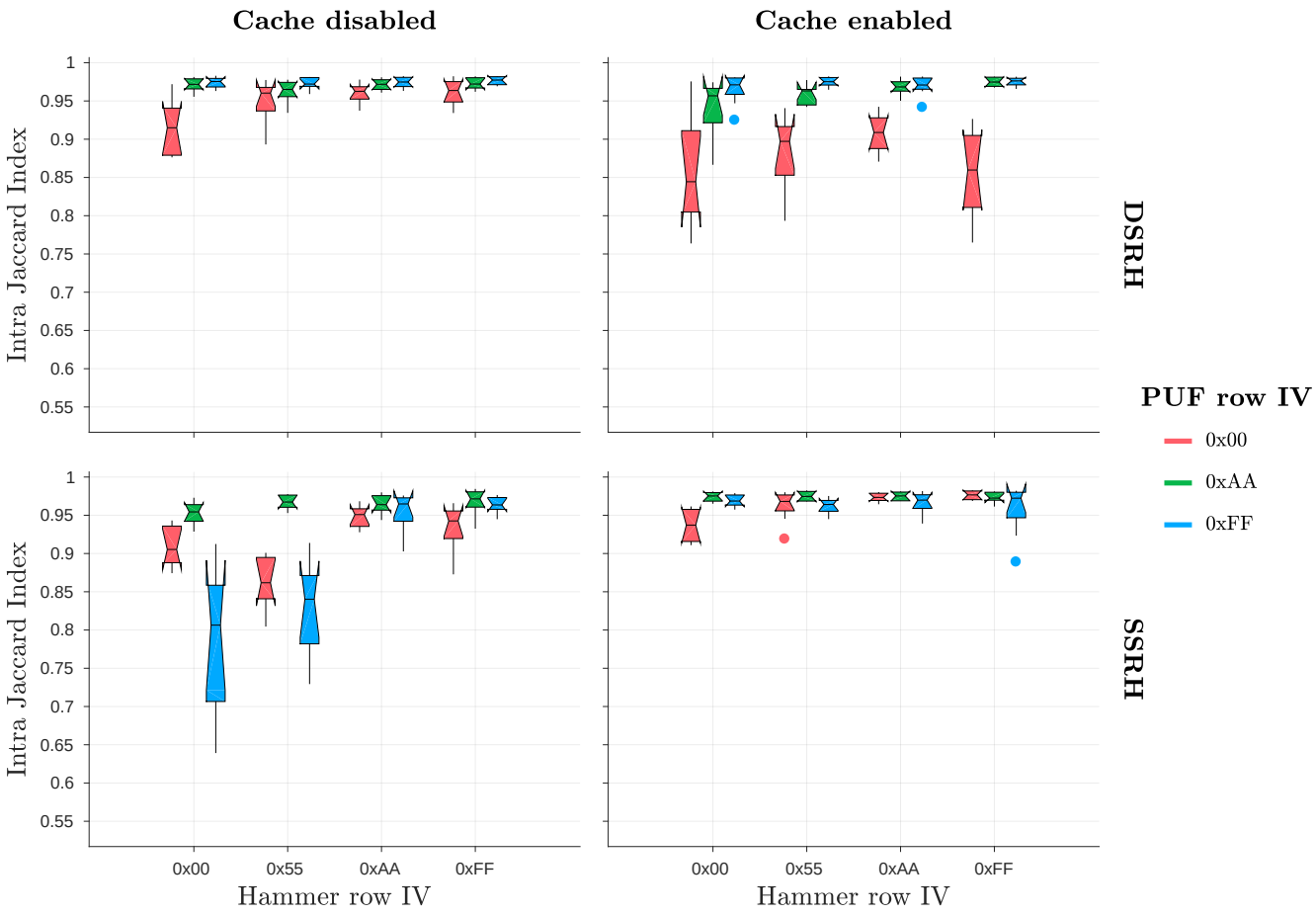


Figure 4.12: Distributions of J_{intra} values, grouped by `hammer row IV`, for different `PUF row IV`, Cache states and `RH type`, for the kernel module implementation under nominal conditions. J_{intra} values for `RH time` = 60s and `RH time` = 120s are grouped in a single distribution, per case.

Figure 4.11 and Figure 4.12 both include values near 0.7, which do not appear to be clear outliers, indicating that the usual error correction schemes may not be applicable for the stabilisation of all the responses of the firmware and kernel module implementations. We, therefore, propose the application of the helper data scheme proposed by Schaller et al. [87], which is based on bit selection, for cryptographic applications utilising such PUF responses.

Nevertheless, for `PUF row IV` = "`0xAA`", the minimum J_{intra} values seem to be over 0.9, in almost all cases, and therefore, their noise can be easily corrected by standard fuzzy extractor schemes [16, 76, 136, 137, 138, 139]. In general, however, as these responses are heavily biased towards the initial pattern written to the PUF rows, it would be rather advisable, for reasons of cost, to again employ the helper data scheme proposed by Schaller et al. [87], which is based on bit selection, rather than a fuzzy extractor scheme that incorporates error correction, in order to utilise the responses of the DRAM Row Hammer PUF in relevant cryptographic applications. At the same time, however, we do observe that the responses produced by this PUF appear to be rather robust and highly unique, under nominal conditions. Nevertheless, the presence of some dissimilarity and outliers in the values of the intra-device and the inter-device Jaccard index for measurements that were taken under nominal conditions, and which may correspond to the same, or to a different, set of relevant parameters, rather portend the presence of ambient temperature variations and their effects, as we will discuss in Section 4.2.2, because

the relevant PUF responses have been collected at room temperature, which may not have been constant, but rather could have fluctuated a little.

4.1.3 Flash-Memory-Based PUFs Under Nominal Conditions

The ability of Flash-memory-based PUFs to provide adequate security under nominal conditions has been examined in a number of relevant works [70, 89, 161, 162, 191]. In this work, as already mentioned, we will focus on NAND-Flash-memory-based PUFs that utilise programming disturbances. In particular, we will examine the quality of the responses of such Flash-memory-based PUFs that have been implemented on multiple instances of the Waveshare NandFlash Board (A), which is a removable external Flash memory module that incorporates a 1-Gigabit (Gbit) Samsung K9F1G08U0E NAND Flash memory.

Following the example of previous works regarding NAND-Flash-memory-based PUFs that utilise programming disturbances [70, 161, 162], we also assume that each individual page of the relevant NAND Flash memory that is affected by the rapidly repeated programming of its nearby pages may constitute a different instance of this PUF, and examine the quality characteristics of the values of the cells of each such page after the operation of this PUF. However, one could also potentially consider that all such pages found in a single block, or in a particular device, constitute one single such PUF. Nevertheless, in this work, we assume that each odd-numbered page [32] of a single block of the Samsung K9F1G08U0E NAND Flash memory found on the Waveshare NandFlash Board (A) constitutes an individual PUF instance. Every even-numbered page of each relevant block is considered a "hammer" page and is to be constantly, i.e., rapidly and repeatedly, (re)programmed during the operation of this PUF. As each memory block of the Samsung K9F1G08U0E NAND Flash memory is made up of 64 pages, 32 of these pages will constitute individual PUF instances, with 31 of them (pages '1', '3', '5', . . . , '61') receiving disturbances from both of their adjacent pages (double-side program "hammering"), and one of them (page '63') receiving disturbances from its only adjacent page (single-sided program "hammering", from page '62'). For each page constituting a PUF instance, 20 responses have been received of a size of 2 KB each [33], for a total size of 64 KB per block measurement.

Additionally, we need to note that the Waveshare NandFlash Board (A) is controlled using an ST Microelectronics STM32F429I Discovery (STM32F429I-DISC1) board, to which this Flash board has been connected using a Waveshare Open429Z-D Standard, an STM32F4 expansion/development board for the ST Microelectronics STM32F429I Discovery (STM32F429I-DISC1) board. All the measurements were performed at 20°C and at nominal voltage. A single block was fully erased and, thus, the initial bit pattern of cells utilised in the relevant PUF response was "`0xFF`" (all ones), while the other cells of the same block were rapidly and repeatedly programmed with the bit pattern "`0x00`" (all zeros), for 10,000 programming cycles, or until at least 2040 addresses (corresponding to 1 B each [34]), on each page used as a PUF, have had at least one bit "flip" each [35], whichever condition was fulfilled first.

[32] With the numbering starting from page '0'.

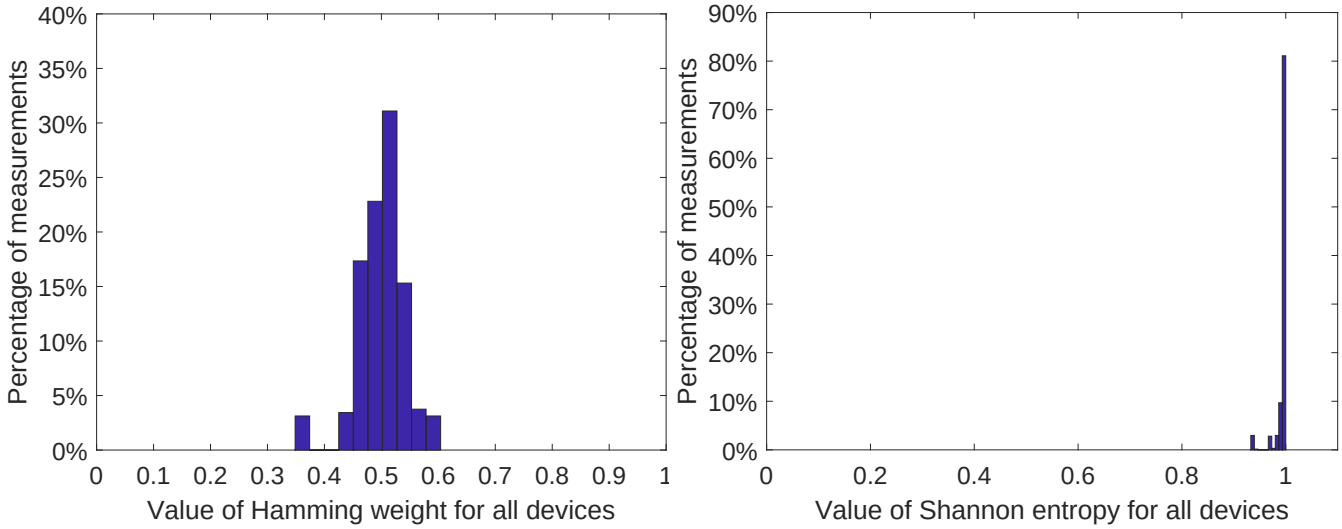
[33] Each relevant page comprises 2 KB of usable addresses and 64 B of spare addresses that are only used instead of segments ("words") of the relevant regular page addresses when these become unusable.

[34] Byte addressing is used in this memory, as is the case for all the memories examined. Therefore, each address uniquely identifies one byte, which encodes two hexadecimal numbers. The first and the last four bits of such a byte, each encode a single hexadecimal.

[35] Thus, in case this condition holds true, in each of the 2040 bytes of each relevant PUF page, which consists of 2048 bytes in total, there will be at least one bit "flip", i.e., all but one of each PUF page's bytes will contain at least one bit "flip".

Evaluation of the Entropy of the Examined Flash-Memory-Based PUF Under Nominal Conditions

We compute the binary (Shannon) entropy of the relevant PUF responses, as well as their fractional Hamming weight, in order to gain information on their entropy and, therefore, also insights into their randomness. In particular, Figure 4.13a presents the fractional Hamming weight values for all the individual measurements. As it can be easily seen, some of the measurements have a fractional Hamming weight that is not very close to the ideal value of 0.5, with the minimum fractional Hamming weight observed being 0.34853 and the maximum 0.60404. However, the mean fractional Hamming weight is 0.50101. Figure 4.13b shows the binary (Shannon) entropy values for all the individual measurements. As it can be easily seen, all of the measurements have a Shannon entropy that is rather close to the ideal value of 1, with the minimum Shannon entropy observed being 0.93275 and the maximum 1. The mean Shannon entropy value is 0.99437. We can, thus, conclude that the examined PUF exhibits a high level of entropy and, therefore, also a high degree of randomness, under nominal conditions.



(a) Fractional Hamming weight **(b)** Binary (Shannon) entropy

Figure 4.13: Evaluation of the entropy of the responses of the examined Flash-memory-based PUF under nominal conditions.

Nevertheless, in order to examine further the reason behind the observed variance of the fractional Hamming weight values, we present in Figure 4.14, the fractional Hamming weight values for each measurement performed, grouped by the relevant Flash memory page, each of which constitutes an instance of the examined Flash-memory-based PUF. Although we note that these measurements show a consistent behaviour per PUF instance, with each response for the same Flash memory page having a similar Hamming weight value to any other, the responses of particular PUF instances consistently correspond to a Hamming weight value either significantly lower than the ideal value of 0.5, such as the responses corresponding to Flash memory pages '25', and '57', or significantly higher than the ideal value of 0.5, such as the responses corresponding to pages '27', and '43'. Therefore, we believe that the exclusion of these PUF instances from usage, or the consideration of multiple Flash memory pages as a single PUF instance, could provide a much lower level of variance for the Hamming weight and binary entropy values of this PUF type, and, thus, also a higher level of security in practice. In general, however, we note that the values observed for the relevant metrics, even when single pages are considered as

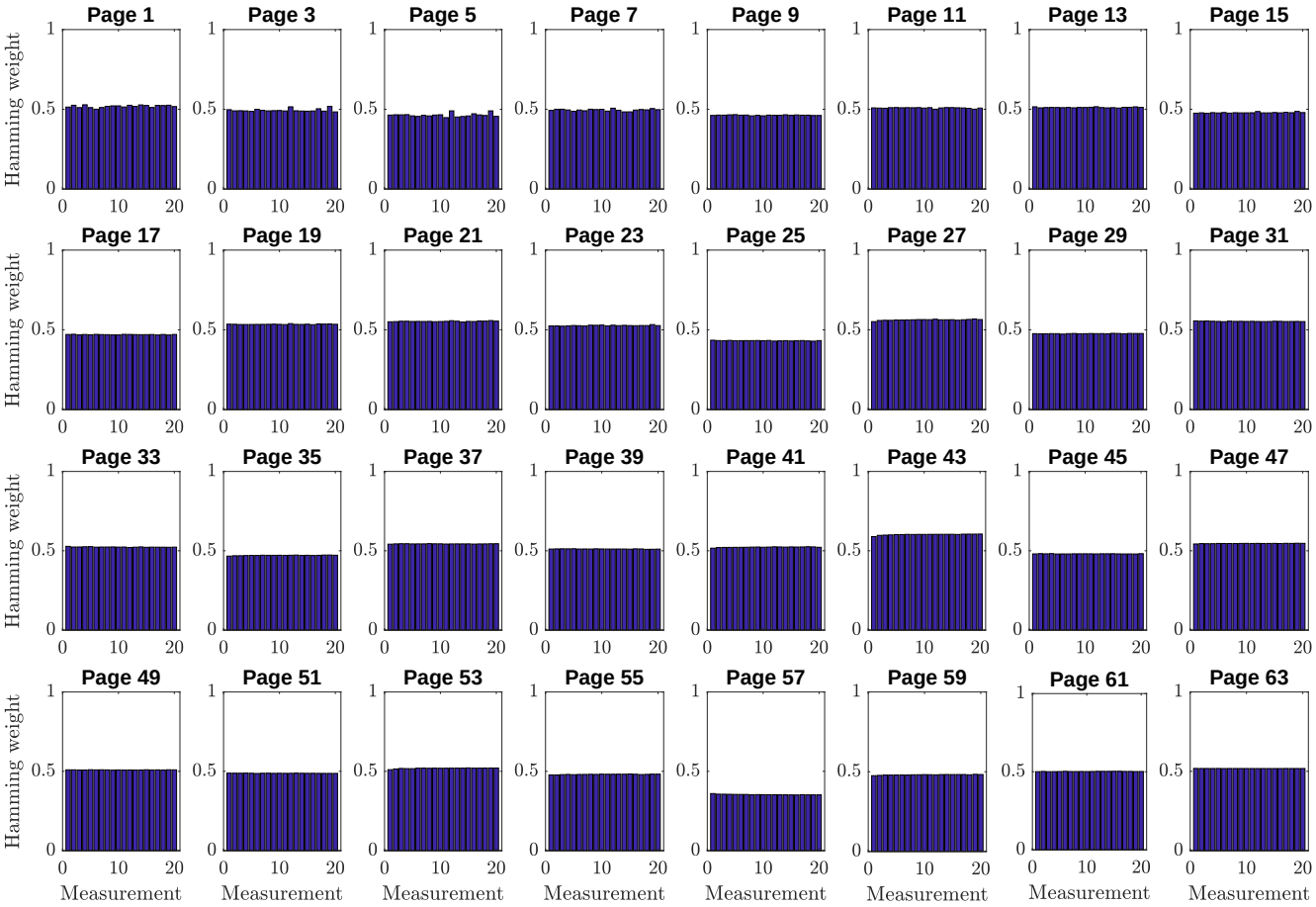


Figure 4.14: Evaluation of the Hamming weight of each response of each instance (i.e., each PUF page used as an instance) of the examined Flash-memory-based PUF under nominal conditions.

individual instances of this PUF, do not seem to preclude the employment of this PUFs as an adequate security mechanism in the context of practical IoT applications.

Evaluation of the Robustness of the Examined Flash-Memory-Based PUF Under Nominal Conditions

In order to measure the robustness of the PUF responses produced by the examined PUF, we compute the fractional intra-device [36] Hamming distance for all the measurements originating from the same Flash memory page, for all the pages being utilised as PUF instances. In this way, we can know how different are the PUF responses measured from the same PUF instance at different times and, in this way, measure their robustness. As already mentioned, PUF responses tend to be noisy, which is the reason why a (reverse) fuzzy extractor algorithm is required in order to stabilise them and, at the same time, allow for the production of a cryptographic token that is based on them.

Figure 4.15 presents the fractional intra-device Hamming distance values for all the measurements originating from the same PUF instance, for all the instances being tested. As it can be easily seen, most of the measurement pairs originating from the same memory page have a fractional intra-device Hamming distance that is rather close to the ideal values of this metric, which lie in the range between 0 and 0.1. However, we also note the presence of a very small number of significant outliers (around the

[36] Here, the term "device" is used to refer to an instance of the examined Flash-memory-based PUF, i.e., to each relevant page of this memory being utilised as a PUF.

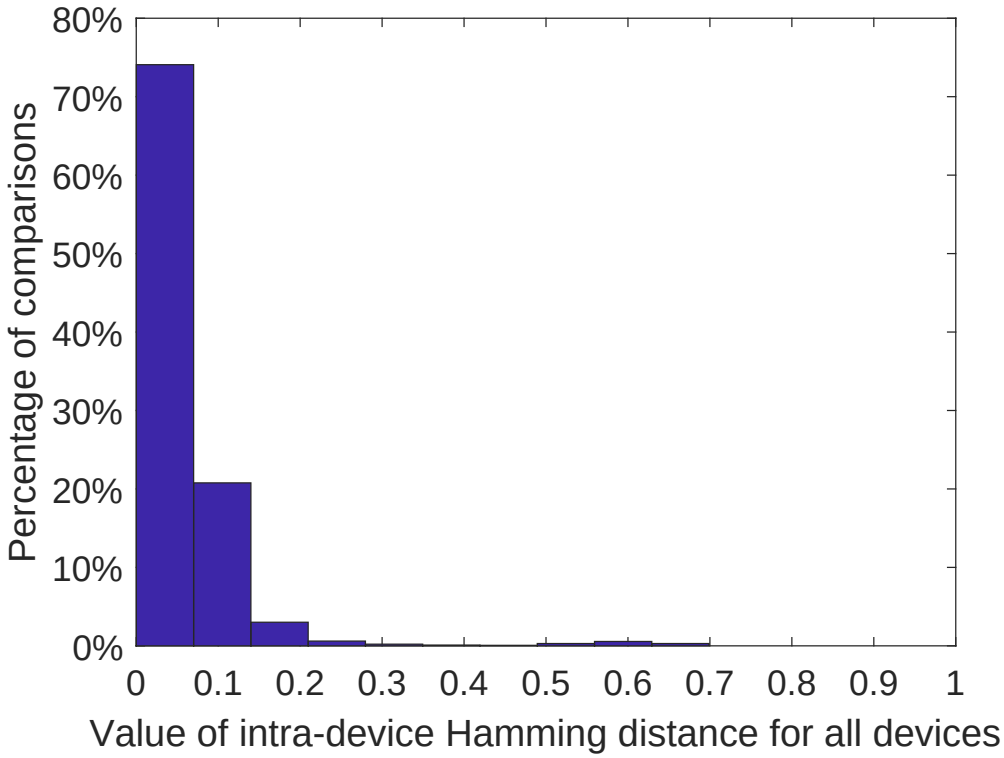


Figure 4.15: Evaluation of the robustness of the responses of the examined Flash-memory-based PUF under nominal conditions.

value of 0.6) that could not be addressed by the usual error correction schemes employed in order to deal with the inherent noise of the responses of memory-based PUFs. In particular, the minimum fractional intra-device Hamming distance observed is 0.00012, but the maximum fractional intra-device Hamming distance observed is 0.69896. Nevertheless, as the mean fractional intra-device Hamming distance is 0.05645 and the number of significant outliers, which have a fractional intra-device Hamming distance value larger than 0.2, is very small. Therefore, our results indicate that PUF responses originating from the same instance vary, on average, less than 6% and, for most cases, much less than 20%, which allows us to conclude that the examined PUF is rather robust under nominal conditions, as such levels of noise could be corrected using an error correction code, in the context of a fuzzy extractor scheme [16, 76, 136, 137, 138, 139]. Finally, we can perhaps attribute the large variance in the fractional intra-device Hamming distance values observed to the variances in the Hamming weight values among the different pages utilised as instances of the Flash-memory-based PUF examined, and to the small size of each PUF instance being utilised. Therefore, we again believe that the exclusion of particular PUF instances from usage, or the consideration of multiple Flash memory pages as a single PUF instance, could provide a much lower level of variance for the fractional intra-device Hamming distance values of this PUF type, and, thus, also a higher level of security in practice.

Evaluation of the Uniqueness of the Examined Flash-Memory-Based PUF Under Nominal Conditions

In order to test the uniqueness of the responses produced by the examined PUF, we compute the fractional inter-device [37] Hamming distance between all the pairs of responses, so that the two PUF

[37] Here, the term "device" is used to refer to an instance of the examined Flash-memory-based PUF, i.e., to each relevant page of this memory being utilised as a PUF.

responses being compared originate from different Flash memory pages, for all the pages being utilised as PUF instances. In this way, we can know how different are the PUF responses measured from different PUF instances and, therefore, also estimate their uniqueness.

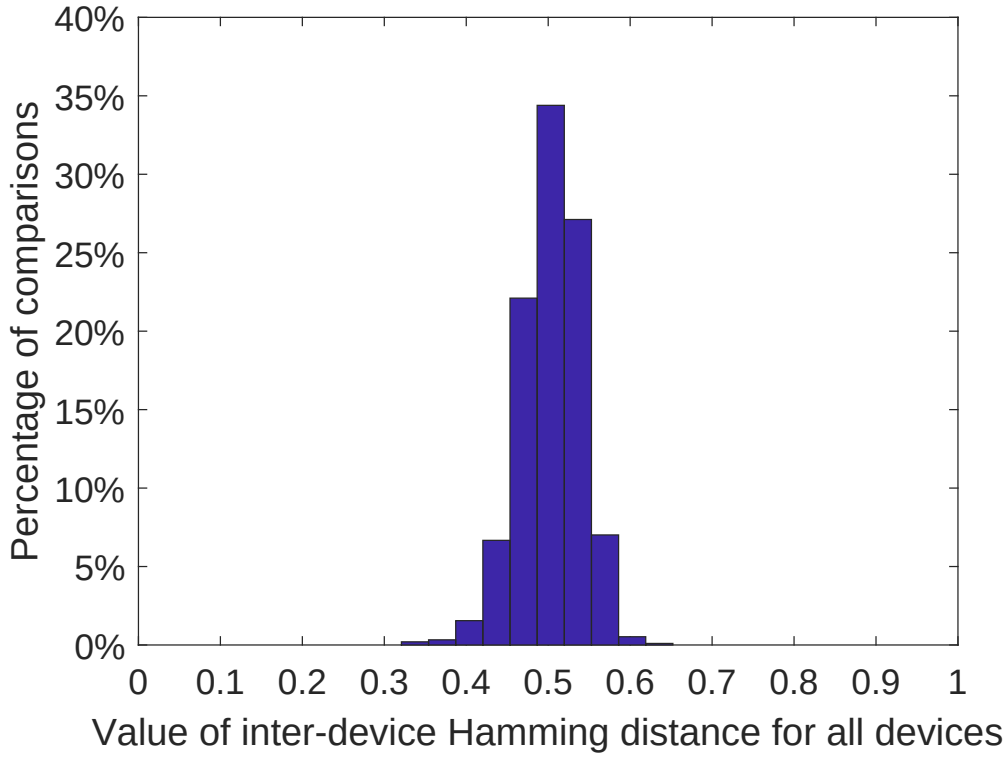


Figure 4.16: Evaluation of the uniqueness of the responses of the examined Flash-memory-based PUF under nominal conditions.

Figure 4.16 presents the fractional inter-device Hamming distance values for all the measurements originating from two different PUF instances, for all the instances being tested. As it can be seen, all of the measurement pairs have a fractional inter-device Hamming distance that is relatively close to the ideal value of 0.5, with the minimum fractional inter-device Hamming distance observed being 0.32089 and the maximum 0.65214. The mean fractional inter-device Hamming distance is 0.50329. Therefore, the PUF responses produced by the different instances of this Flash-memory-based PUF appear to be rather unique, under nominal conditions. Again, we can perhaps attribute the variance observed in the fractional inter-device Hamming distance values observed to the variances in the Hamming weight values among the different pages utilised as instances of the Flash-memory-based PUF examined, and to the small size of each PUF instance being utilised. Therefore, we again believe that the exclusion of particular PUF instances from usage, or the consideration of multiple Flash memory pages as a single PUF instance, could provide a much lower level of variance for the fractional intra-device Hamming distance values of this PUF type, and, thus, also a higher level of security in practice.

4.2 Memory-Based PUFs Under Ambient Temperature Variations

A number of works have already assessed the quality of the responses of memory-based PUFs under ambient temperature variations. In particular, SRAM PUFs [3, 4, 113, 145, 185, 186, 192], all types of DRAM-based PUFs [79, 80, 82, 83, 85, 86, 87, 88, 143, 144, 147, 153, 158, 192], as well as Flash-

memory-based PUFs [89, 147, 161, 162, 191, 193], have been shown to be mildly or even heavily affected by variations of the relevant ambient temperature. Therefore, we can conclude that most, if not all, memory-based PUFs are somewhat dependent on the ambient temperature and are affected by changes of its value.

However, we note that the degree to which the quality of the responses of each PUF is influenced by ambient temperature variations depends upon the type of the relevant PUF and the exact characteristics of the memory module that has been used for its implementation. Additionally, as it appears highly possible that ambient temperature variations can be utilised for the implementation of practical attacks against memory-based PUFs, we need to investigate further the quality characteristics of the responses of such PUFs under ambient temperature variations, in order to not only examine the degree to which such variations can affect the ability of such PUFs to act as adequate security mechanisms, but also in order to determine whether such variations can indeed be utilised for the implementation of simple attacks against such PUFs in practice.

In particular, we will examine the quality characteristics of SRAM PUFs, DRAM decay-based PUFs, DRAMs Row Hammer PUFs, and NAND Flash memory programming-disturbance-based PUFs under ambient temperature variations. In this way, we will explore how ambient temperature variations affect PUFs implemented in each of the three different types of widely used memory (SRAM, DRAM, and Flash memory), in order to assess whether relevant *practical* attacks based on these variations are possible, and to what degree it is probable that they will be successful, and, thus, determine whether these PUFs can provide an *acceptable* level of security under ambient temperature variations.

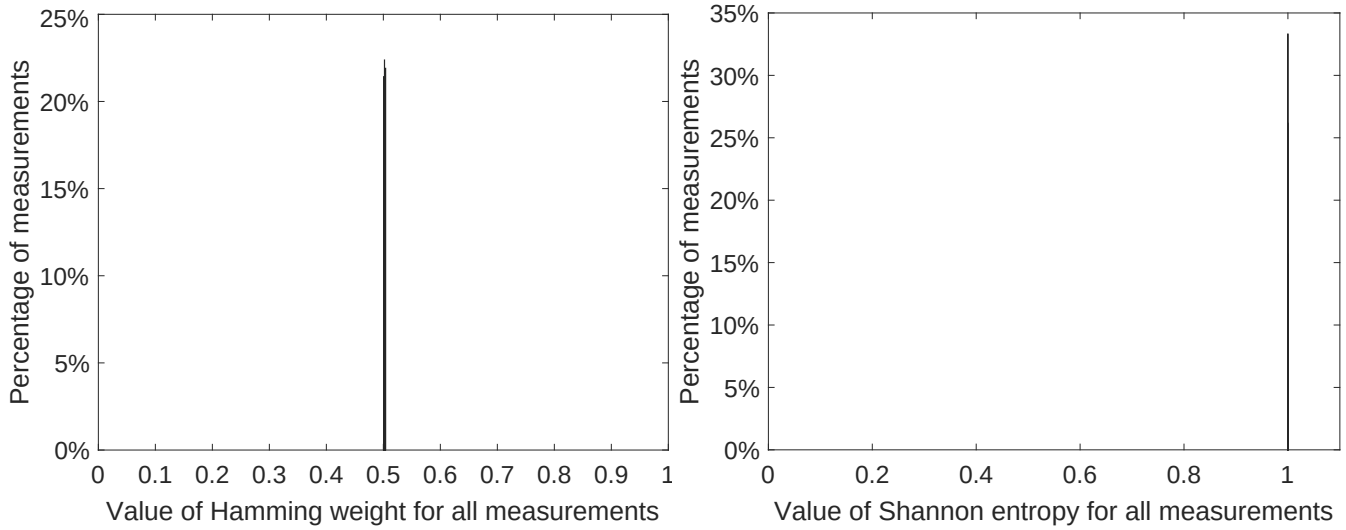
4.2.1 SRAM PUFs Under Ambient Temperature Variations

As already mentioned, the effects of ambient temperature variations on SRAM PUFs have been examined in a number of relevant works. In general, most works note that ambient temperature variations in temperature ranges above 0°C do not significantly affect such PUFs [3, 4, 113, 145, 185, 186, 192]. However, at lower temperatures, data remanence starts to become evident, at least when the relevant device is powered off for short periods of time [3, 4, 192]. Therefore, in this work, we will first investigate the effects of ambient temperature variations in the temperature range between 0°C and 60°C on the quality characteristics of the responses of SRAM PUFs implemented on the "blob_uc_code" memory region of the Qualcomm Atheros QCA9500 wireless communication module of the TP-Link Talon AD7200 router. To this end, a Heraeus Vötsch HC 4005 climate chamber has been utilised, and three TP-Link Talon AD7200 devices have been tested, on which an SRAM PUF has been implemented, as already described, on the Qualcomm Atheros QCA9500 FullMAC IEEE 802.11ad Wi-Fi chip of each of these routers. The relevant SRAM PUFs have been tested in the temperature range between 0°C and 60°C at intervals of 10°C. From each device, 10 PUF responses were recorded at each temperature, in order to study the effects of temperature variations on their entropy, robustness and uniqueness, using the metrics discussed in Section 3.2, namely the Hamming weight, the binary (Shannon) entropy, and the intra-device and inter-device Hamming distances. We note that a previous work [145] has shown, using the same methodology, that such temperature variations do not appear to have a significant effect on the responses of SRAM PUFs implemented using the "blob_fw_code" memory region of this wireless communication module of the TP-Link Talon AD7200 router. Subsequently, we also examine how lower temperatures

can induce short-term data remanence effects on SRAM PUFs, utilising relevant measurements in the temperature range between −110°C and 25°C for two instances of an SRAM PUF implemented on the on-die SRAMs of two LX4F120H5QR MCUs, each of which was inherently found in a Texas Instruments Stellaris LM4F120 LaunchPad evaluation board (EK-LM4F120XL). In this case, the relevant SRAM PUFs have been tested in the temperature range between −110°C and −40°C at intervals of 10°C, as well as at 0°C, and at 25°C. In this case, pressurised air flowing through a heat exchanger placed inside a cryogenic storage dewar vessel of liquid nitrogen was utilised for temperature control, which therefore was not absolute. Therefore, some of our measurements were performed at temperatures that may vary ±2.5°C from the reported temperature.

Examining the Entropy of the PUF Implemented on the "blob_uc_code" Memory Region of the Qualcomm Atheros QCA9500 Wireless Communication Module of the TP-Link Talon AD7200 Router Under Ambient Temperature Variations in the Temperature Range Between 0°C and 60°C

In order to evaluate how temperature variations affect the entropy and, therefore, also the randomness of the responses of the examined PUF, we compute the binary (Shannon) entropy for all the PUF responses collected in the temperature range between 0°C and 60°C, as well as their fractional Hamming weight. Figure 4.17a presents the fractional Hamming weight values for all PUF responses recorded in the temperature range between 0°C and 60°C. As it can be seen, all the measurements have a fractional Hamming weight extremely close to the ideal value of 0.5, with the minimum fractional Hamming weight observed being 0.50027 and the maximum 0.50417. The mean fractional Hamming weight is 0.50225. Figure 4.17b shows the binary (Shannon) entropy values of all PUF responses recorded in the temperature range between 0°C and 60°C. As it can be easily seen, all of the measurements have a Shannon entropy extremely close to the ideal value of 1, with the minimum Shannon entropy observed



(a) Fractional Hamming weight **(b)** Shannon entropy

Figure 4.17: Evaluation of the entropy of all PUF responses recorded in the temperature range between 0°C and 60°C, in order to examine the effects of temperature variations on the SRAM PUF responses.

being 0.99995 and the maximum 1. The mean Shannon entropy value is 0.99998. We can, therefore, conclude that the entropy of the PUF responses is not affected by temperature variations.

Examining the Robustness of the PUF Implemented on the "blob_uc_code" Memory Region of the Qualcomm Atheros QCA9500 Wireless Communication Module of the TP-Link Talon AD7200 Router Under Ambient Temperature Variations in the Temperature Range Between 0°C and 60°C

In order to assess how temperature variations affect the robustness of the responses of the examined PUF, we compute the fractional intra-device Hamming distance for all the PUF responses collected from the same device in the temperature range between 0°C and 60°C, for all the devices being tested. Figure 4.18 indicates that, in this case, the minimum fractional intra-device Hamming distance is 0 and the maximum 0.064219. The mean fractional intra-device Hamming distance is 0.045913. We note that these results not only are within the ideal range of values between 0 and 0.1, but also indicate more robust PUF responses than the original measurements performed at room temperature. We attribute this lack of outliers, which are present in the measurements shown in Figure 4.2, potentially, to the smaller number of devices being tested, or to smaller deviations in the experimental conditions, as the temperature was not fully controlled in the measurements performed at room temperature. In any case, however, our results suggest that temperature variations do not affect the robustness of the responses of the examined PUF.

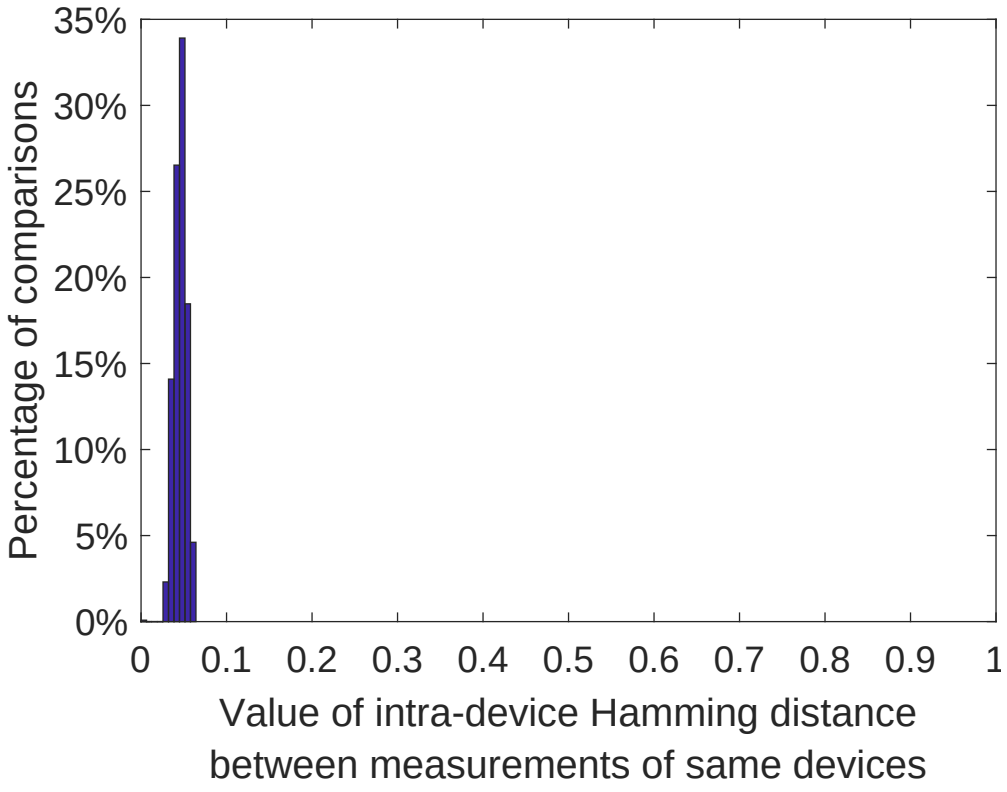


Figure 4.18: Evaluation of the robustness of all PUF responses recorded in the temperature range between 0°C and 60°C, in order to examine the effects of temperature variations on the SRAM PUF responses.

Examining the Uniqueness of the PUF Implemented on the "blob_uc_code" Memory Region of the Qualcomm Atheros QCA9500 Wireless Communication Module of the TP-Link Talon AD7200 Router Under Ambient Temperature Variations in the Temperature Range Between 0°C and 60°C

Similarly to the previous paragraph, in order to examine the effects of temperature variations on the uniqueness of the PUF responses, we compute the fractional inter-device Hamming distance for all pairs of measurements coming from two different devices, for all measurements taken in the temperature range between 0°C and 60°C and for all the devices being tested. Figure 4.19 indicates that, in this case, the minimum fractional inter-device Hamming distance is 0.49715 and the maximum 0.50020. The mean fractional inter-device Hamming distance is 0.49887. We note that these results are extremely close to the ideal value of 0.5 and, therefore, conclude that temperature variations do not affect the uniqueness of the responses of the examined PUF.

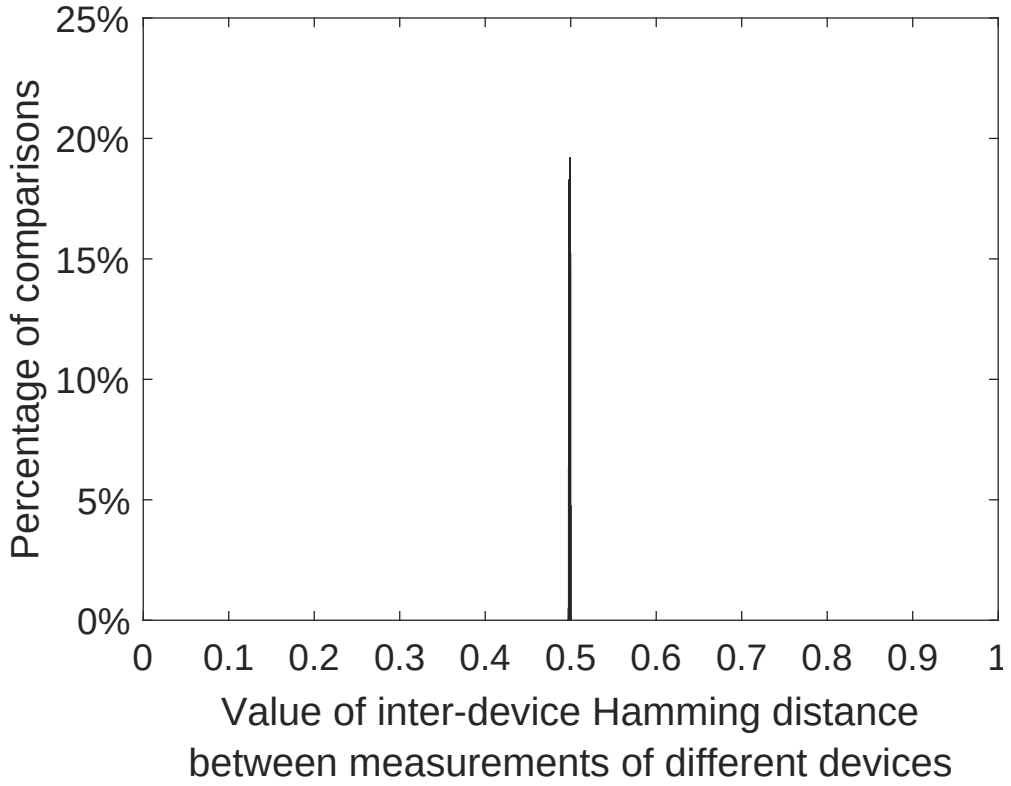


Figure 4.19: Evaluation of the uniqueness of all PUF responses recorded in the temperature range between 0°C and 60°C, in order to examine the effects of temperature variations on the SRAM PUF responses.

Examining the Data Remanence Effects Caused by Ambient Temperature Variations in the Temperature Range Between −110°C and −40°C on the SRAM PUFs Implemented on the On-Die SRAMs of the LX4F120H5QR MCUs of the Texas Instruments Stellaris LM4F120 LaunchPad Evaluation Board (EK-LM4F120XL)

While the quality of the responses of the PUF implemented on the "blob_uc_code" memory region of the Qualcomm Atheros QCA9500 wireless communication module of the TP-Link Talon AD7200 router does not seem to be affected by ambient temperature variations in the temperature range between 0°C

and 60°C [38], it has been shown in the relevant literature [194, 195, 196, 197, 198, 199, 200, 201, 202, 203] that SRAM is affected by low temperatures, which cause the appearance of data remanence effects. As such data remanence effects can significantly limit the ability of SRAM PUFs to act as adequate security mechanisms [3, 4, 5, 6, 192, 204, 205], we will examine such data remanence effects caused on the SRAM PUF implemented on the on-die SRAMs of the LX4F120H5QR MCUs of the Texas Instruments Stellaris LM4F120 LaunchPad evaluation board (EK-LM4F120XL) by low temperatures, in the temperature range between −110°C and −40°C, as well as relevant attacks and countermeasures.

In this work, we have employed the notion of a well-known metric, i.e., the Hamming distance, in order to measure the level of data remanence exhibited by the SRAMs of the Texas Instruments Stellaris LM4F120 LaunchPad evaluation boards (EK-LM4F120XL) being tested. By calculating the Hamming distance between the known pattern that we write to the SRAM cells before powering off the Texas Instruments Stellaris LM4F120 LaunchPad evaluation boards (EK-LM4F120XL) and their values after start-up, we can easily calculate how many SRAM cells have kept the value that was written to them before power-off and get a good measure of the data remanence. Moreover, by dividing the number of cells that kept the value stored in them by the total number of SRAM cells used, for each SRAM, we can easily get the *proportion of cells that keep their values*. This *fractional Hamming distance* can also be easily presented as a *percentage* and is a good metric for the data remanence of the SRAMs of the examined boards, as it reveals what percentage of the SRAM cells have kept their values after the reboot.

However, as the two patterns that we actually write to the SRAM cells are all-zeros and all-ones, this fractional Hamming distance can also be very easily computed by calculating the percentage of cells having one of the two logical values after the reboot. We chose to measure the *percentage of cells that have the logical value '1' after the reboot*, as this metric is equivalent to the notion of the *fractional Hamming weight*. However, equivalent results can be attained by calculating the percentage of cells that have the logical value '0' after the reboot, as the two aforementioned percentages added together, are equal to 100%.

If the SRAM cells have all been set to the logical value of '1' before power-off, the percentage of cells that have the logical value '1' after the reboot is the same as the percentage of cells that have kept their values and, therefore, immediately reveals the level of data remanence. Nevertheless, even if the SRAM cells have all been set to a logical value of '0' before power-off, the percentage of cells that have the logical value '1' after the reboot is still a very good measure of data remanence, as this percentage is equal to 100% minus the percentage of cells that have kept their value after the reboot. Table 4.2 demonstrates how the metric being used can provide a measure of data remanence in both cases. The advantage of using this single metric in both cases is that not only the results of both cases can be presented together, but also that it is very easy to calculate, while its significance is not difficult to understand.

However, we need to stress here that, after a reboot, normally, around 50% of the SRAM cells will have the logical value '1' and around 50% will have the logical value '0'. It is exactly this property of the SRAM cells that allows an SRAM module to be utilised as a PUF, as the concatenation of the values of the

[38] We note that the PUF implemented in the Qualcomm Atheros QCA9500 wireless communication module of the TP-Link Talon AD7200 exhibits almost ideal characteristics, even under temperature variations, as our evaluation results show. This allows us to suggest that this PUF implementation constitutes a security primitive that can be utilised to implement lightweight, cost-efficient, and practical security solutions.

Table 4.2: Using the Percentage of SRAM Cells Having the Logical Value '1' After the Reboot (the *Fractional Hamming Weight* After the Reboot) as a Metric for Data Remanence.

Pattern written	P_1 **indicates**	P_0 **indicates**	**Data remanence indicator**
all '1'	data remanence	data loss	P_1
all '0'	data loss	data remanence	$P_0 = 100\% - P_1$
P_1: the percentage of cells that have the logical value '1' after the reboot P_0: the percentage of cells that have the logical value '0' after the reboot			

SRAM cells after a normal reboot forms a fairly robust bit-string that is rather unique for each individual SRAM module, as it contains an almost equal amount of logical '0' and logical '1', which are, however, found in more or less different positions for each such bit-string. In practice, a slight bias towards one of the two logical values is most often observed.

In particular, we note that the two SRAM modules examined exhibited a slight bias towards the logical value '0'. On average, for each of the two SRAM modules that have been tested, $\approx 43.5\%$ of the relevant SRAM cells had the logical value '1', after a reboot under nominal conditions, and, thus, around $\approx 56.5\%$ of the SRAM cells had the logical value '0', after a reboot under nominal conditions. Nevertheless, we also need to note that SRAM cells exhibit a relative instability among different measurements, with, most often, less than 10% of the values being different between two measurements [185]. Nevertheless, as the actual error rate for the SRAM modules tested tends to be lower, with a noted maximum value of $\approx 5.25\%$ [146], and as the characteristics of the errors occurring also influence the overall bias observed in the values of the SRAM cells among different measurements, we chose to consider all our results in which the percentage of cells that had the logical value '1' after the reboot was within the nominal levels of $43.5\% \pm (10\% \times 43.5\%)$, i.e., between 39.15% and 47.85%, as indicating a state of complete data loss, i.e., a state of total lack of data remanence.

It is also important to note that the fact that not all SRAM cells will have the same logical value after a normal reboot means that, for some of the cells, the logical value that was written to them will not change after a reboot during our experiments, not due to the effects of data remanence, but due to that value being the normal start-up value of such cells. It is exactly for this reason that we choose to conduct experiments by writing both an all-zeros and an all-ones pattern to the SRAM cells, as this can indeed reveal the real level of data remanence under different conditions (ambient temperature and power-off time), as the real data remanence level of the cells that will typically contain the logical value '1' after a normal reboot can be revealed by writing a pattern of all-zeros to the SRAM before power-off, while the real data remanence level of the cells that will typically contain the logical value '0' after a normal reboot can be revealed by writing a pattern of all-ones to the SRAM before power-off. In either case, we choose not to examine the percentage of cells that seem to keep the value written to them after a reboot, not because of data remanence, but because this is their usual start-up value, as our methodology specifically addresses this bias (through having both logical values being written alternatively to all the SRAM cells), and as an attacker would not need to distinguish between the *real* data remanence level and the *observed* data remanence level.

Regarding our experimental results, we first present an overview of the experimental results obtained in the temperature range between -110°C and 0°C for the examined SRAM modules, in order to

demonstrate the observed data remanence level for the SRAM in the temperature range from −110°C to −40°C, under conditions of high thermal isolation. Then, we present further results based on additional experiments performed for the purposes of this work, in order to compare the observed levels of data remanence under conditions of high thermal isolation and of relative lack of such isolation. We also discuss not only the role of temperature on these results, but also the role of the power-off time, in order to demonstrate the importance of both factors on the observed level of data remanence.

Results of the Experiments Conducted Under High Thermal Isolation. In order to measure the data remanence of the on-die SRAMs of the LX4F120H5QR MCUs of the Texas Instruments Stellaris LM4F120 LaunchPad evaluation board (EK-LM4F120XL), which may significantly affect its operation as a PUF, we wrote a known pattern of either all-zeros or all-ones to the whole SRAM, powered off the board for a very short time interval (for either 10 or 20 milliseconds), at a very low temperature (ranging between −120°C and −40°C, at intervals of 10°C), and then powered it on again, so that we could measure the percentage of SRAM cells that had the logical value '1' after the reboot. In this way, we could easily determine the percentage of SRAM cells that had kept the value written to them before the reboot and, therefore, also the level of data remanence. It is worth noting that all experiments performed for the purposes of the data remanence study presented and discussed in this segment were performed under conditions of relative thermal isolation, as the Texas Instruments Stellaris LM4F120 LaunchPad evaluation board (EK-LM4F120XL) was placed inside a closed Expanded Polystyrene Foam (EPF) – styrofoam – box, which had only small holes that allowed the relevant Universal Serial Bus (USB) cables connected to the Texas Instruments Stellaris LM4F120 LaunchPad evaluation board (EK-LM4F120XL), as well as the copper wires connected to an external temperature sensor glued on top of its SRAM, to pass through.

The levels of data remanence observed in the SRAM PUF implemented on the on-die SRAMs of the LX4F120H5QR MCUs of the Texas Instruments Stellaris LM4F120 LaunchPad evaluation board (EK-LM4F120XL) between −110°C and −40°C were measured for 10ms and 20ms power-off times, as shown in Figures 4.20 and 4.21. As shown in Figure 4.20, for a power-off time of 10ms, the results indicated an extremely high level of data remanence at temperatures lower than −100°C, even though the board was fully functional and, thus, the SRAM was grounded. Additionally, it was noted that the data remanence decreased as the temperature increased. At 0°C and above, there was essentially no data remanence, as the results indicate that the cells returned to their usual start-up values after the device had been rebooted. We observed that, on average, more than 75% of the SRAM cells' contents were preserved even for temperatures as low as −40°C and that the variance of the data remanence results increased as the temperature increased.

As shown in Figure 4.21, the average data remanence levels for the 20ms power-off time were (for all temperature levels) at least 20% lower than those for the 10ms, while the SRAM cells had almost completely returned to their start-up values as the temperature approached −50°C and higher. This degradation becomes more noticeable in Figures 4.22 and 4.23, where the results for 10ms and 20ms power-off times at different temperatures are compared, first, for all the SRAM cells having been written with the logical value '1' (Figure 4.22) and, then, for all the cells having been written with the logical value'0' (Figure 4.23).

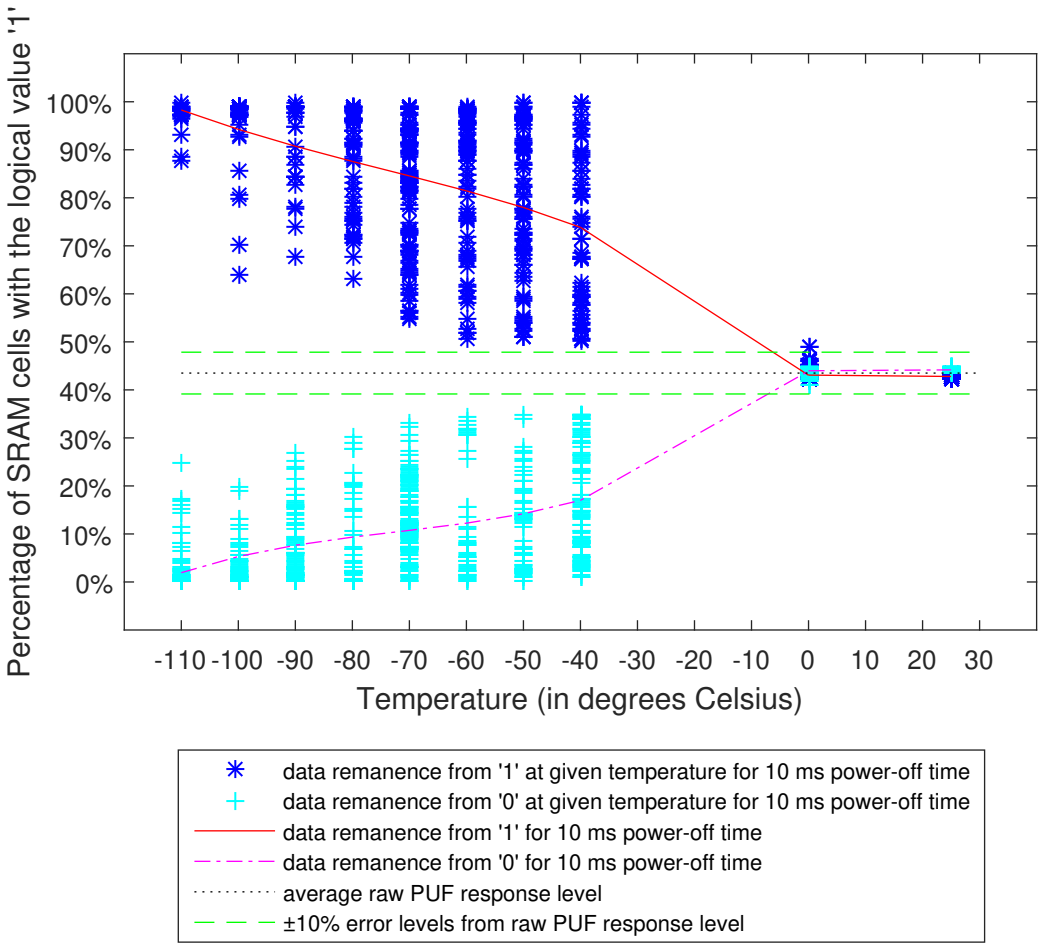


Figure 4.20: Results reflecting the level of data remanence at different temperatures for 10ms power-off time, after having written to all the cells of the on-die SRAMs of the LX4F120H5QR MCUs of the Texas Instruments Stellaris LM4F120 LaunchPad evaluation board (EK-LM4F120XL) either the logical value ‘1’ or the logical value ‘0’.

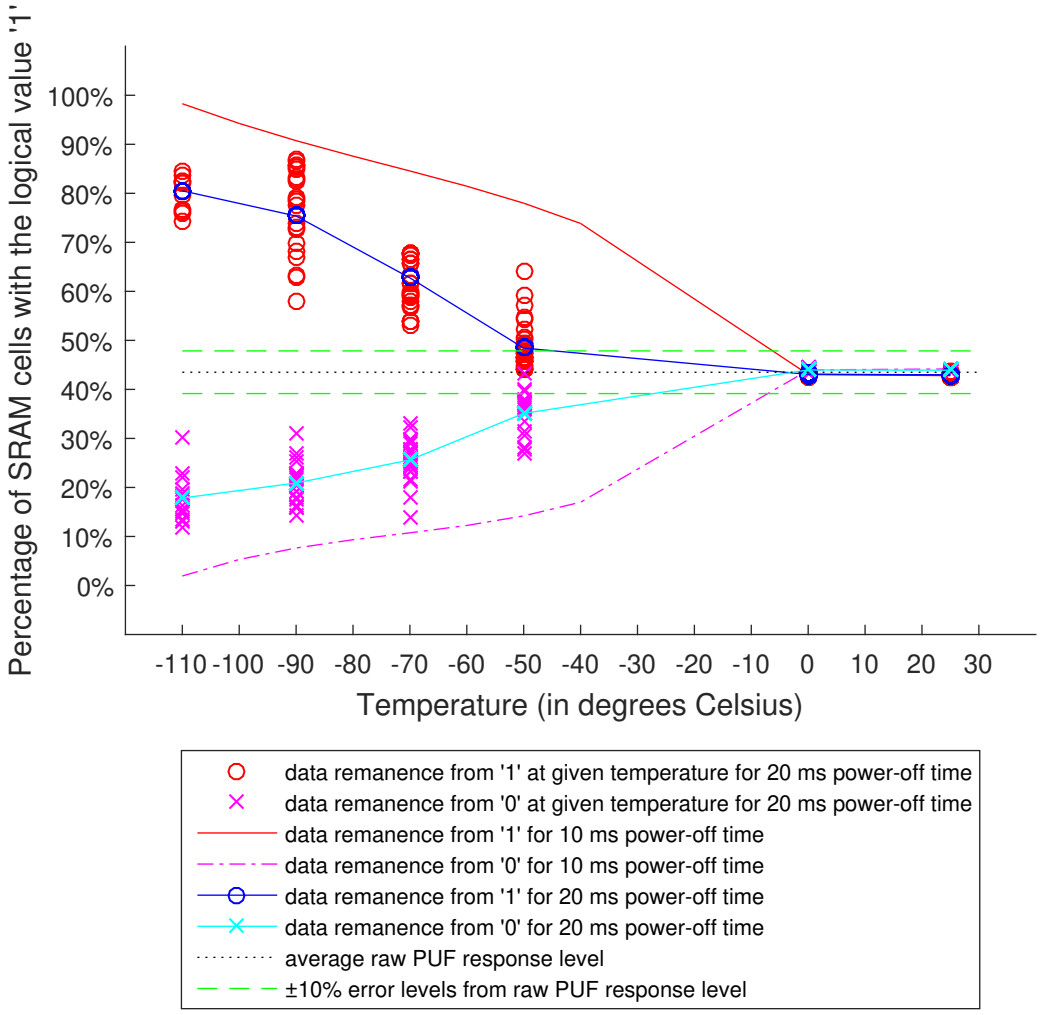


Figure 4.21: Results comparing the level of data remanence at different temperatures for 20ms power-off time to that observed at the same temperatures for 10ms power-off time, after having written to all the cells of the on-die SRAMs of the LX4F120H5QR MCUs of the Texas Instruments Stellaris LM4F120 LaunchPad evaluation board (EK-LM4F120XL) either the logical value '1' or the logical value '0'.

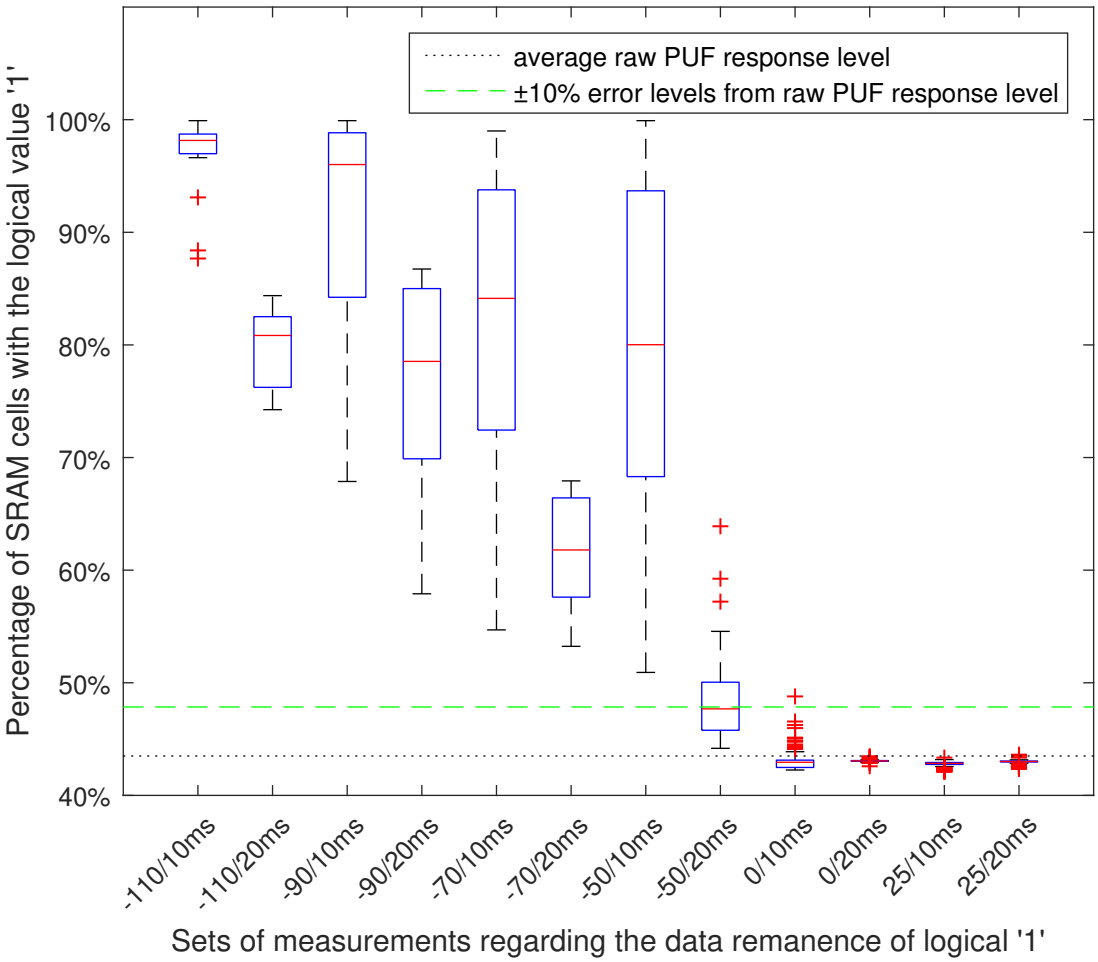


Figure 4.22: A comparison of measurements for 10ms and 20ms power-off times at different temperatures for the data remanence of the logical value '1'. In the identifier of each set, the first number denotes the temperature at which the measurement took place (in °C), and the second indicates the relevant power-off time (in ms).

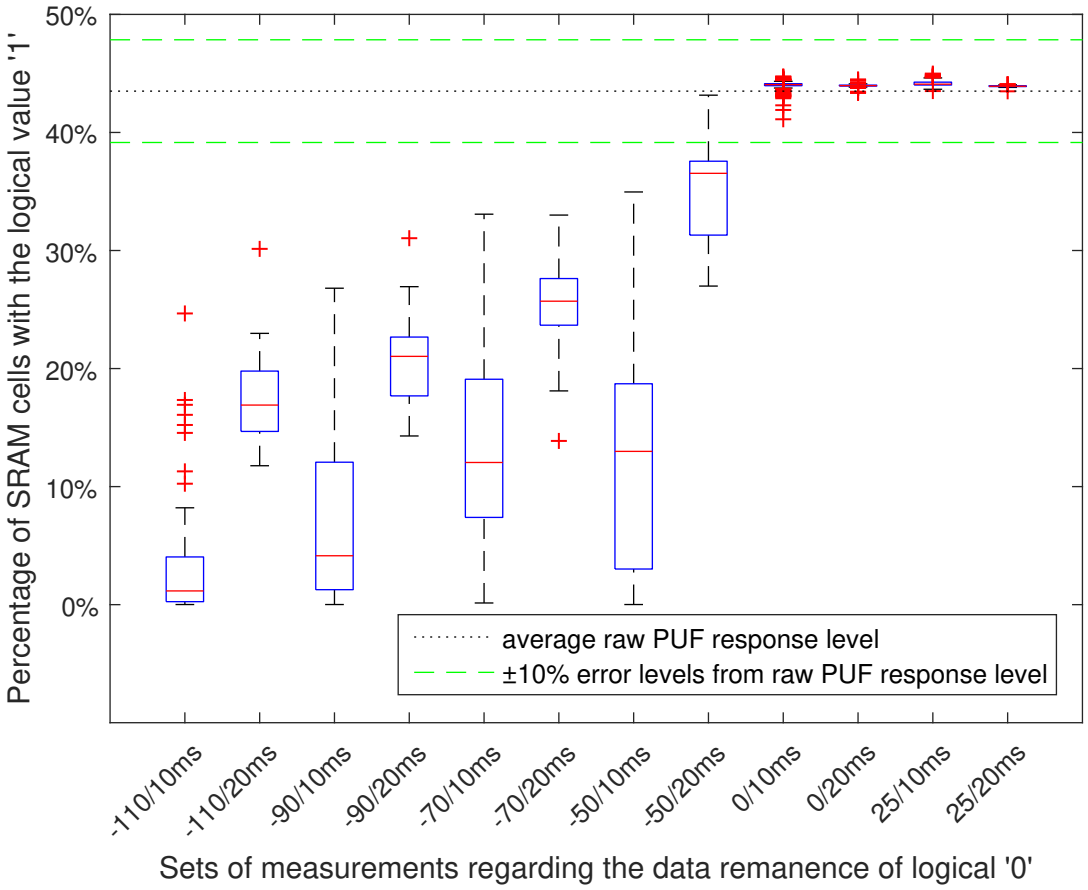


Figure 4.23: A comparison of measurements for 10ms and 20ms power-off times at different temperatures for the data remanence of the logical value '0'. In the identifier of each set, the first number denotes the temperature at which the measurement took place (in °C), and the second indicates the relevant power-off time (in ms).

Finally, it was noted that when the board was frozen to temperatures around −120°C, even higher average data remanence levels were observed. However, at this temperature, the board became unstable and either halted temporarily for some time interval, continuing its execution off where it had stopped, when it became operational again, or it became completely unresponsive for some time and subsequently rebooted. Therefore, it was noted that attacks that require a responsive SRAM PUF cannot be performed at this temperature and below. Nevertheless, at temperatures around −120°C, data remanence levels ranging between 80% and 98% were observed, even after an extended power-off time. In general, as Figure 4.24 shows, for very low temperatures, the levels of data remanence are very high, even for a power-off time of 20ms.

Comparison Between the Results for Experiments Conducted With and Without Thermal Isolation. In addition to the experimental results already presented and discussed, we have also taken further measurements using the same experimental setup as before, with the only difference being that an open-top EPF – styrofoam – box was used. By doing so, we are able to compare the effects of data remanence on the SRAMs being tested, not only under the condition of relative thermal isolation, but also when such isolation is rather lacking. In this way, we are providing a much better view on the feasibility and practicality of the low-temperature data remanence attack proposed in this work.

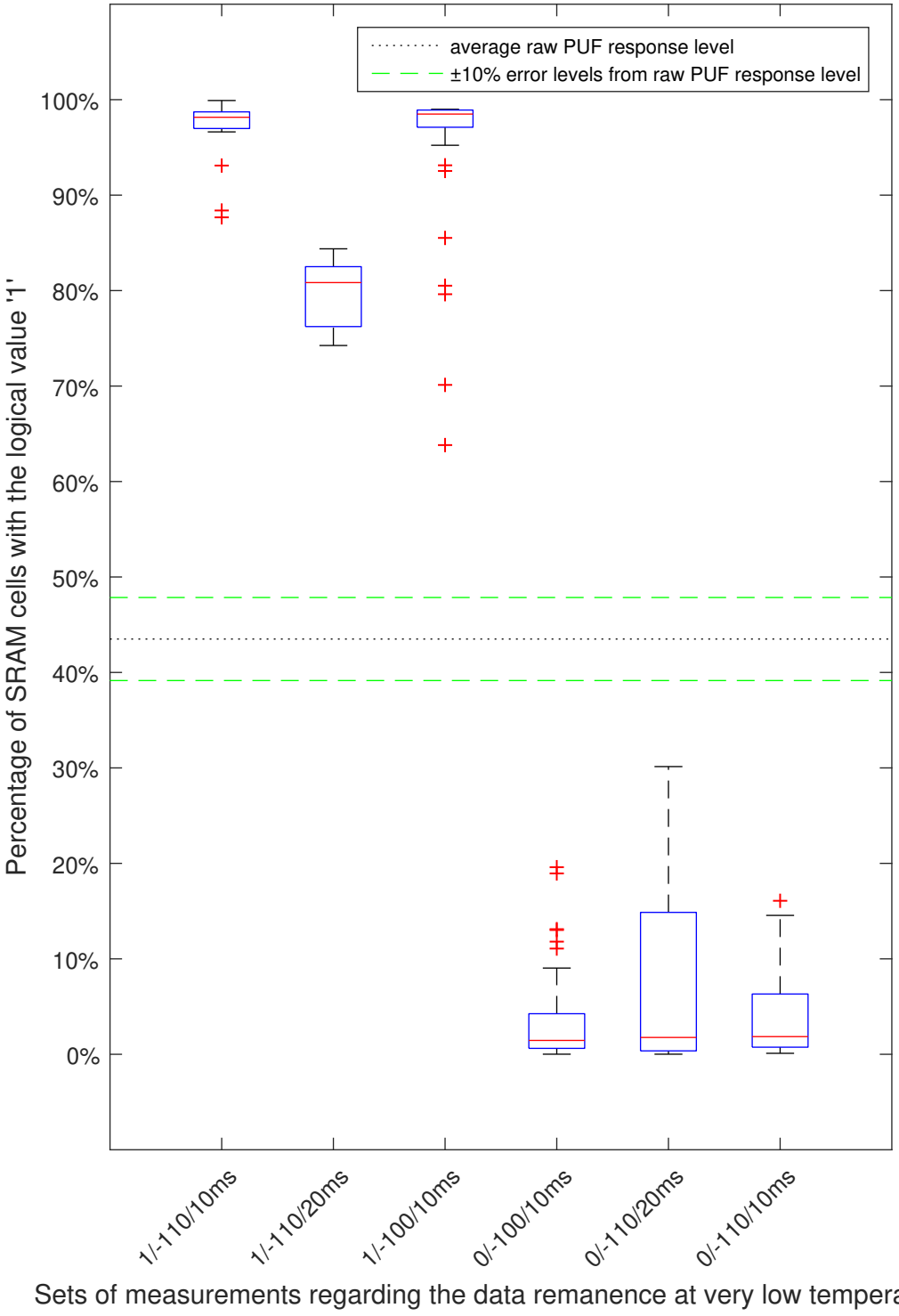


Figure 4.24: A comparison of data remanence measurements at very low temperatures. In the identifier of each set, the first number indicates whether the SRAM was written with logical '1' or logical '0'. The second number denotes the temperature at which the measurement took place, and the third the relevant power-off time in milliseconds.

As our results show, the intrinsic SRAMs under examination exhibit relatively high data remanence at very low temperatures. In this section, we prove that such temperatures can easily be achieved in a relatively stable manner even when thermal isolation is lacking. We note, however, that, of course, the results achieved under the condition of thermal isolation exhibit higher data remanence and better

stability. Nevertheless, it is obvious from the Figures presented already in the previous segment, that even our initial setup of a closed-top EPF – styrofoam – box was not providing total thermal isolation, as the measurements reveal relatively high instability in the level of data remanence achieved for each particular temperature, which is evident in the high variance of the data remanence levels measured at each particular temperature. Nevertheless, taking into account that we examine a COTS IoT device, which can find use in remote applications, quite often in the open, and outside the isolated space of a laboratory, we believe that it is worth examining the behaviour of the device when thermal isolation is particularly lacking, by placing the device in a box that is completely open on one of its sides, i.e., on the top.

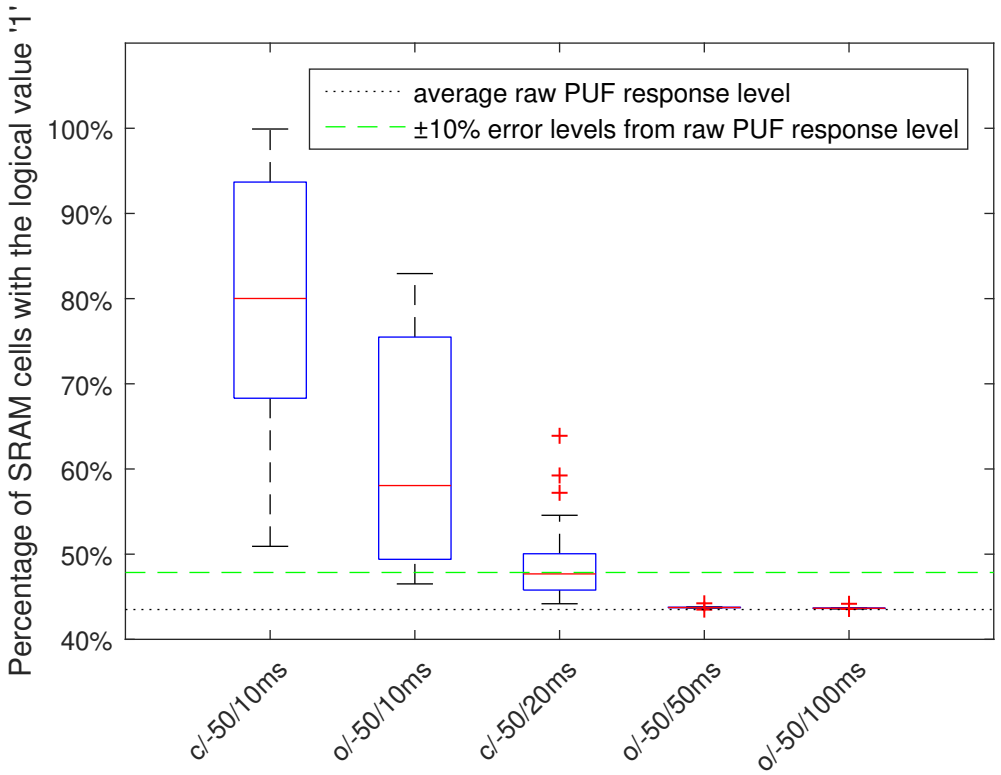


Figure 4.25: A comparison of measurements at −50°C for the data remanence of the logical value '1'. In the identifier of each set, a 'c' signifies a closed EPF – styrofoam – box measurement, while 'o' an open box measurement. The first number denotes the temperature at which the measurements took place, and the second indicates the relevant power-off time in milliseconds.

Our experimental results, indeed, demonstrate a clear degradation in the level of data remanence when the relevant EPF – styrofoam – box is open-top, as shown in Figures 4.25 and 4.26. Figure 4.25 presents our results for the crucial temperature of −50°C, which is easily achievable and at which a closed-top thermal isolation box can still provide a high enough level of data remanence for 10ms power-off time. Nevertheless, as it is evident in Figure 4.25, the data remanence level for 10ms power-off time and an open EPF – styrofoam – box is significantly lower than that for the same power-off time and a closed box. Additionally, while the SRAM cells have almost completely returned to their normal start-up values, indicating very little to no data remanence, after a power-off time of 20ms and a closed EPF – styrofoam – box, they have completely done so, indicating no data remanence, for an open box and a

longer power-off time of 50ms or 100ms. We, therefore, note that, at the critical temperature of −50°C, thermal isolation plays a decisive role. We also note a very clear effect of the power-off time on the level of data remanence observed in the SRAM, which also appears to play a key role in whether a high level of data remanence can be achieved or not.

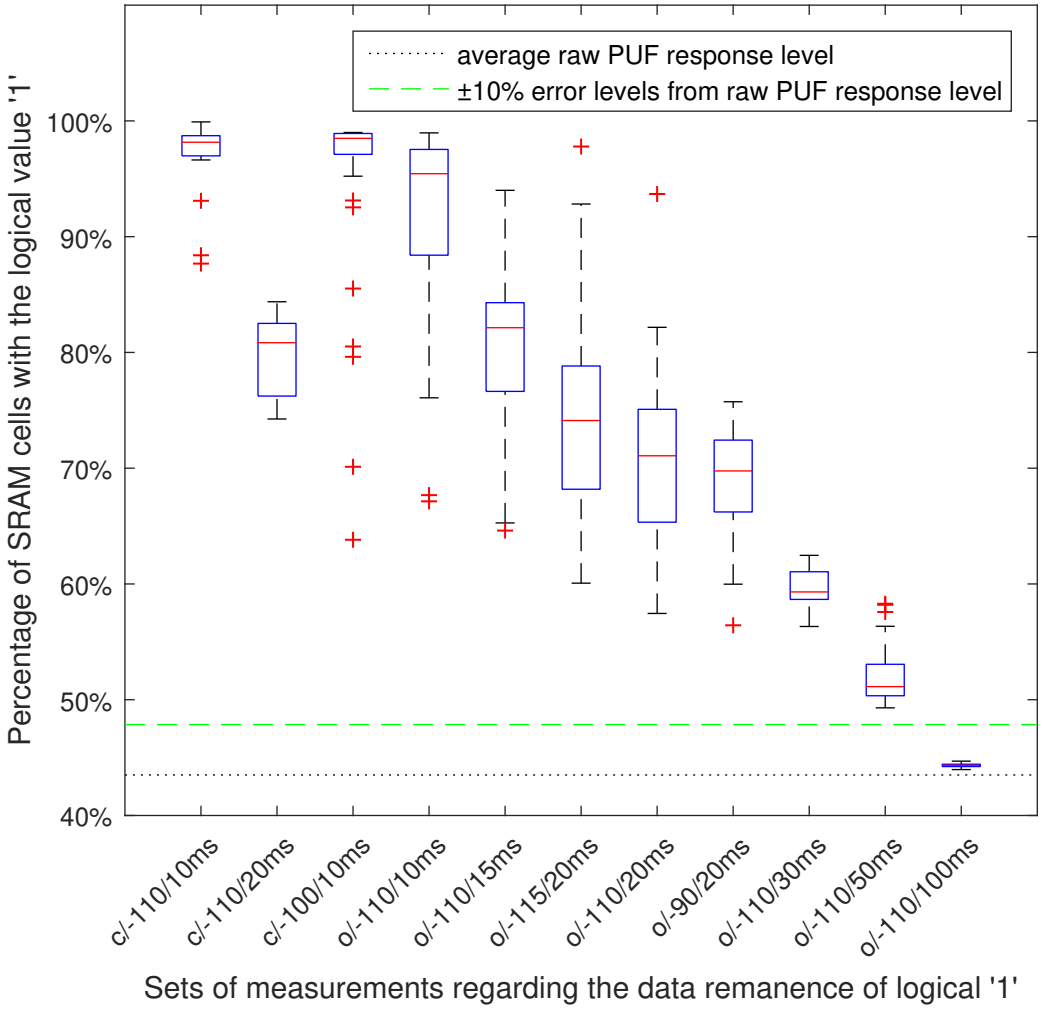


Figure 4.26: A comparison of measurements at the lowest temperatures for the data remanence of the logical value '1'. In the identifier of each set, a 'c' signifies a closed EPF – styrofoam – box measurement, while 'o' an open box measurement. The first number denotes the temperature at which the measurements took place, and the second indicates the relevant power-off time in milliseconds.

Moreover, Figure 4.26 provides more insight into the data remanence level at the lowest temperatures at which the boards continue to be fully responsive. It is again easy to observe that the data remanence level at −110°C for an open EPF – styrofoam – box is much lower than the one for a closed box, both for 10ms and for 20ms power-off times. Additionally, it can be seen that the data remanence for a closed EPF – styrofoam – box at −100°C is also higher than that for an open box at −110°C. Furthermore, this figure also shows how data remanence decreases as the power-off time for an open EPF – styrofoam – box at −110°C increases. Nevertheless, the same figure also indicates that the data remanence level for an open EPF – styrofoam – box for a 20ms power-off time remains significantly high at around 70%, even when the temperature increases to −90°C. Finally, we also note that even when the

temperature is around −110°C, for relatively high power-off times, such as 30ms and 50ms, the data remanence level drops at ≈60% and ≈50%, respectively, which indicates little, but noticeable, data remanence, while for a power-off time of 100ms, the SRAM cells have almost completely returned to their normal start-up values, which indicates a complete lack of data remanence.

We can, therefore, conclude that, although it would be more difficult to achieve a high level of data remanence in the SRAMs under examination in the open, this does not prove to be impossible, or even improbable. Nevertheless, we also note that adequate care would need to be taken in order to control the power-off time for the board, as our results clearly prove its crucial role in the achieved level of data remanence in the SRAMs that we have examined. We will, therefore, proceed to examine relevant low-temperature data remanence attacks against the examined PUFs, as well as briefly discuss some potential countermeasures against such attacks.

Low-Temperature Data Remanence Attacks Against SRAM PUFs

In this segment, we will examine how low-temperature data remanence can be used to attack SRAM PUFs. First, we present such a relevant attack, based on the experimental results for measurements conducted under thermal isolation, and, then, based on the results of the comparison of the experiments conducted with and without thermal isolation, we will discuss and examine whether this attack is practical and feasible in non-isolated environments, outside the laboratory, where most COTS IoT devices are usually used. Finally, we have observed that the described attack seems to be highly effective also against extrinsic (dedicated use) SRAM PUFs, which do not serve as inherent memory components of the system, but only as PUFs.

The relevant attack exploits the observed data remanence effects that have been discussed in the previous segments. In particular, the attacker wants to preserve in the SRAM, through low-temperature data remanence, an own bit-string that has been written on the SRAM before reboot. In this way, the SRAM can be forced to have particular contents that are known to the attacker, when it is used as an SRAM PUF, at start-up, and, in this way, the attacker can know the relevant SRAM PUF response. The experimental results presented in the previous segments verify this attack's potential for success.

Regarding the attack, it is assumed that the relevant SRAM PUF, whose response is the concatenation of the start-up values of the cells of the SRAM module (and therefore a reboot is required for its operation), operates according to the following steps, which are essentially the same as the ones described in Section 3.1.1:

N1. When the device reboots, the bootloader queries the SRAM for its start-up values, which form the raw SRAM PUF response.

N2. Then, helper data for error correction are acquired either from the device itself or from another device.

N3. The raw PUF response and the helper data are combined, using a fuzzy extraction scheme, to generate a cryptographic token, e.g., a secret key.

N4. Finally, the bootloader overwrites the SRAM cells with a specific logical pattern, which erases the raw SRAM PUF response from the SRAM module.

Again, the relevant helper data and the relevant fuzzy extraction scheme are considered as publicly available information, with the only secrets being the response of the SRAM PUF and the resulting cryptographic token.

The relevant low-temperature data remanence attack is based on performing the following two additional steps *before* the normal SRAM PUF operation commences:

A1. The attacker chooses a cryptographic token, e.g., a key, that will be forged and a random bit-string that will replace the raw PUF response. Then, based on this key and the chosen bit-string, the corresponding helper data are produced. These helper data will then either replace the helper data stored on the device or be sent to it, instead of the intercepted legitimate helper data, in connection to step N2. above.

A2. The attacker, then, writes the chosen bit-string on the SRAM, and ensures that the ambient temperature is significantly low, so that an adequately high level of data remanence is achieved, until a reboot occurs.

In this way, the attacker can make sure that the helper data corresponding to the chosen bit-string will be used, resulting in the chosen cryptographic token, e.g., a key, being constructed and used by the device. Using this method, the attacker is able to construct a forged secret key by manipulating the PUF response, even though the PUF response is overwritten by the system immediately after being extracted and used. Additionally, such an attack requires relatively low expertise and resources. Nevertheless, it is clear that such an attacker requires logical access to the SRAM module, as well as the ability to control the relevant ambient temperature. However, if the relevant legitimate helper data are indeed public, this attack can be successful, even if the attacker does not substitute the legitimate helper data with the relevant forged ones.

For the purpose of testing out the attack scenario experimentally, non-privileged attacker code that could write to the SRAM was implemented. Additional non-privileged code that could read the SRAM values and then copy them to a different memory segment, or transmit them to a computer through the Universal Asynchronous Receiver-Transmitter (UART) interface, was also implemented. A random key, which is a pre-selected bit-string of a particular size, as well as the raw PUF response that the SRAM PUF would be forced to produce (either all '0' or all '1' in our experiments), were chosen. Finally, the chosen key and the forged PUF response were combined through a fuzzy extractor scheme [16, 76, 136, 138, 139], in order to produce the corresponding helper data.

In the experiments presented in this work, first, the forged PUF response is written to the SRAM, then, the ambient temperature is lowered to and kept at a certain value, until the device reboots, and, finally, after start-up, the system is manipulated into using the helper data we have constructed in order to generate a key, as described in step A1. above. Subsequently, it is checked whether the produced key matches the one that had been chosen. It is also worth noting that the Texas Instruments Stellaris LM4F120 LaunchPad evaluation boards (EK-LM4F120XL) can reboot instantly and, therefore, their power-off time during a reboot is almost non-existent. Nevertheless, for the purposes of this work, we utilise the results presented in the previous segment and assume the (non-optimal) power-off times of 10ms and 20ms. In either case, however, the attacker needs to ensure that the power-off time during the reboot is as little as possible. An overview of the level of data remanence in the on-die SRAM module

of the LX4F120H5QRMCU of a Texas Instruments Stellaris LM4F120 LaunchPad evaluation board (EK-LM4F120XL) in the (ambient) temperature range between −110°C and 25°C, after 10ms of power-off time, is provided in Figure 4.27.

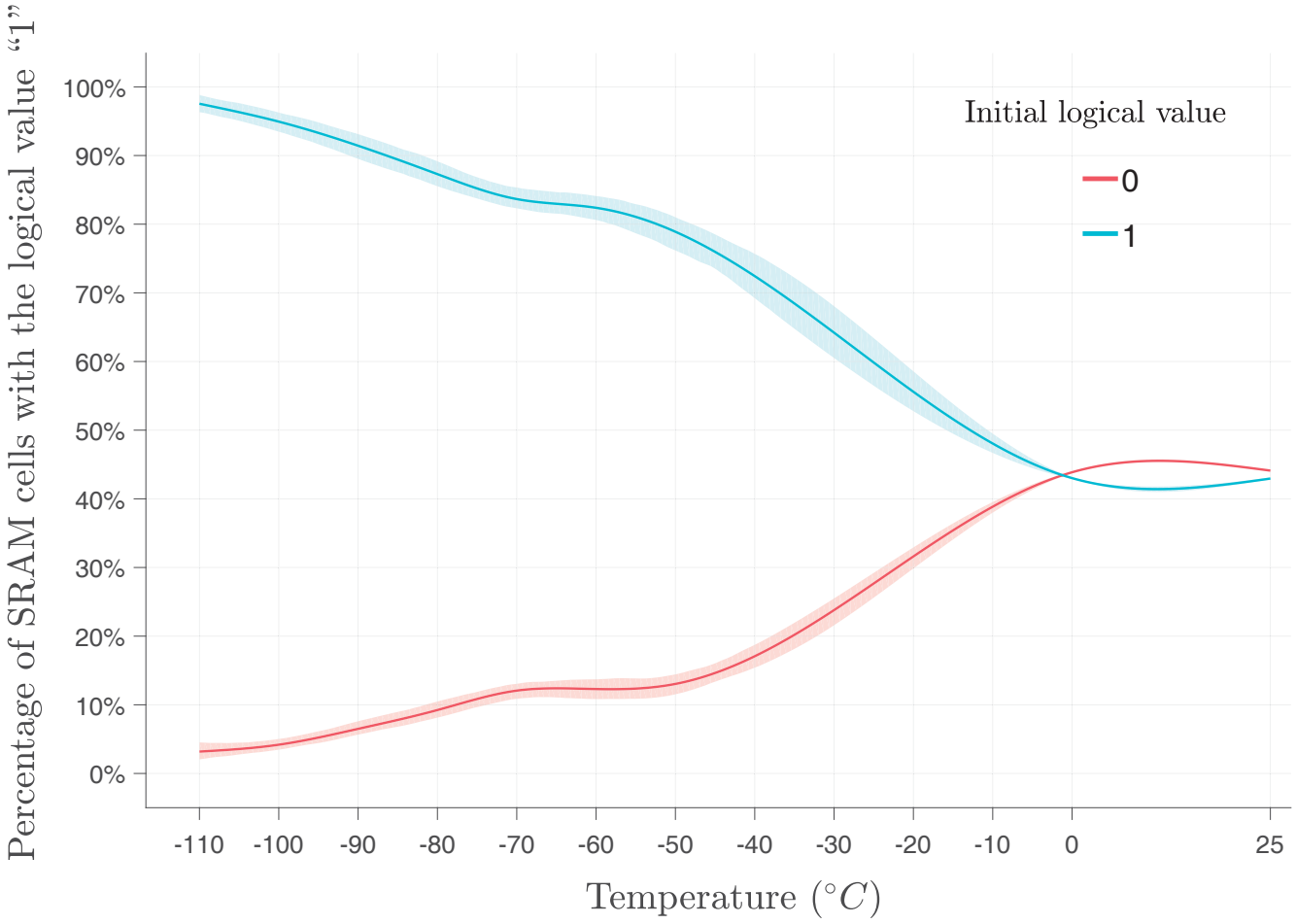


Figure 4.27: An overview of the level of data remanence in the on-die SRAM module of the LX4F120H5QRMCU of a Texas Instruments Stellaris board, in the (ambient) temperature range between −110°C and 25°C, as a percentage of SRAM cells with the logical value '1' after 10ms of power-off time, when all the cells initially had either the logical value '1' (blue) or the logical value '0' (red).

Additionally, the error correction of the fuzzy extractor scheme used is based on a simple repetition code and the binary Golay $(23, 12, 7)$ ECC, which is also referred to as the perfect binary Golay code, a proven error correction scheme for SRAM PUFs [133, 136], which can correct errors accounting for up to ≈10% of a raw PUF response, if they are uniformly distributed.

Unfortunately, errors in the SRAM PUF responses that were collected in the experiments conducted, were not uniformly distributed, and, thus, although the error correction code can in all cases correct errors accounting for up to at least 5% of the SRAM response, the exact correction threshold per raw response depends on both the number of errors contained in it and their actual distribution. Nevertheless, our scheme has been successfully tested using both 128-bit and 512-bit keys. It is also important to note here that the binary Golay $(23, 12, 7)$ ECC can detect up to 6 errors and correct up to 3 errors, in a block of 23 bits containing 12 bits of data [133].

The resulting rates of success for the attack described in this work, for different values of ambient temperature and for power-off times of 10ms and 20ms are shown in Table 4.3. The experimental setup used provides relative thermal isolation.

Table 4.3: Rates of Successfully Reconstructing the Key Forged by the Attacker Using the Raw SRAM PUF Responses Collected After a Power-Off Time of 10ms and After a Power-Off Time of 20ms.

Pattern written	**10ms power-off time**									
	−110°C	−100°C	−90°C	−80°C	−70°C	−60°C	−50°C	−40°C	0°C	25°C
all '1'	90%	≈88%	60%	≈32%	<20%	<20%	≈11%	≈8%	≡0%	≡0%
all '0'	≈93%	≈92%	≈64%	≈63%	≈51%	<50%	≈38%	≈28%	≡0%	≡0%

Pattern written	**20ms power-off time**			
	−110°C	−90°C	−70°C	−50°C
all '1'	<10%	<10%	≡0%	≡0%
all '0'	<10%	<10%	≡0%	≡0%

Based on the results presented on Table 4.3, for a reset time of 10ms, it is noted that the attack is successful in 90% of all cases for temperatures below −100°C and in more than 60% of all cases for temperatures below −90°C. Additionally, a discrepancy in the success rates for the two logical values is observed, which can be attributed to the SRAM PUF response being slightly biased towards the logical value of '0'. It is, therefore, also noted that an attacker is more successful when storing the logical value towards which the SRAM PUF is biased. Furthermore, it also observed, based again on the results shown in Table 4.3, that when a pattern of all '0' has been stored on the SRAM, successful key reconstruction rates are close to 40% even for temperatures close to −50°C. Finally, it is also noted that for a reset time of 20ms, the successful key reconstruction rate falls below 10% even for a temperature around −110°C and is definitively equivalent to 0% for temperatures around −70°C and above. Therefore, the attacker really needs to try to keep the ambient temperature as low as possible and the power-off time as little as possible.

Discussion Regarding the Feasibility and Practicality of the Described Low-Temperature Data Remanence Attack

In these paragraphs, we utilise the results noted in the previous segments in order to demonstrate that the proposed attack can be considered as practical, as it is feasible even in environments that are not thermally isolated. In particular, we note that the described attack is based only on the temperature of the SRAM and the time that the SRAM remains powered off. Taking into account that the examined device is not stable at temperatures around −120°C and below, we examined its data remanence at temperatures down to −110°C and above. Our results, which have already been discussed in the previous segments, demonstrate that, even without adequate thermal insulation, a high level of data remanence can be achieved at very low temperatures. In particular, the resulting rates of success for the proposed attack, when either an open EPF – styrofoam – box or a closed such box was used, for different power-off times, are presented in Table 4.4. Again, this table refers to rates of successful reconstruction achieved using a fuzzy extractor scheme that utilises the binary Golay $(23, 12, 7)$ ECC, as already described in the previous segment.

Table 4.4: Rates of Successfully Reconstructing the Key Forged by the Attacker Using the Collected Raw SRAM PUF Responses, When the Device Was Contained in an Open-Top EPF – Styrofoam – Box and the Pattern Written to the SRAM Was All '1', and Relevant Rates of Successfully Reconstructing the Forged Key From the Collected Raw SRAM PUF Responses, When the Device Was Contained in a Closed-Top EPF – Styrofoam – Box and the Pattern Written to the SRAM Was All '1'.

open EPF – styrofoam – box					
−110°C & 10ms reset	−110°C & 15ms reset	−115°C & 20ms reset	−110°C & 20ms reset	−90°C & 20ms reset	−110°C & 30ms reset
≈59%	≈2.1%	≈1.5%	≈0.8%	≡0%	≡0%

closed EPF – styrofoam – box		
−110°C & 10ms reset	−110°C & 20ms reset	−100°C & 10ms reset
90%	<10%	≈88%

open box	**closed box**	
−50°C & 10ms reset	−50°C & 10ms reset	−50°C & 20ms reset
≡0%	≈11%	≡0%

As Table 4.4 clearly indicates, the proposed attack can be highly successful even when an open EPF – styrofoam – box is used, when the ambient temperature is extremely low and the power-off time extremely small. In particular, we note that, for an open box, the attack is successful in ≈59% of the experiments conducted at −110°C for a power-off time of 10ms. However, the rates of success are only ≈2.1% at −110°C for a power-off time of 15ms, ≈0.8% at −110°C for a power-off time of 20ms, and ≡0% at −110°C for a power-off time of 30ms, when using an open box. Additionally, when using an open box, the rate of success at −115°C for a power-off time of 20ms is ≈1.5%, but ≡0% at −90°C for a power-off time of 20ms. Moreover, we need to stress that for a controlled environment with good thermal isolation, the attack can even be successful – in more than one tenth of the experiments conducted – at temperatures as high as −50°C for a power-off time of 10ms.

We can, therefore, conclude that the attack will be successful when an open box is used, in order to simulate an environment without thermal isolation, only when a very low temperature can be achieved as well as a very small power-off time. Nevertheless, we note that our experiments demonstrate that this is possible at a very low cost, and does not require high expertise or resources, as, e.g., a cooling spray could be used. Thus, we believe that the attack is highly practical, being feasible in adverse uncontrolled environments, outside the laboratory, such as the ones where COTS IoT devices are expected to be found.

We also note that, by using a different ECC in the fuzzy extractor scheme employed, such as one of the Bose–Chaudhuri–Hocquenghem (BCH) codes, which allow for more flexibility in error correction than the binary Golay ECC, the rate of success for the proposed attack could be higher. In particular, an attacker would have achieved even higher rates of success for most of the low temperatures being examined, if a BCH code was incorporated into the fuzzy extractor scheme instead of a binary Golay code, as, in some cases, errors seem to be clustered in memory regions that may have a higher temperature than others during the measurement, and BCH codes allow for the correction of both highly concentrated

and widely scattered errors. We, therefore, note that the rate of success of the attack depends also on the fuzzy extractor scheme being used and the ECC that this includes.

Finally, we need to note that our preliminary results from a setup that includes an extrinsic (dedicated) SRAM PUF seem to indicate that such a PUF also exhibits a similar behaviour. In particular, we have tested an Alliance Memory AS7C34098A SRAM module as an extrinsic SRAM PUF connected to an ST Microelectronics STM32F429I Discovery (STM32F429I-DISC1), using the same experimental setup as for the measurements already presented. Our preliminary results regarding the data remanence of this extrinsic SRAM PUF at very low temperatures have clearly indicated a behaviour that is highly similar to that of the intrinsic SRAM PUF implemented using the on-die SRAMs of the LX4F120H5QR MCUs of the Texas Instruments Stellaris LM4F120 LaunchPad evaluation board (EK-LM4F120XL). We, therefore, strongly believe that the low-temperature data remanence attack proposed in this work is highly applicable to different SRAM PUF implementations incorporated into a variety of devices, which may be used in different environments and configurations.

Potential Countermeasures Against the Proposed Data Remanence Attacks

A variety of countermeasures against data remanence have been proposed in the relevant literature. An overview of the potential ways of mitigating the low-temperature data remanence attacks against SRAM PUFs that are described in this work can be found on Table 4.5.

Table 4.5: Evaluation of Potential Countermeasures Against the Described Data Remanence Attacks

Category of Countermeasures	Potential for Successful Mitigation
Altering the SRAM design	High, but impractical, due to manufacturing costs
Using a different type of memory	High, but impractical and infeasible, due to manufacturing costs and the replacement of the SRAM module with another type of memory; Precludes the existence of an SRAM PUF
Addition of wire meshes and physical defences	Low to medium, depending on how difficult it is to remove or bypass these defences, how much thermal isolation they provide, and how cost-efficient they are
Obfuscation or encryption of the SRAM	Low to medium, depending on how difficult it is to circumvent these measures; May hinder SRAM PUF operation
Restriction of access to the boot-loader and other privileged code components, or to the SRAM module in general	Medium to high, depending on how difficult it is to overcome this restriction; May hinder SRAM PUF operation (by non-privileged code)
Overwriting or erasure of the SRAM	Very low, as the attack has been proven to work even when the SRAM is overwritten (or erased)
Physical destruction or significant physical alteration (e.g., degaussing) of the SRAM	Very high, but totally impractical; Renders SRAM completely unusable, both in general and as a PUF, or at least changes significantly (and, most likely, permanently) its PUF response and characteristics
Setting a specific minimum power-off time	Medium, as it is not practical and could be circumvented, especially at extremely low temperatures; The device would need to never lose power, and keep constant track of time and the on-off state of the SRAM
Using temperature sensors to detect abrupt temperature changes	Low, as it is not practical, detection can be avoided, and may also require additional manufacturing costs; The device would need to never lose power and keep constant track of the temperature

More specifically, solutions which alter the memory design architecture by adding extra circuits and components [206, 207, 208], cannot be considered as efficient, practical, or even feasible, as they also cause extra manufacturing costs which can usually be high enough to not allow such solutions to enter mass production. Furthermore, solutions which are based on a different kind of memory, other than an SRAM, such as the ones proposed by Zhang et al. [209], not only suffer from high manufacturing costs, but also obviously exclude the existence of an SRAM PUF, which is the target of the data remanence attacks that we examine.

Some more conventional countermeasures against the described attacks include the addition of wire meshes and other physical defences, the obfuscation or encryption of the memory, the restriction of access to privileged code components, such as the boot-loader, and the overwriting or erasure of the memory [206, 210, 211, 212, 213, 214]. However, these countermeasures can only make data remanence attacks harder to perform, but not completely prevent them, as there are ways to circumvent them. For example, it has been proven that the success of the presented attack is not dependent on whether the SRAM is overwritten after the PUF operation [3, 4]. Moreover, such countermeasures may also require additional resources and incur further costs.

Huffmire et al. [215] have even proposed physically destructing the device as a means of preventing data remanence attacks. Destructing the device, or significantly altering it, cannot serve as an efficient way of protection, especially in the case of SRAM PUFs, where the SRAM has to remain fully functional in order to serve as a PUF, while, e.g., degaussing it would probably significantly alter its PUF characteristics and, thus, its response. In either case, such actions would also result in a successful DoS attack, as the SRAM, as well as the PUF implemented on it, would be rendered unusable. Therefore, such countermeasures can only serve as a last resort, in order to prevent other parties from gaining access to the device and its SRAM PUF at all costs.

Defining a specific minimum power-off time for the device [204, 216] can also be a potential countermeasure. However, in order to implement such a countermeasure, the device should never lose power completely, as the system would need to constantly determine the time and the SRAM's on-off state, in some way. In this case, an attacker could bypass such an extra system by powering it off or disrupting its operation and/or registers, or even by cooling down the device to such a low temperature that the duration of the relevant data remanence effects exceeds the minimum power-off time set.

Another interesting countermeasure against data remanence attacks is ensuring that the SRAM module used as a PUF cannot be accessed by non-privileged software. For example, if, instead of an intrinsic SRAM PUF, an extrinsic (dedicated use) SRAM module is used as a PUF, an attacker may not be able to write to its cells, or even read them, as non-privileged user code could potentially not have access to this SRAM. Nevertheless, even in this case, efficient attacks based on low-temperature data remanence may still exist, such as physically probing this SRAM module in order to write to its cells and then freezing it until reboot. The implementation and examination of such attacks, however, are left to future research.

Additionally, the use of temperature sensors to detect abrupt changes in the temperature and take adequate action, in order to protect the device [212], could potentially prevent the success of the described attacks, but only if the device remains constantly operational. In particular, an attacker could avoid detection by powering off the system or by disrupting such sensors, especially if they were not mounted on the SRAM module itself. In either case, however, the effectiveness of this countermeasure

in the case of low-temperature data remanence attacks against SRAM PUFs remains the potential subject of further future research.

In general, the effectiveness of using temperature sensors as a countermeasure is highly dependent on what happens when a deep surge in the temperature is detected. In case only the SRAM is overwritten, the presented attack would still succeed. In case the system is powered off, the attack may still be successful, if the attacker can power it back on quickly enough. Finally, in case the SRAM is somehow disabled or otherwise affected, temporarily or permanently, that would also impair the operation of the SRAM PUF, thus, effectively achieving at least a DoS attack. We must also note that not all devices have an internal temperature sensor and its potential addition will inevitably also increase the manufacturing costs of the device.

4.2.2 DRAM-Based PUFs Under Ambient Temperature Variations

A number of works have demonstrated that some DRAM-based PUFs can be significantly affected by temperature variations. In particular, it has been shown that DRAM startup-based PUFs, DRAM decay-based PUFs, and DRAM Row Hammer PUFs are highly affected by ambient temperature variations [48, 79, 82, 83, 85, 86, 87, 143, 147, 153, 158, 192], while the effects of ambient temperature variations on DRAM latency-based PUFs are rather limited [80, 88, 144]. In this work, we will examine in detail the dependency of DRAM decay-based PUFs that have been implemented on Intel Galileo Gen 2 boards, and of DRAM Row Hammer PUFs that have been implemented on PandaBoard ES Rev B3 devices, on ambient temperature variations. As we have already discussed, the quality of the responses of these PUFs is strongly related to the number of bit "flips", i.e., the changes in the logical values, that occur in the cells that are utilised for the production of the relevant PUF responses. Thus, we have measured the number of bit "flips" that are present in the responses of the DRAM retention-based PUFs that have been implemented on the Intel Galileo Gen 2 boards, and in the responses of the DRAM Row Hammer PUFs that have been implemented on the PandaBoard ES Rev B3 devices, in the temperature range between 0°C and 70°C, at intervals of 10°C. Additionally, we have also calculated the intra-device Jaccard index for measurements originating from the same PUF instance but collected at varying ambient temperature levels. In this way, we examine the dependency of these PUF to ambient temperature variations, which can significantly affect the ability of such PUF to provide an *acceptable* level of security, as it could, on its own, lead to a possible DoS or, even worse, potentially be exploited by an attacker. In general, we note that works regarding these two types of DRAM-based PUFs have acknowledged the dependency of these PUFs on ambient temperature variations, but have rather refrained from addressing it in an effective way, and rather focused on the stability of their responses at all relevant ambient temperature values tested, as long as the temperature remains stable [82, 86, 87, 158]. In this work, however, we will not only examine this dependency in detail, but also we will address it, by proposing a protocol, in Section 5.2 that utilises such PUFs in a robust way, even under temperature variations. Nevertheless, we also note that, in order to prevent an attacker from exploiting this dependency, adequate measures must be taken to mitigate the effects of ambient temperature variations on these PUFs, similar to those discussed in the previous text segments regarding SRAM PUFs. In order to study the effects of ambient temperature variations on the relevant DRAM-based PUFs, we have utilised a Heraeus Vötsch HC 4005 climate chamber. Finally, in all cases examined, the voltage provided to the relevant devices is nominal.

DRAM Decay-Based PUFs Under Ambient Temperature Variations

Regarding the ambient temperature, we assume that an adversary can change the ambient temperature of the place where an IoT device lies, without being noticed, and, therefore, also potentially affect the operation of any DRAM-based PUFs found on this device. To this end, we have tested how the operation of a DRAM retention-based PUF found on the Intel Galileo Gen 2 board may be affected by temperature variations in the temperature range between 0°C and 70°C, at intervals of 10°C.

Our results regarding the number of bit "flips" occurring in the responses of the DRAM retention-based PUFs that have been implemented on the Intel Galileo Gen 2 boards, in the temperature range between 0°C and 70°C, are presented in Figure 4.28. As already discussed, these results demonstrate that the number of bit "flips" rises significantly as the ambient temperature increases, while also data

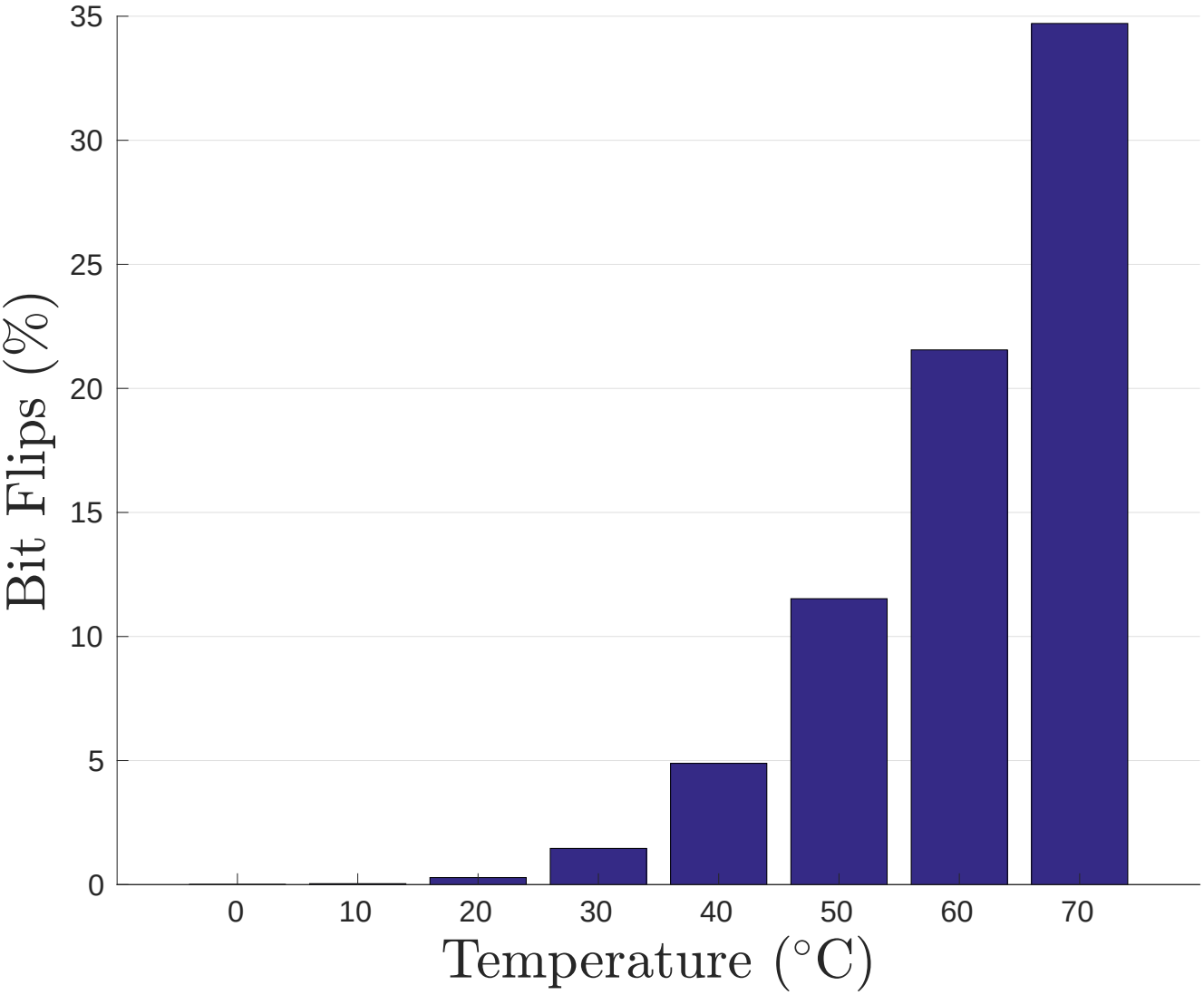


Figure 4.28: Average fractional number of bit "flips" observed in the responses of a DRAM retention-based PUF implemented on the Intel Galileo Gen 2 board, in the temperature range between 0°C and 70°C, at intervals of 10°C. The DRAM region being employed as a PUF is 8 KB and the decay time, i.e., the time period during which the DRAM refresh operation has been suspended, is 300s. Initially, the values of the cells utilised in the relevant PUF responses had all been set to logical '0'.

remanence seems to be evident at low temperatures, e.g., at 0°C, as well as at 10°C, the number of bit "flips" is extremely small. The fractional number of bit "flips" shown in this Figure denotes the total number of cells utilised in the relevant PUF responses that have "flipped", i.e., have changed their logical values, during the operation of the relevant DRAM retention-based PUFs, divided by the total number of cells utilised in the relevant PUF responses. In general, therefore, we note that our results confirm a strong dependency of this type of DRAM-based PUFs on the ambient temperature.

In order to further examine the effects of ambient temperature variations on the relevant DRAM retention-based PUFs, we have also computed the intra-device Jaccard index values for pairs of responses, for which one response was collected at 20°C and the other at a temperature ranging from 0°C to 70°C, at intervals of 10°C, as shown in Figure 4.29. In this way, we can know how different are the PUF responses measured at 20°C from PUF responses from the same PUF instance that have been measured at a temperature ranging from 0°C to 70°C, at intervals of 10°C. As Figure 4.29 reveals, the PUF

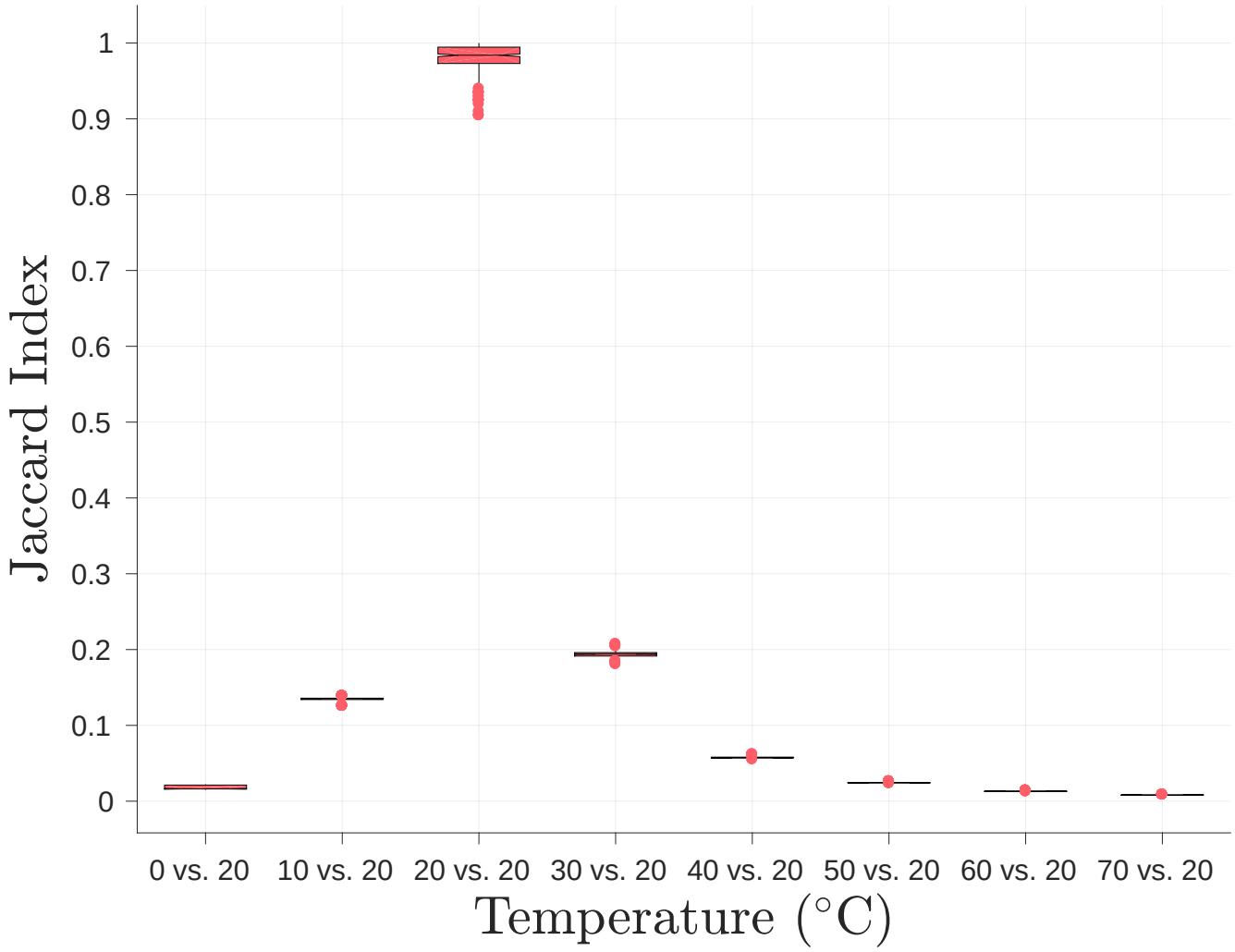


Figure 4.29: Intra-device Jaccard index values for pairs of DRAM retention-based PUF responses, for which one response was collected at 20°C and the other at a temperature ranging from 0°C to 70°C, at intervals of 10°C. The DRAM region being employed as a PUF is 8 KB and the decay time, i.e., the time period during which the DRAM refresh operation has been suspended, is 300s. Initially, the values of the cells utilised in the relevant PUF responses had all been set to logical '0'.

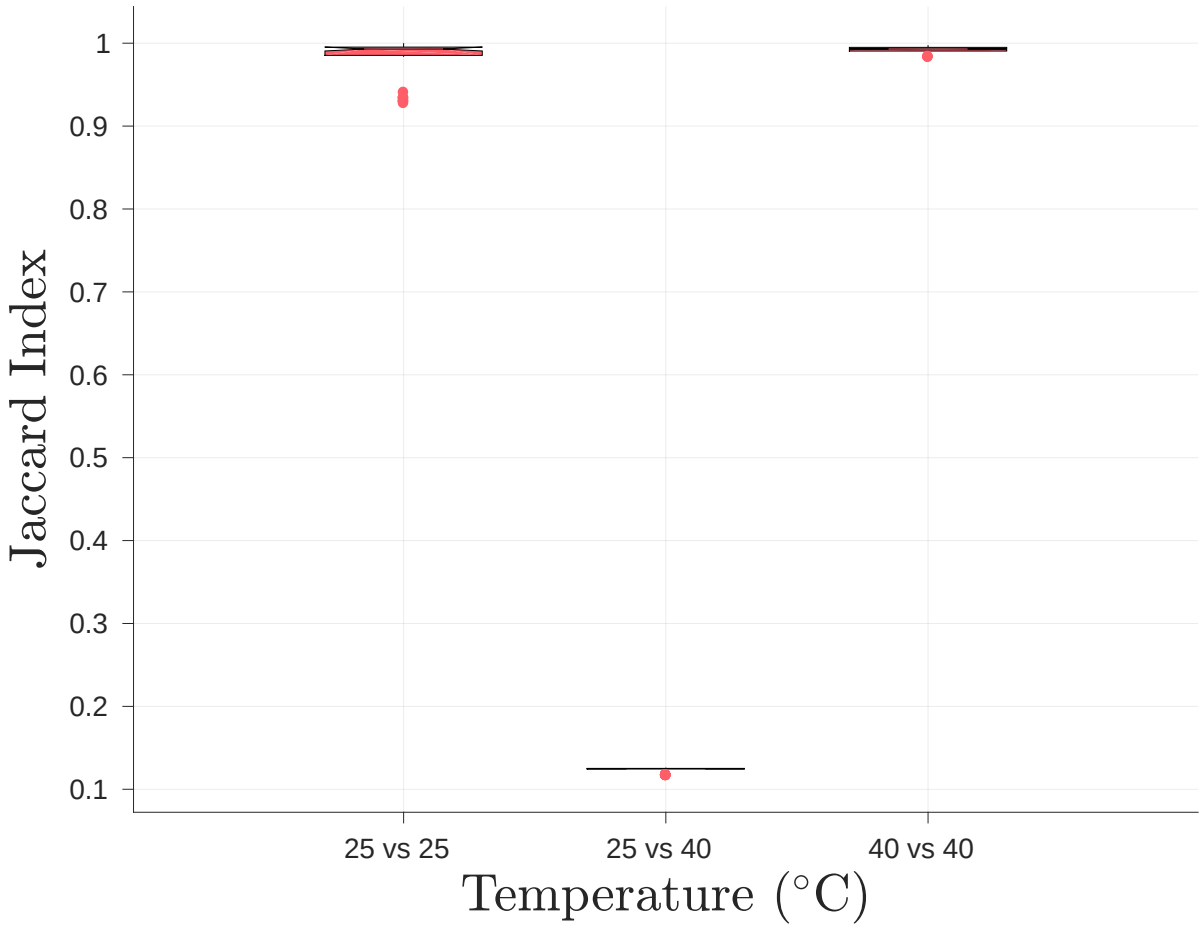


Figure 4.30: Comparison of intra-device Jaccard index values for DRAM retention-based PUF responses collected at 20°C and at 40°C. The DRAM region being employed as a PUF is 8 KB and the decay time, i.e., the time period during which the DRAM refresh operation has been suspended, is 300s. Initially, the values of the cells utilised in the relevant PUF responses had all been set to logical '0'.

responses collected at 20°C are extremely different from PUF responses originating from the same device but collected at a different ambient temperature, while PUF responses collected at the same temperature are extremely similar to each other, both for 20°C and for 40°C, as Figure 4.30 also confirms.

In general, we note that, based on our results for the DRAM retention-based PUF implemented on the Intel Galileo Gen 2 board, a PUF response collected at a particular temperature may lead to a different cryptographic token, e.g., a key, being produced than the one that will be produced based on a PUF response taken from the same PUF instance at a different temperature. Therefore, we note that this DRAM-based PUF suffers from a high dependency on the ambient temperature, which can affect its integrity and accessibility. In particular, an adversary who can control the ambient temperature, can affect in this way both the accessibility and the integrity of such PUFs, as measurements taken at different temperatures are highly different, and measurements taken at very low, temperatures contain a very low number of bit "flips". It is, therefore, apparent that such dependencies on external conditions can significantly decrease the ability of such PUFs to serve as adequate security mechanisms, unless they are properly addressed. Nevertheless, as we will show in Section 5.2, it may be possible to construct robust cryptographic protocols, even from memory-based PUFs that are highly dependent on ambient

temperature, in a rather cost-efficient way, by taking advantage of their internal temperature sensors, in order to take into account the temperature variations that may potentially occur during the collection of their responses.

DRAM Row Hammer PUFs Under Ambient Temperature Variations

Once again, we assume that an adversary can change the ambient temperature of the place where an IoT device lies, without being noticed, and, therefore, also potentially affect the operation of any DRAM Row Hammer PUFs found on this device. For this purpose, we have tested how the operation of a DRAM Row Hammer PUF found on the PandaBoard ES Rev B3 device may be affected by temperature variations in the temperature range between 0°C and 70°C, at intervals of 10°C. Moreover, two row hammer times, i.e., time periods during which the DRAM refresh operation has been suspended and the relevant hammer rows have been being rapidly and repeatedly accessed, were examined, 60s and of 120s. For each level of ambient temperature tested, and for each combination of the PUF parameters noted, 20 PUF responses have been collected.

Our results regarding the number of bit "flips" occurring in the responses of the DRAM Row Hammer PUF that have been implemented on PandaBoard ES Rev B3 devices, in the temperature range between 0°C and 70°C, are presented in Figure 4.31. As already discussed, these results demonstrate that the number of bit "flips" rises significantly as the ambient temperature increases, in the temperature range between 0°C and 70°C. The fractional number of bit "flips" shown in this Figure denotes the total number of cells utilised in the relevant PUF responses that have "flipped", i.e., have changed their logical values, during the operation of the relevant DRAM Row Hammer PUFs, divided by the total number of cells utilised in the relevant PUF responses. The results for both row hammer times examined, 60s and of 120s, indicate a significantly larger fractional number of bit "flips" for the responses of this DRAM-based PUF in comparison to that for the responses of the DRAM retention-based PUF, for the same temperature values. Moreover, we note that our results confirm a strong dependency also of the DRAM Row Hammer PUFs on the ambient temperature. For PUF responses collected using a row hammer time of 60s, a very small fractional number of bit "flips" is exhibited at 0°C and at 10°C, and the fractional number of bit "flips" keeps increasing as the temperature increases. For PUF responses collected using a row hammer time of 120s, the fractional number of bit "flips" exhibited even at 0°C is rather noticable, and again the fractional number of bit "flips" keeps increasing as the temperature increases.

In order to further examine the effects of ambient temperature variations on the relevant DRAM Row Hammer PUFs, we have also computed the intra-device Jaccard index values for pairs of responses, for which one response was collected at 20°C and the other at a temperature ranging from 0°C to 70°C, at intervals of 10°C, as shown in Figure 4.32. In this way, we can know how different are the PUF responses measured at 20°C from PUF responses from the same PUF instance that have been measured at a temperature ranging from 0°C to 70°C, at intervals of 10°C. As Figure 4.32 reveals, the PUF responses collected at 20°C are extremely different from PUF responses originating from the same device but collected at a different ambient temperature, while PUF responses collected at the same temperature, e.g., at 20°C, are extremely similar to each other.

We note that, as it has been discussed in [143], the results are similar for the firmware and the kernel module implementation of this PUF, while the relevant PUF responses start to become less and less

robust as the ambient temperature increases, as the intra-device Jaccard index for responses originating from the same device and collected at the same temperature starts to increase exponentially at ambient temperatures above 60°C. In particular, at 70°C, the intra-device Jaccard index for responses originating from the same device and collected at the same temperature reaches a value of ≈0.25 and ≈0.45 for a row hammer time of 60s and 120s, respectively, for the firmware implementation of this PUF, and a value of ≈0.35 and ≈0.55 for a row hammer time of 60s and 120s, respectively, for the kernel module implementation of this PUF [143]. At the same time, the inter-device Jaccard index remains at values above 0.8, for all the ambient temperature values tested [143].

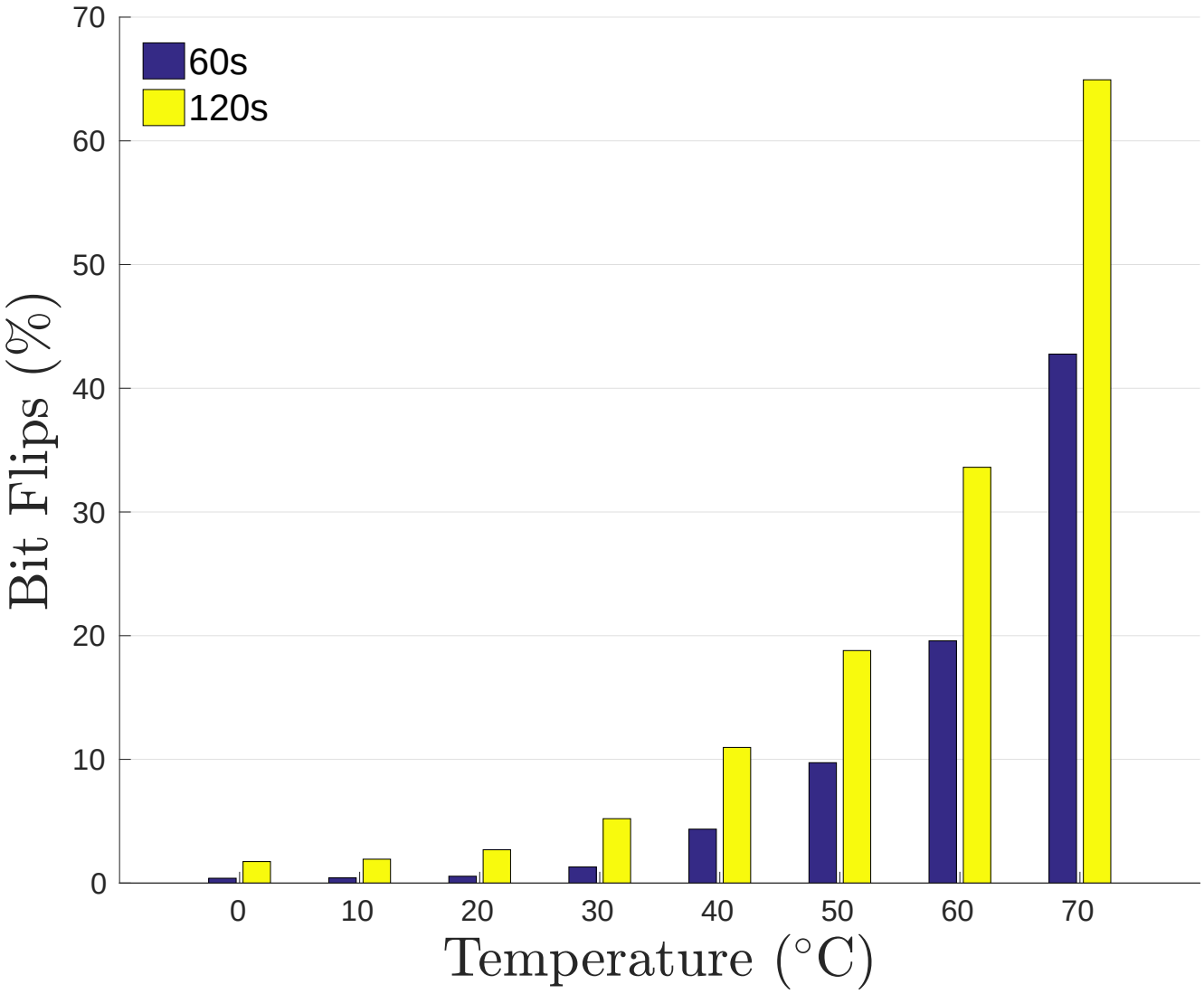


Figure 4.31: Average fractional number of bit "flips" observed in the responses of a DRAM Row Hammer PUF implemented on the PandaBoard ES Rev B3, in the temperature range between 0°C and 70°C, at intervals of 10°C. The DRAM region being employed as a PUF is 128 KB, and two row hammer times, i.e., time periods during which the DRAM refresh operation has been suspended and the relevant hammer rows have been being rapidly and repeatedly accessed, have been examined, 60s and 120s. The rest of the relevant configuration: `RH type=DSRH`, `PUF row IV="0xAA"`, `hammer row IV="0x55"`, `Cache` disabled, according to Table 4.1.

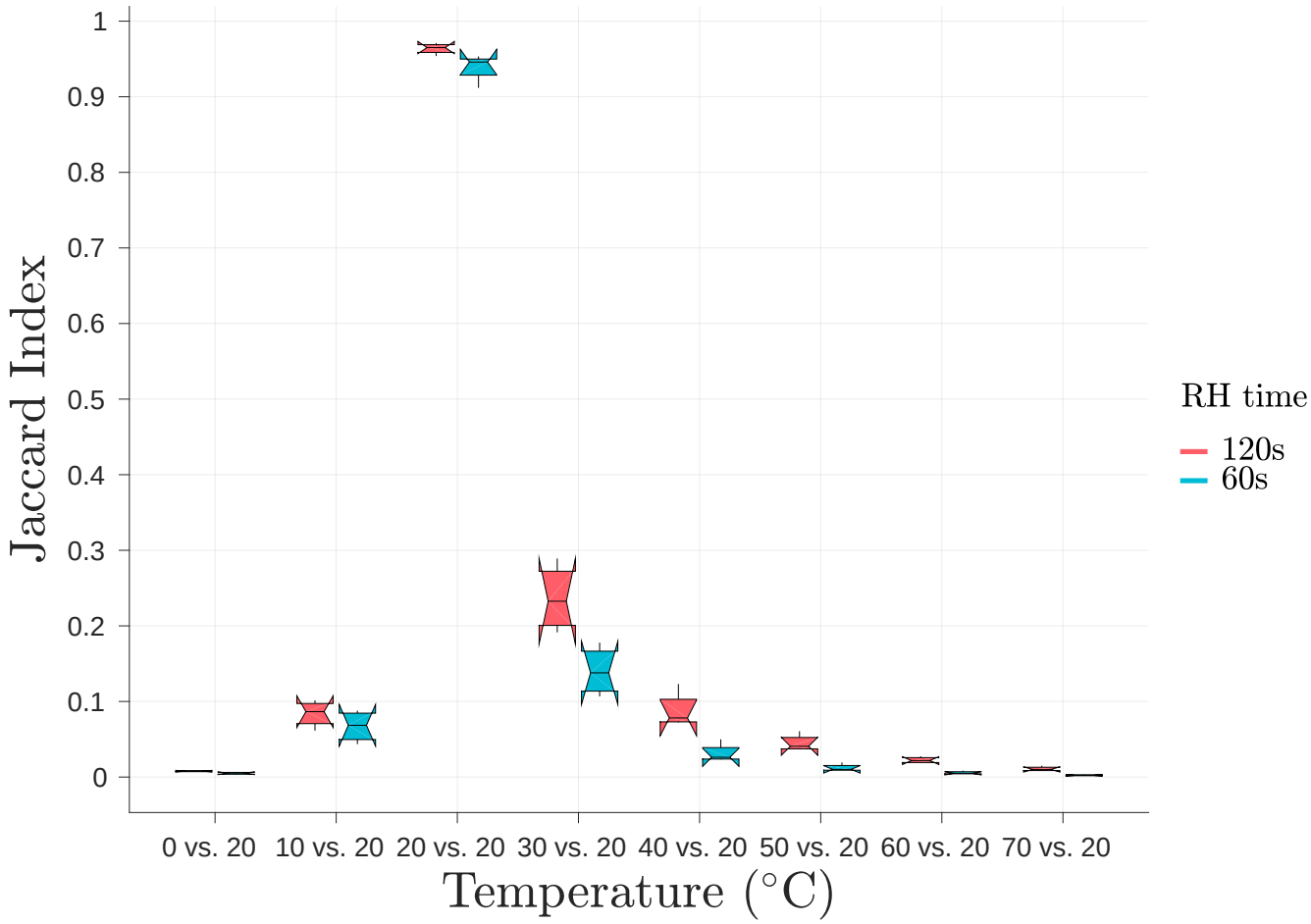


Figure 4.32: Intra-device Jaccard index values for pairs of DRAM Row Hammer PUF responses, for which one response was collected at 20°C and the other at a temperature ranging from 0°C to 70°C, at intervals of 10°C. The DRAM region being employed as a PUF is 128 KB, and two row hammer times, i.e., time periods during which the DRAM refresh operation has been suspended and the relevant hammer rows have been being rapidly and repeatedly accessed, have been examined, 60s and 120s. The rest of the relevant configuration: `RH type=DSRH`, `PUF row IV="0xAA"`, `hammer row IV="0x55"`, Cache disabled, according to Table 4.1.

Therefore, we again note that, based on our results for the DRAM Row Hammer PUF implemented on the PandaBoard ES Rev B3 device, a PUF response collected at a particular temperature may lead to a different cryptographic token, e.g., a key, being produced than the one that will be produced based on a PUF response taken from the same PUF instance at a different temperature. Therefore, we note that also this DRAM-based PUF suffers from a high dependency on the ambient temperature, which can affect its integrity and accessibility. In particular, an adversary who can change the ambient temperature, can affect in this way both the accessibility and the integrity of such PUFs, as measurements taken at different temperatures are highly different, and measurements taken at very high, or very low, temperatures contain a very high, or a very low, respectively, number of bit "flips". It is, therefore, apparent that such dependencies on external conditions can significantly decrease the ability of such PUFs to serve as adequate security mechanisms, unless they are properly addressed. To this end, as we will show in Section 5.2, it may be possible to construct robust cryptographic protocols, even from memory-based PUFs that are highly dependent on ambient temperature, in a rather cost-efficient way, by taking advantage

of their internal temperature sensors, in order to take into account the temperature variations that may potentially occur during the collection of their responses.

A Simple DRAM Decay-Based PUF-Based Protocol Proposed by Xiong et al. [86] and Its Shortcomings Under Ambient Temperature Variations

As we have noted in the previous text segment, the examined DRAM-based PUFs seem to exhibit a significant degree of data remanence at low temperatures, below 0°C. Such data remanence effects on DRAM modules have been noted and extensively investigated in the relevant literature [201, 202, 206, 211, 212, 217, 218]. Additionally, low-temperature data remanence attacks against DRAM modules have also been considered in the relevant literature, with a well-known attack targeting DRAM modules being based on successfully freezing the DRAM module and removing it from the targeted system, in order to gain access to its contents [206, 212, 219]. We note, here, that the potential countermeasures to such attacks are the same as, or extremely similar to, the countermeasures discussed, in a previous text segment, regarding low-temperature data remanence attacks against SRAM modules. Therefore, we refrain from examining in detail low-temperature data remanence attacks against DRAM-based PUFs in this work, noting, however, at the same time, that such attacks are possible, and even rather probable, in the cases of the DRAM retention-based PUF and the DRAM Row Hammer PUF. We also observe, however, that the countermeasures proposed against such attacks will also suffer from the same potential issues, problems, and shortcomings, as the countermeasures proposed in order to address low-temperature data remanence attacks in the case of the SRAM PUF.

In general, as we have already noted, the DRAM retention-based PUF, and the DRAM Row Hammer PUF, are based on the data decay/retention characteristics of the relevant DRAM modules when the DRAM refresh operation has been suspended, which, of course, are describing the data remanence of such devices under the relevant conditions. Therefore, and as data remanence in DRAM modules is highly dependent on the ambient temperature, it is natural that the operation of such PUFs is strongly affected by ambient temperature variations. However, the work of Xiong et al. [86] that introduced implementations of the DRAM retention-based PUF that were accessible at run-time, as well as the work of Schaller et al. that introduced the DRAM Row Hammer PUF [82], did not address the dependency of such PUFs on ambient temperature variations. For example, Xiong et al. [86] proposed an authentication protocol which does not take into account temperature variations at all.

In particular, the relevant authentication protocol consists of two phases; an enrollment phase that needs to be run only once, and a device authentication phase that is run each time the relevant device that hosts the DRAM decay-based PUF needs to be authenticated. The steps of the protocol are as follows:

One-Time Enrollment Phase

1. The enrollment phase is assumed to be conducted in a secure environment by a trusted party. Xiong et al. assume this to be the device manufacturer or a system integrator. However, for reasons of simplicity, we may as well assume that the enrollment phase takes place through a secure and honest server S, which initially has possession of the DRAM decay-based PUF. The PUF is queried by this server and a number of measurements are taken, constituting a set of n measurements, $M = \{m_0, m_1, \ldots, m_n\}$. Each such measurement, i.e., each collected PUF response, contains only

the positions of bit "flips", and corresponds to a particular decay time selected from a relevant matching set of n decay times, $T = \{t_0, t_1, \ldots, t_n\}$, such that $t_0 < t_1 < \cdots < t_n$, in order to allow for the number of bit "flips" contained in each measurement to be larger than the number of bit "flips" contained in the previous one, by at least ε_{bf} new bit "flips", which means that $|m_{i+1}| - |m_i| \geq \varepsilon_{bf}$. Therefore, each PUF response m_i is collected at the corresponding decay time t_i from the same DRAM cells being utilised as a PUF for all n measurements, to which the same bit pattern had been written, before the DRAM refresh operation was suspended, for all the measurements taken. Additionally, the authors note that ε_{bf} can be adjusted to enhance the security and usability of the protocol they propose [86].

2. Then, the server S selects a set of n cryptographic tokens, which the authors propose to be uniformly distributed keys, $K = \{k_0, k_1, \ldots, k_n\}$, and utilises a particular fuzzy extractor scheme in order to produce the set of n helper data values that correspond to each pair of a measurement and its matching key, $HD = \{hd_0, hd_1, \ldots, hd_n\}$, such that $hd_i = Gen(m_i, k_i)$, where $Gen()$ is the relevant fuzzy extractor function to generate the helper data. The authors propose using a tailored fuzzy extractor scheme that would not leak any information useful to an attacker, which they do not specify [86]. The authors propose this step in order to use the sets K and HD for the implementation of a relevant protocol for "secure channel establishment", which essentially is a key agreement protocol. However, for the purposes of this authentication protocol, this step is simply not needed.

3. At this point in time, the server has knowledge of the two relevant sets of measurements, and decay times, respectively, M, and T, as well as of the unneeded relevant sets of keys, and helper data, K, and HD, respectively. Hence, the possession of the PUF is transferred, in a secure manner, to the relevant client C, in order for the PUF to be used to authenticate the client in the future.

Device Authentication Phase

1. Each time the client C wants to be authenticated by the server S, the server will choose the smallest decay time t_x from the set T that has not previously been used in a run of the authentication protocol. Hence, the authors note that, for subsequent authentication trials, decay times are monotonically increasing [86]. However, the authors omit to mention that, obviously, the client C should first send an authentication request to the server, before the server selects t_x at this stage of the protocol, unless the device authentication phase is initiated by the server.

2. Subsequently, the server must provide to the client t_x, as well as the relevant addresses add_{cell} of the DRAM cells that should be utilised as a PUF, and the bit pattern $patt_{cell}$ that should be written to them before the DRAM refresh operation was suspended. This information should, obviously, correspond to the values used in the enrollment phase.

3. Then, the client should use this information in order to calculate, using the DRAM retention-based PUF, the measurement corresponding to this information, i.e., a measurement m'_x that corresponds to t_x, add_{cell}, and $patt_{cell}$. After m'_x has been calculated using the relevant PUF, it should be sent to the server.

4. When the server receives m'_x, it checks if it matches the relevant stored measurement m_x, up to a certain error threshold, ε_{auth}. This can be done by calculating the Jaccard index of the positions of the bit "flips" for the pair of measurements m'_x and m_x, and checking if it is higher than or equal to ε_{auth}, which would lead to successfully authenticating the client, or not, in which case the client is considered as an illegitimate, potentially malicious, entity. The authors propose that ε_{auth} be defined based on the noise observed in measurement m_x, obviously indicating that more than one measurements must be taken for each set of t_x, add_{cell}, and $patt_{cell}$, during the enrollment phase, in order to establish the noise level relevant to m_x. Finally, obviously, for the protocol to work properly, $\varepsilon_{auth} = min(J(m'_x, m_x)) = min(\frac{|m'_x \cap m_x|}{|m'_x \cup m_x|}) > max(J(m_{x+1}, m_x)) = max(\frac{|m_{x+1} \cap m_x|}{|m_{x+1} \cup m_x|}) = \frac{min(|m_{x+1}|) - \varepsilon_{bf}}{max(|m_x|) + \varepsilon_{bf}} = \frac{min(|m_{x+1}|) - \varepsilon_{bf}}{min(|m_{x+1}|)}$. In general, ε_{auth} must be such that m_x and m_{x+1} (as well as m_{x-1}) remain distinguishable despite the inherent noise level denoted by ε_{bf}.

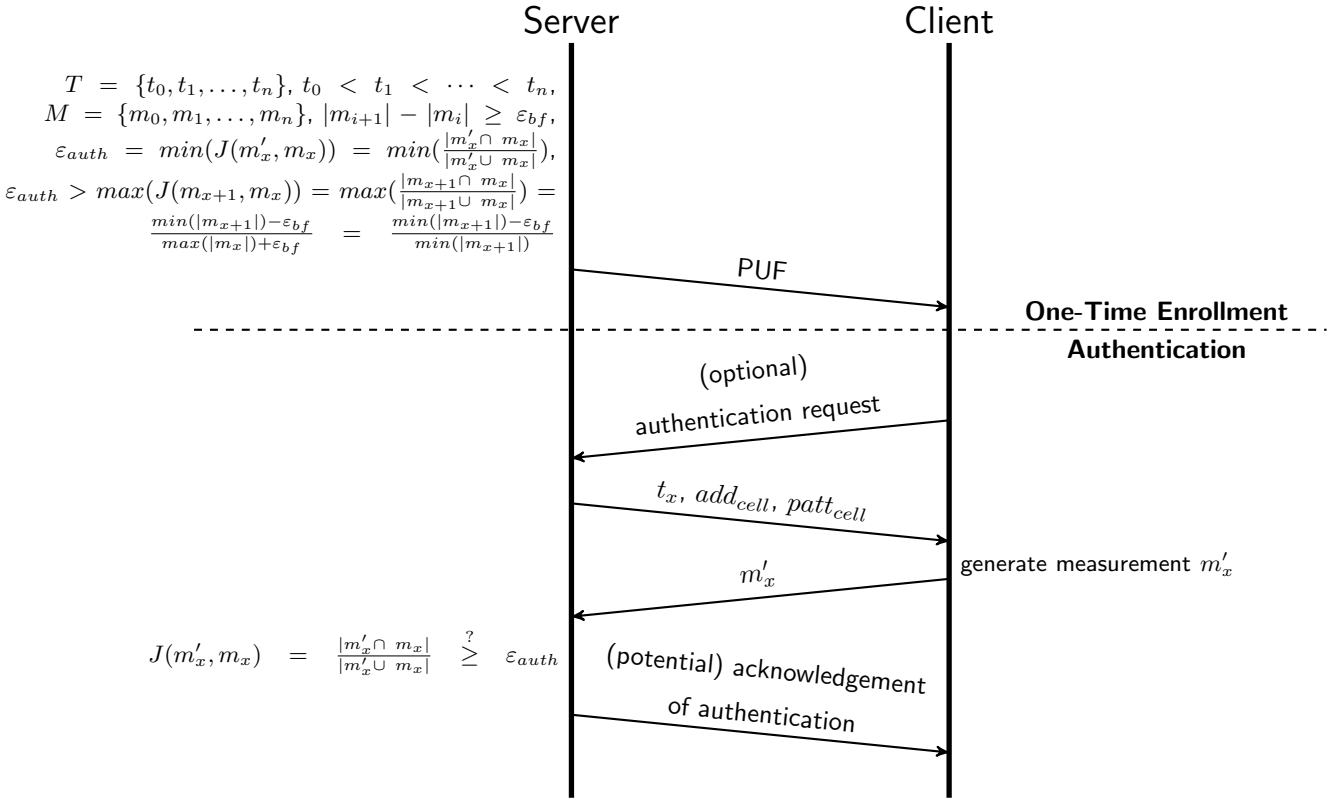


Figure 4.33: An enhanced version of the authentication protocol proposed by Xiong et al. in [86].

An enhanced version of the authentication protocol described by Xiong et al. in [86] is shown in Figure 4.33, based on the observations noted in the description of its steps. Xiong et al. [86] note that the authentication protocol they propose is designed to be lightweight for the client in terms of computational overhead and memory footprint, as it does not require the employment of a fuzzy extractor scheme. Evidently, however, this protocol is *not* robust to temperature variations, as the authors seem to implicitly acknowledge. The authors propose addressing the dependency of a DRAM retention-based PUF on ambient temperature variations by adapting the decay time, i.e., the time period during which the DRAM refresh operation is suspended, as needed. Hence, according to the authors, if the PUF is

evaluated at a different temperature than during enrollment, this can be compensated by adapting the decay time according to the following equation:

$$t_e = \ t \times e^{-0.0662\times(T_2-T_1)} \ , \tag{4.3}$$

which describes the relation between decay time and ambient temperature for a DRAM retention-based PUF implemented on an Intel Galileo Gen 2 board. In Equation (4.3), t denotes a particular decay time required for a specific number (and positions) of DRAM cells to change their value, i.e., to "flip", at a particular ambient temperature T_1, and t_e denotes the decay time required for the same number (and positions) of DRAM cells to "flip" at a particular, potentially different, temperature T_2. For this reason, t_e is referred to as the equivalent time.

Nevertheless, it is easy to observe that if the ambient temperature variations concern more than two levels of temperature values, or the enrollment phase takes place at a very low temperature and the device authentication phase takes place at a much higher temperature, or vice versa, then, the employment of Equation (4.3) on its own will not be able to successfully mitigate the effects of the ambient temperature variations and the proposed authentication protocol will fail to work properly, as, although the relevant measurements may originate from the legitimate PUF, their Jaccard index value will be much lower than ε_{auth}. Additionally, in case the positions of the bit "flips" become the majority, which is the case for very high temperatures, probably the positions of the PUF cells that have not yet "flipped" should be utilised with regards to the Jaccard index values, as, otherwise, an attacker could potentially gain an advantage in successfully authenticating using the proposed protocol, especially with regards to ε_{auth}. Similar observations hold true also for the relevant protocol for "secure channel establishment", which essentially is a key agreement protocol, that Xiong et al. propose in the same work [86].

In general, we note that the dependency of DRAM retention-based PUFs and DRAM Row Hammer PUFs on ambient temperature variations has to be taken into account when a cryptographic protocol based on them is proposed [48], in order to allow for the efficient utilisation of these PUF as security mechanisms in the context of such protocols. To this end, we propose, in Section 5.2, a cryptographic protocol that allows for the utilisation of such PUFs as security mechanisms in a robust manner, even under ambient temperature variations, by taking advantage of their internal temperature sensors, in order to take into account the potential ambient temperature variations occurring during their operation.

Finally, we observe that ambient temperature variations are one of the most effective ways of attacking DRAM-based security mechanisms, and that the approaches proposed in the relevant literature in order to address the naturally occurring bit errors found in the responses of such PUFs at a particular ambient temperature level, such as error threshold levels [86] or bit selection schemes [87], are not able to successfully mitigate the effects of ambient temperature variations on these PUF responses. In particular, the dependency of DRAM retention-based PUFs and DRAM Row Hammer PUFs on ambient temperature variations can make them vulnerable to DoS attacks, as well as affect their integrity and, therefore, allow an attacker to successfully guess their PUF response and/or the relevant cryptographic tokens that are based on them, such as keys, etc. [143]. As such attacks are not difficult to implement, as they do not require a high level of expertise or resources, adequate mitigation methods and countermea-

sures against them, which may be similar to, or even the same as, the ones discussed, in a previous text segment, regarding low-temperature data remanence attacks against SRAM modules. Nevertheless, we note that the countermeasures proposed against such attacks will, inevitably, also suffer from the same potential issues, problems, and shortcomings, as the countermeasures proposed in order to address low-temperature data remanence attacks in the case of the SRAM PUF. To this end, we strongly suggest the employment of inherent system mechanisms for the amelioration of the effects of ambient temperature variations on these PUFs, as is the case with the cryptographic protocol proposed in Section 5.2, which takes advantage of intrinsic temperature sensors, in order to take into account the potential ambient temperature variations occurring during the operation of such PUFs as security mechanisms.

4.2.3 Flash-Memory-Based PUFs Under Ambient Temperature Variations

As already mentioned, a number of relevant works examine the effects of ambient temperature variations on Flash-memory-based PUFs [89, 147, 161, 162, 191, 193]. In general, these works observe that ambient temperature variations have a rather minor effect on the quality of the responses of Flash-memory-based PUFs. In this work, we will examine the effects of ambient temperature variations on NAND-Flash-memory-based PUF that utilise programming disturbances, utilising multiple instances of the Waveshare NandFlash Board (A), which is a removable external Flash memory module that incorporates a 1-Gbit Samsung K9F1G08U0E NAND Flash memory.

Again, we assume that each individual page of the relevant NAND Flash memory that is affected by the rapidly repeated programming of its nearby pages constitutes a different instance of this PUF, and examine the quality characteristics of the values of the cells of each such page after the operation of this PUF. However, we, once again, note that all such pages found in a single block, or in a particular device, could be considered as a single such PUF. Nevertheless, in this work, we assume that each odd-numbered page [39] of a single block of the Samsung K9F1G08U0E NAND Flash memory found on the Waveshare NandFlash Board (A) constitutes an individual PUF instance. Every even-numbered page of each relevant block is considered a "hammer" page and is to be constantly,i.e., rapidly and repeatedly, (re)programmed during the operation of this PUF. As each memory block of the Samsung K9F1G08U0E NAND Flash memory is made up of 64 pages, 32 of these pages will constitute individual PUF instances, with 31 of them (pages '1', '3', '5', ..., '61') receiving disturbances from both of their adjacent pages (double-side program "hammering"), and one of them (page '63') receiving disturbances from its only adjacent page (single-sided program "hammering", from page '62').

Additionally, we need to again note that the Waveshare NandFlash Board (A) is controlled using an ST Microelectronics STM32F429I Discovery (STM32F429I-DISC1) board, to which this Flash board has been connected using a Waveshare Open429Z-D Standard, an STM32F4 expansion/development board for the ST Microelectronics STM32F429I Discovery (STM32F429I-DISC1) board. A single block was fully erased and, thus, the initial bit pattern of cells utilised in the relevant PUF response was "`0xFF`" (all ones), while the other cells of the same block were rapidly and repeatedly programmed with the bit pattern "`0x00`" (all zeros), for 10,000 programming cycles, or until at least 2040 addresses (corresponding to 1

[39] With the numbering starting from page '0'.

B each [40]), on each page used as a PUF, have had at least one bit "flip" each [41], whichever condition was fulfilled first.

A Heraeus Vötsch HC 4005 climate chamber has been utilised in order to study the effects of ambient temperature variations on the relevant NAND-Flash-memory-based PUFs. The PUFs have been tested in the temperature range between 20°C and 60°C at intervals of 20°C. In this way, we examine whether it is possible for an attacker to modify the ambient temperature for the Waveshare NandFlash Board (A), in order to affect the operation of the relevant NAND-Flash-memory-based PUFs in a manner similar to which the DRAM-based PUFs examined in Section 4.2.2 can be affected. At each temperature tested, 20

[40] Byte addressing is used in this memory, as is the case for all the memories examined. Therefore, each address uniquely identifies one byte, which encodes two hexadecimal numbers. The first and the last four bits of such a byte, each encode a single hexadecimal.

[41] Thus, in case this condition holds true, in each of the 2040 bytes of each relevant PUF page, which consists of 2048 bytes in total, there will be at least one bit "flip", i.e., all but one of each PUF page's bytes will contain at least one bit "flip".

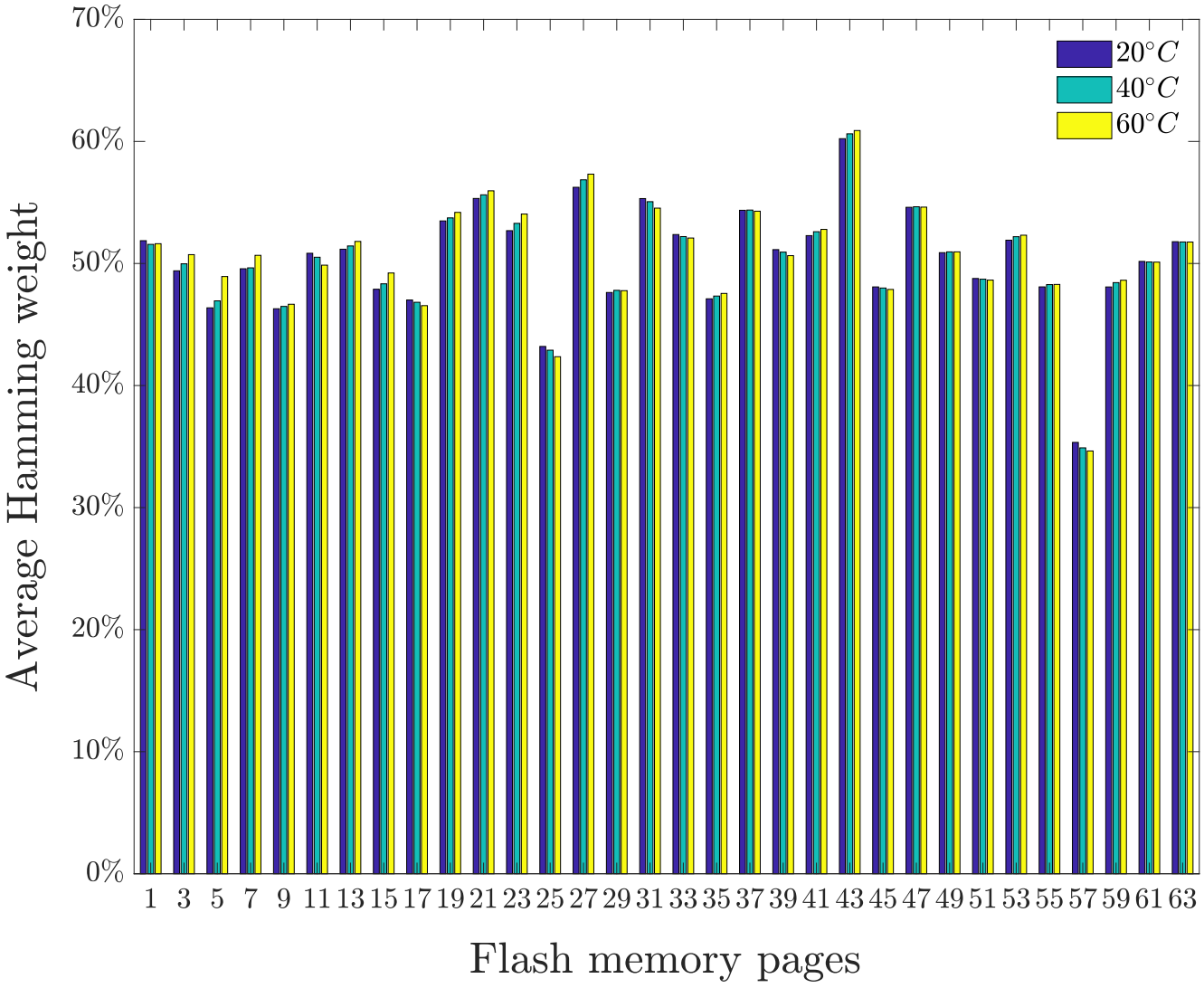


Figure 4.34: Average fractional Hamming weight values per instance of the examined NAND-Flash-memory-based PUFs that utilises programming disturbances, for the different values of the ambient temperature tested.

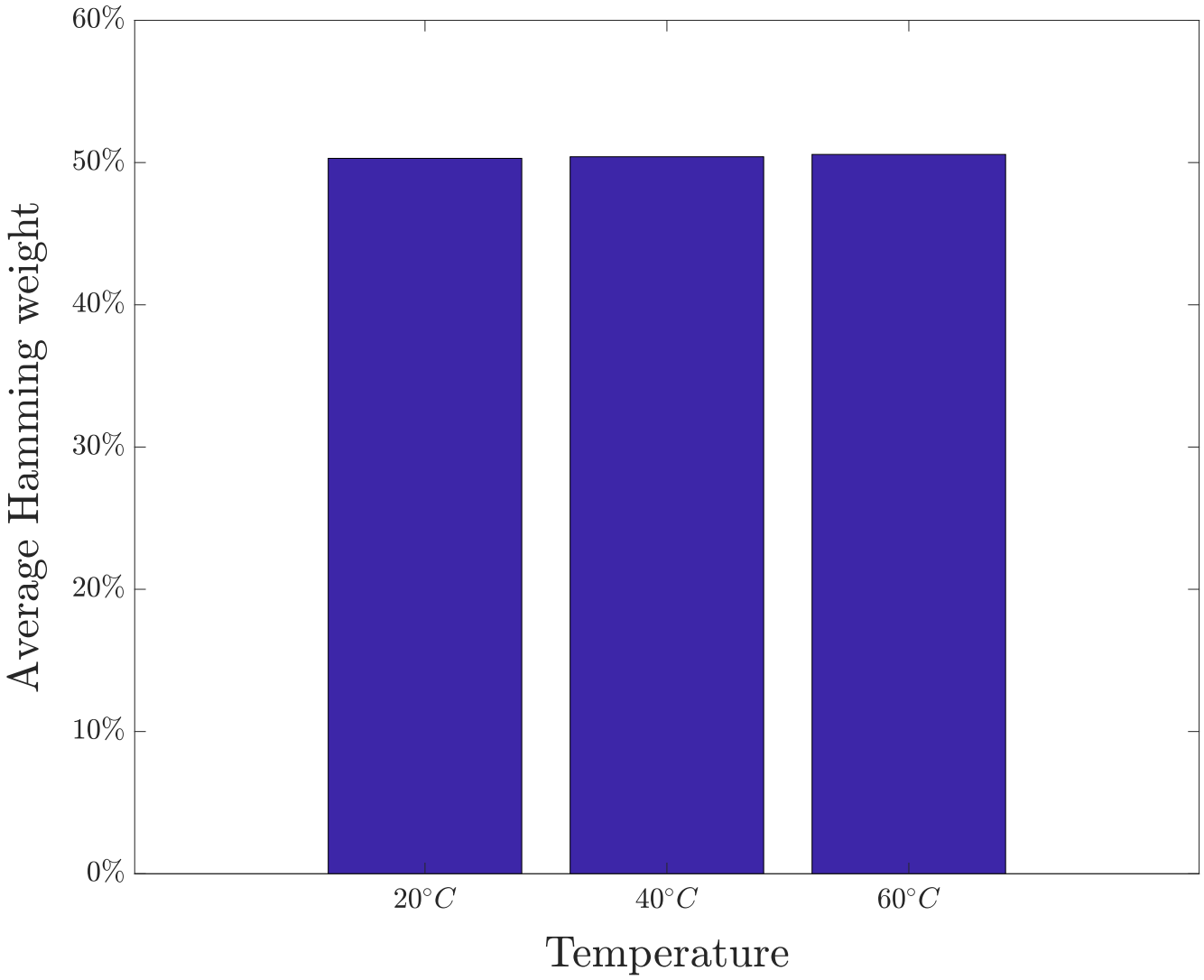


Figure 4.35: Overall average fractional Hamming weight value of the examined NAND-Flash-memory-based PUFs that utilises programming disturbances, for each value of the ambient temperature tested.

responses have been received for each page constituting a PUF, of a size of 2 KB each [42], for a total size of 64 KB per memory block measurement.

In order to examine how ambient temperature variations affect the quality characteristics of the NAND-Flash-memory-based PUFs implemented on the Waveshare NandFlash Board (A), we first examine the relevant average fractional Hamming weight values for each Flash memory constituting an instance of this PUF. In particular, as Figure 4.34 shows, these values are rather similar for all the temperatures tested, i.e., 20°C, 40°C, and 60°C, for each page. This can also be confirmed by Figure 4.35, which demonstrates the overall average fractional Hamming weight value for each level of ambient temperature tested. The overall fractional Hamming weight value for the responses of the NAND-Flash-memory-based PUFs implemented on the Waveshare NandFlash Board (A) is almost exactly 0.5 for all the ambient temperature values tested, i.e., 20°C, 40°C, and 60°C. However, we also note that the fractional

[42] Each relevant page comprises 2 KB of usable addresses and 64 B of spare addresses that are only used instead of segments ("words") of the relevant regular page addresses when these become unusable.

Hamming weight values for the different pages used as PUF instances are different from each other, in the same way that was noted in Section 4.1.3. Thus, we note that for all the power supply voltage values tested, the responses of particular PUF instances consistently correspond to a fractional Hamming weight value either significantly lower than the ideal value of 0.5, such as the responses corresponding to Flash memory pages '25', and '57', or significantly higher than the ideal value of 0.5, such as the responses corresponding to pages '27', and '43'.

In order to further examine the effects of ambient temperature variations on the relevant NAND-Flash-memory-based PUF, we have also computed the fractional intra-device [43] Hamming distance for all the measurements originating from the same Flash memory page, for all the pages being utilised as PUF instances, first for each pair of PUF responses taken at the same ambient temperature level, as shown in Figure 4.36, and, then, for each pair of PUF responses taken at different ambient temperature levels, as shown in Figure 4.37. In this way, we can know how different are the PUF responses measured from the same PUF instance at different ambient temperature conditions and, in this way, measure their robustness.

[43] Here, the term "device" is used to refer to an instance of the examined Flash-memory-based PUF, i.e., to each relevant page of this memory being utilised as a PUF.

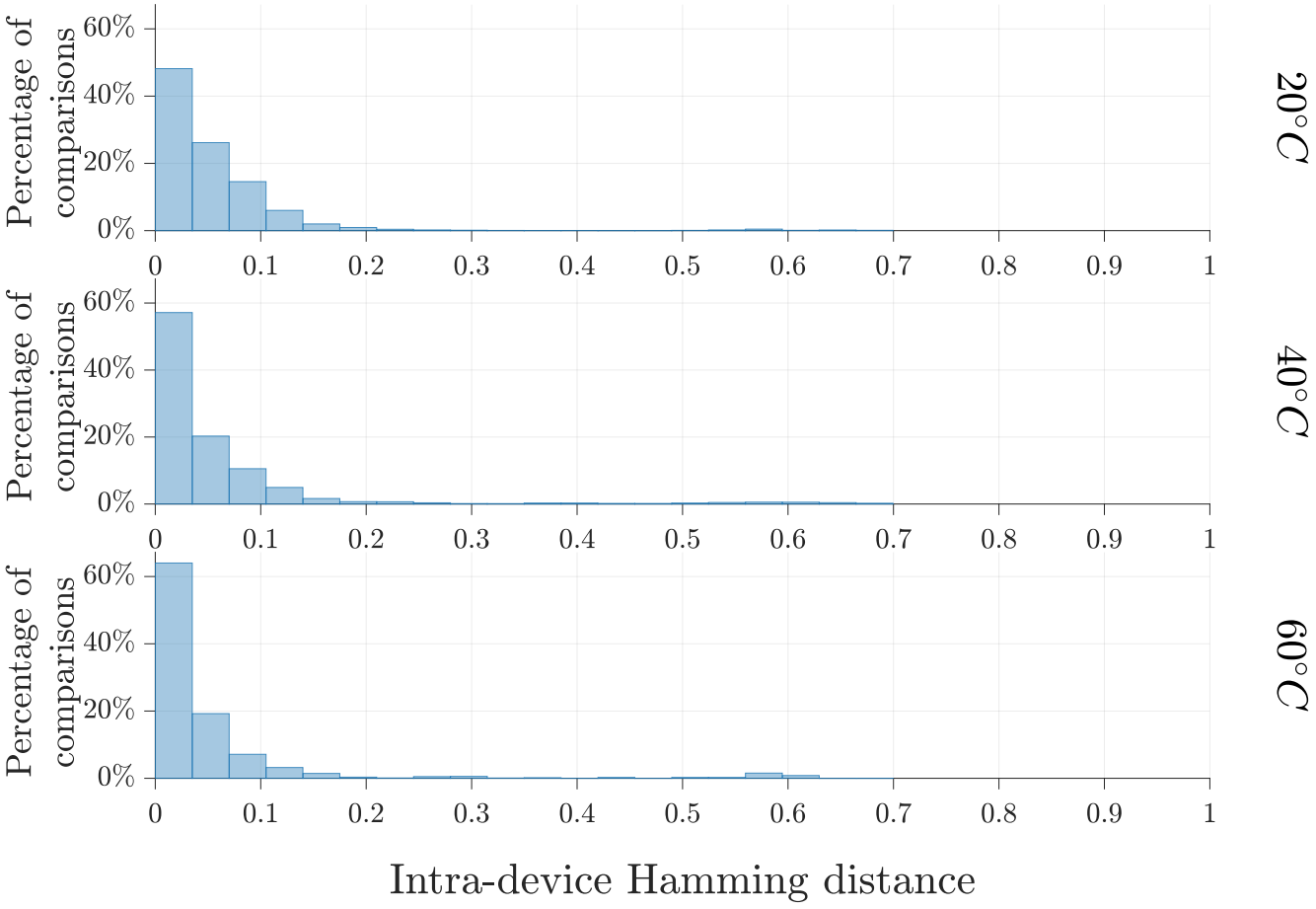


Figure 4.36: Intra-device Hamming distance values for the examined NAND-Flash-memory-based PUFs that utilises programming disturbances, for pairs of PUF responses taken at the same ambient temperature level.

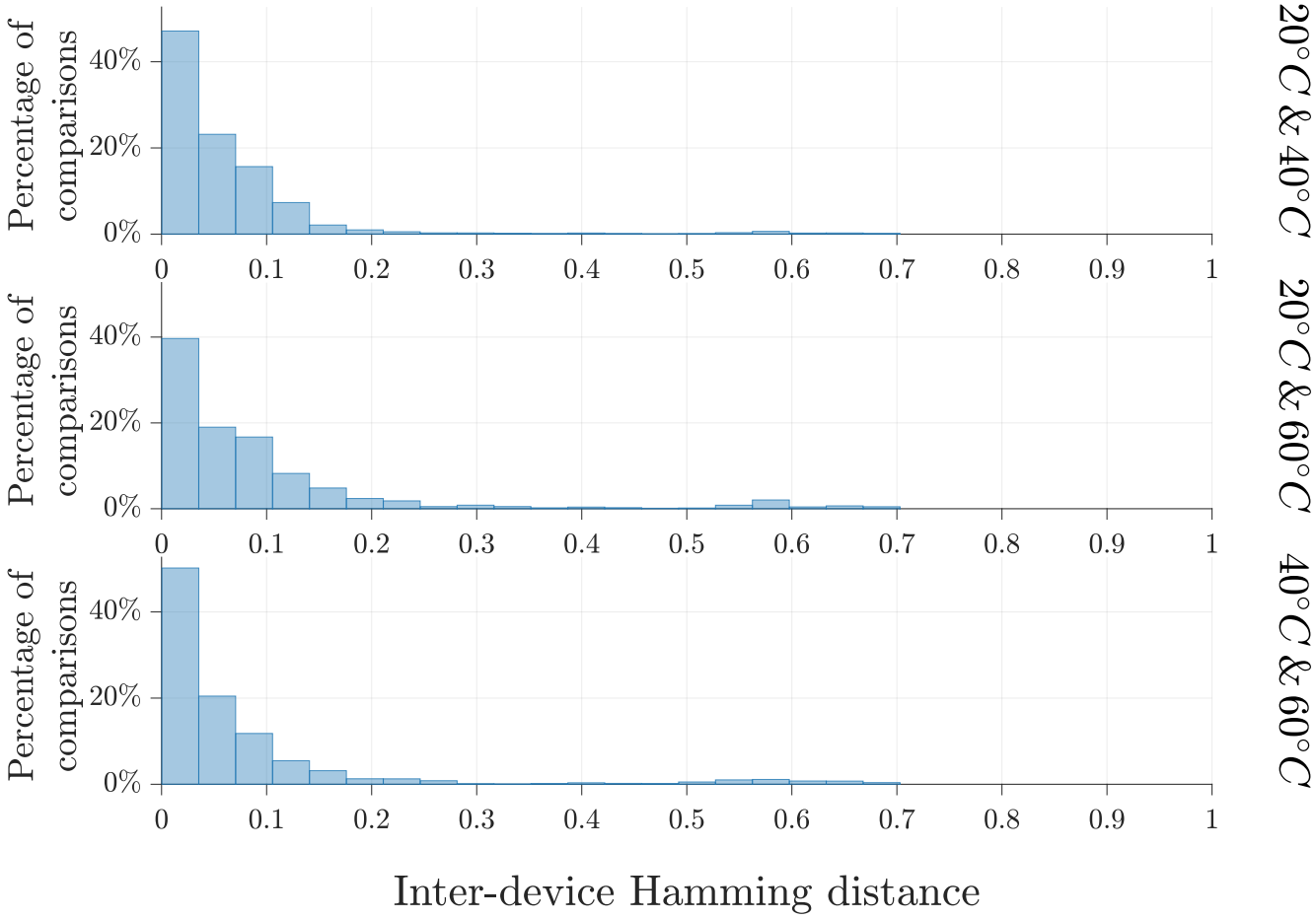


Figure 4.37: Intra-device Hamming distance values for the examined NAND-Flash-memory-based PUFs that utilises programming disturbances, for pairs of PUF responses taken at different ambient temperature levels.

We note that Figure 4.36 shows that the fractional intra-device Hamming distance values for pairs of PUF responses taken at all ambient temperature values tested, i.e., 20°C, 40°C, and 60°C, indicate a high degree of robustness, with most of the values being below 0.2, and only an extremely small number of outliers occurring around 0.6, in a fashion similar to the values noted under nominal conditions in Section 4.1.3. This is also reflected in Figure 4.37, as the fractional intra-device Hamming distance values for pairs of PUF responses for which one response was measured at 20°C and the other at 40°C, for which one response was measured at 20°C and the other at 60°C, and for which one response was measured at 40°C and the other at 60°C, all seem to be similar to the fractional intra-device Hamming distance values of this PUF under nominal conditions, which were presented Section 4.1.3.

Therefore, we note that ambient temperature variations do not seem to have an effect on the responses of the examined NAND-Flash-memory-based PUF. Thus, the examined NAND-Flash-memory-based PUF can provide an *acceptable* level of security under such ambient temperature variations. Nevertheless, of course, we note, once again, that the exclusion of particular PUF instances from usage, or the consideration of multiple Flash memory pages as a single PUF instance, could provide a lower level of variance for the fractional intra-device Hamming distance values of this PUF type under ambient temperature variations, and, thus, also a higher level of security in practice.

4.3 Regarding the Effects of Supply Voltage Variations on Memory-Based PUFs

In this section, we will briefly consider the effects of supply voltage changes on memory-based PUFs. It is worth noting that the effects of supply voltage variations on memory-based PUF are rather hard to investigate, as most of the devices that host the relevant memory components also include voltage regulator components that are most likely to prevent the effects of supply voltage variations from reaching the relevant memories. Therefore, relevant works utilise either external memory modules, or even ASICs, that serve as dedicated PUFs, while the majority of the available bibliography refrains from examining the effects of supply voltage variations on memory-based PUFs. We also observe that attacks based on the effects of supply voltage variations can only be considered as practical, if they do not require the detachment of the inherent memory component of the system that is being used as a PUF, as this action would be immediately noticed, or physical tampering with the host device itself, as this action would almost certainly require a high level of expertise and/or the utilisation of expensive equipment, and would also, sooner or later, be noticed. Therefore, in the case of intrinsic PUFs, only attacks that are based on variations of the supply voltage provided to the overall system can be considered as rather practical, while in the case of external memories being utilised as PUFs, attacks that are based on the variation of the supply voltage of such external memory-based PUFs could also be considered as practical, especially if the relevant memory modules are removable.

Moreover, we need to clarify here that we will refrain from examining the effects of supply voltage variations on SRAM PUFs, as Katzenbeisser et al. [185, 186] have examined the effects of voltage variations directly on several dedicated SRAM PUFs that had been incorporated into a number of ASICs, alongside various instances of other PUF types. As these works [185, 186] note, the reduction and the increase of the voltage supplied directly to these dedicated SRAM PUF modules had no noticable effect on the quality of their PUF responses. Therefore, it would be rather counterproductive to investigate further the effects of supply voltage variations on SRAM PUFs, especially by varying the supply voltage of the overall device, and not only the voltage supplied to the relevant SRAM module itself. Nevertheless, we also note that Lipps et al. [113] did also investigate the effects of supply voltage variations on an SRAM PUF that was implemented on a COTS device, namely on an ATMEGA 2560-16AU MCU which was part of an Arduino MEGA ADK R3 device, and also noted that such effects were rather insignificant, if any existed at all.

Therefore, in this subsection, we will proceed to examine the effects of supply voltage variations on a number of more novel DRAM-based and Flash-memory-based PUFs, namely on the DRAM decay-based PUF, the DRAM Row Hammer PUF, and the NAND-Flash-memory-based PUF that utilises programming disturbances. We note the existence of relevant literature regarding the effects of supply voltage variations on DRAM-based PUFs [48, 79, 80, 83, 85, 144], as well as literature regarding the effects of supply voltage variations on Flash-memory-based PUFs [193], although the latter can be considered as rather sparse, at best. Therefore, we will attempt to investigate further, in this work, the effects of power supply voltage variations on such PUFs, in order to examine whether their quality characteristics remain the same, and they can continue to provide an *acceptable* level of security, as is the case with the SRAM PUF, or if they rapidly deteriorate in the presence of such variations, in a way similar to the effects that temperature variations may have upon certain memory-based PUFs, e.g., the DRAM Row Hammer PUF, as discussed in Section 4.2.2.

4.3.1 A Brief Examination of the Effects of Supply Voltage Variations on DRAM PUFs

As already discussed, the ability of DRAM-based PUFs to provide adequate security in the presence of supply voltage variations has been examined in the relevant literature. In general, most works agree that, when supply voltage variations are applied directly to the DRAM modules that have been utilised for the implementation of the relevant PUFs, then the responses of the relevant DRAM-based PUFs are affected by such voltage variations, but, in most cases, not to a degree that can preclude the utilisation of these DRAM-based PUFs as adequate security mechanisms [48, 79, 80, 83, 85, 144]. Nevertheless, as we have already explained, the application of supply voltage variations directly to the relevant DRAM module can only be considered as practical when an external DRAM-based PUF is utilised. As the industry strives to decrease production costs, especially in the context of the IoT, in most cases, the relevant memory-based PUFs being proposed as security mechanisms must be intrinsic, in order to be considered for adoption in practical applications [44]. Thus, in this work, we will examine the effects of supply voltage variations with regards to the voltage provided to the overall device that incorporates either a DRAM decay-based PUF, i.e., in our experiments, an Intel Galileo Gen 2 board, or a DRAM Row Hammer PUF, i.e., in our experiments, a PandaBoard ES Rev B3.

In general, we assume that an adversary can potentially modify the power supply voltage for both the Intel Galileo Gen 2 boards that incorporate the relevant DRAM retention-based PUFs and the PandaBoard ES Rev B3 devices that incorporate the relevant DRAM Row Hammer PUFs, either completely unnoticed or, potentially, sometimes, causing the board to just reboot. As IoT devices are often placed in remote locations or may not be accessed all the time, a potential reset of the device may go unnoticed, without any other cautionary signs. Therefore, if voltage variations have a significant effect on the operation of the examined PUFs, this should be taken into account in regards to their role as potential security mechanisms for IoT.

As we have already discussed, the quality of the responses of these PUFs is strongly related to the number of bit "flips", i.e., the changes in the logical values, that occur in the cells that are utilised for the production of the relevant PUF responses. Thus, we have measured the number of bit "flips" that are present in the responses of the DRAM retention-based PUFs that have been implemented on the Intel Galileo Gen 2 boards, and in the responses of the DRAM Row Hammer PUFs that have been implemented on the PandaBoard ES Rev B3 devices, when different supply voltages were provided to these devices. We note that the bit "flips" occurring in the responses of these PUF types can provide some insights into the entropy of these responses and, therefore, into the level of security these PUFs can provide. We also need to note that we tested the two different types of boards within power supply voltage ranges at which they were still operational, and at which the voltage being supplied would not cause permanent damages to them, i.e., 6V to 12V for the Intel Galileo Gen 2 boards, and 4.5V to 5.5V for the PandaBoard ES Rev B3 devices. Additionally, we note that the Intel Galileo Gen 2 board can be powered through either a power jack socket or a set of two power pins, while the PandaBoard ES Rev B3 can be powered through either a power jack socket or a USB port. In our experiments, we have tested both ways of providing power to these two boards. We, finally, note that the PandaBoard ES Rev B3 has an LED indicating power

[44] Certain Advanced Reconfigurable PUFs (AR-PUFs) that are based on the combination of intrinsic PUFs and external ones can be considered as an exception to this, as they are able to provide *advanced* security applications.

supply voltage overload, nevertheless, the board can still be damaged if, e.g., it is supplied with reverse (incorrect) polarity current.

Our results regarding the number of bit "flips" occurring, for different supply voltages, in the responses of the DRAM retention-based PUFs that have been implemented on the Intel Galileo Gen 2 boards, and in the responses of the DRAM Row Hammer PUFs that have been implemented on the PandaBoard ES Rev B3 devices, are presented in Figure 4.38 and Figure 4.39 respectively. The *fractional number of bit "flips"* shown in these Figures denotes the total number of cells utilised in the relevant PUF responses that have "flipped", i.e., have changed their logical values, during the operation of the relevant DRAM-based PUFs, divided by the total number of cells utilised in the relevant PUF responses. In general, our results do not seem to indicate a significant change in the operation and/or in the quality characteristics of these PUFs when different power supply voltages are provided. We do note, however, a

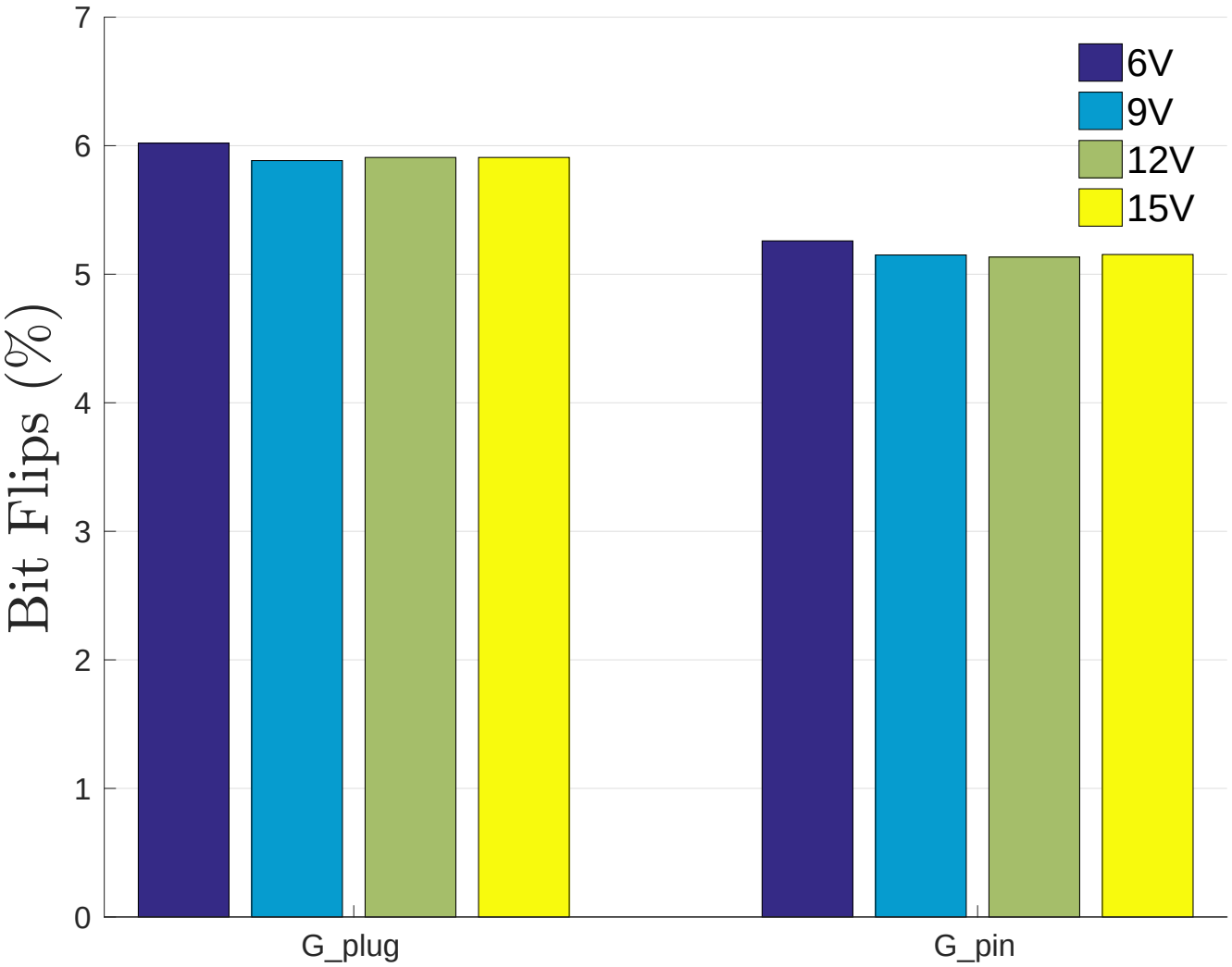


Figure 4.38: Average fractional number of bit "flips" observed in the responses of a DRAM retention-based PUF implemented on the Intel Galileo Gen 2 board, for a power supply voltage of 6V, 9V, 12V, and 15V, that has been provided either through the board's power supply pins or its power jack socket at 40°C. The DRAM region being employed as a PUF is 8 KB and the decay time, i.e., the time period during which the DRAM refresh operation has been suspended, is 300s. Initially, the values of the cells utilised in the relevant PUF responses had all been set to logical '0'.

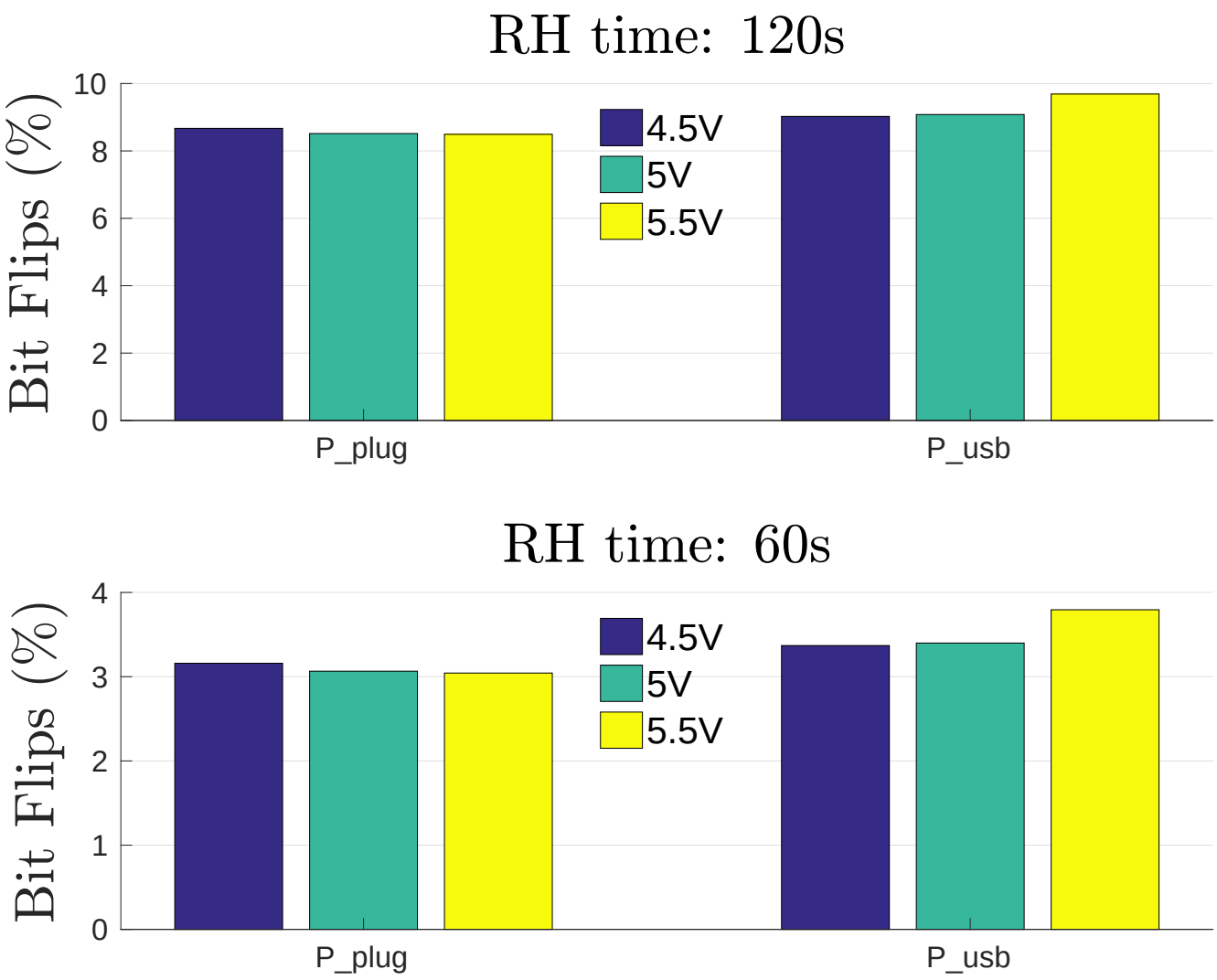


Figure 4.39: Average fractional number of bit "flips" observed in the responses of a DRAM Row Hammer PUF implemented on the PandaBoard ES Rev B3, for a power supply voltage of 4.5V, 5V, and 5.5V, that has been provided either through the board's USB port or its power jack socket at 40°C. The DRAM region being employed as a PUF is 128 KB, and two row hammer times, i.e., time periods during which the DRAM refresh operation has been suspended and the relevant hammer rows have been being rapidly and repeatedly accessed, have been examined, 60s and 120s. The rest of the relevant configuration: `RH type=DSRH`, `PUF row IV="0xAA"`, `hammer row IV="0x55"`, `Cache` disabled, according to Table 4.1.

noticeable difference in the number of bit "flips" occurring when the DRAM retention-based PUF on the Intel Galileo is supplied through the relevant power jack socket and when supplied through the relevant power pins.

Additionally, we have also examined the intra-device Jaccard index values for the responses of the DRAM retention-based PUFs that have been implemented on the Intel Galileo Gen 2 boards, and for the responses of the DRAM Row Hammer PUFs that have been implemented on the PandaBoard ES Rev B3, considering, for each comparison relevant to an intra-device Jaccard index value, only responses stemming from the same board type, which have been collected using the same way of supplying power, for each of the two ways of supplying power to each of the two types of boards, for any of the different supply voltage levels utilised for each of the two board types.

As our results, which are presented in Figure 4.40, for the DRAM retention-based PUFs implemented on the Intel Galileo Gen 2 boards, and in Figure 4.41, for the DRAM Row Hammer PUFs implemented on the PandaBoard ES Rev B3, indicate, measurements stemming from the same board type, which have been collected using the same way of supplying power, for each of the two ways of supplying power to each of the two types of boards, for any of the different supply voltage levels utilised for each of the two board types, seem to be very highly similar to each other, for each of the two PUF types being tested.

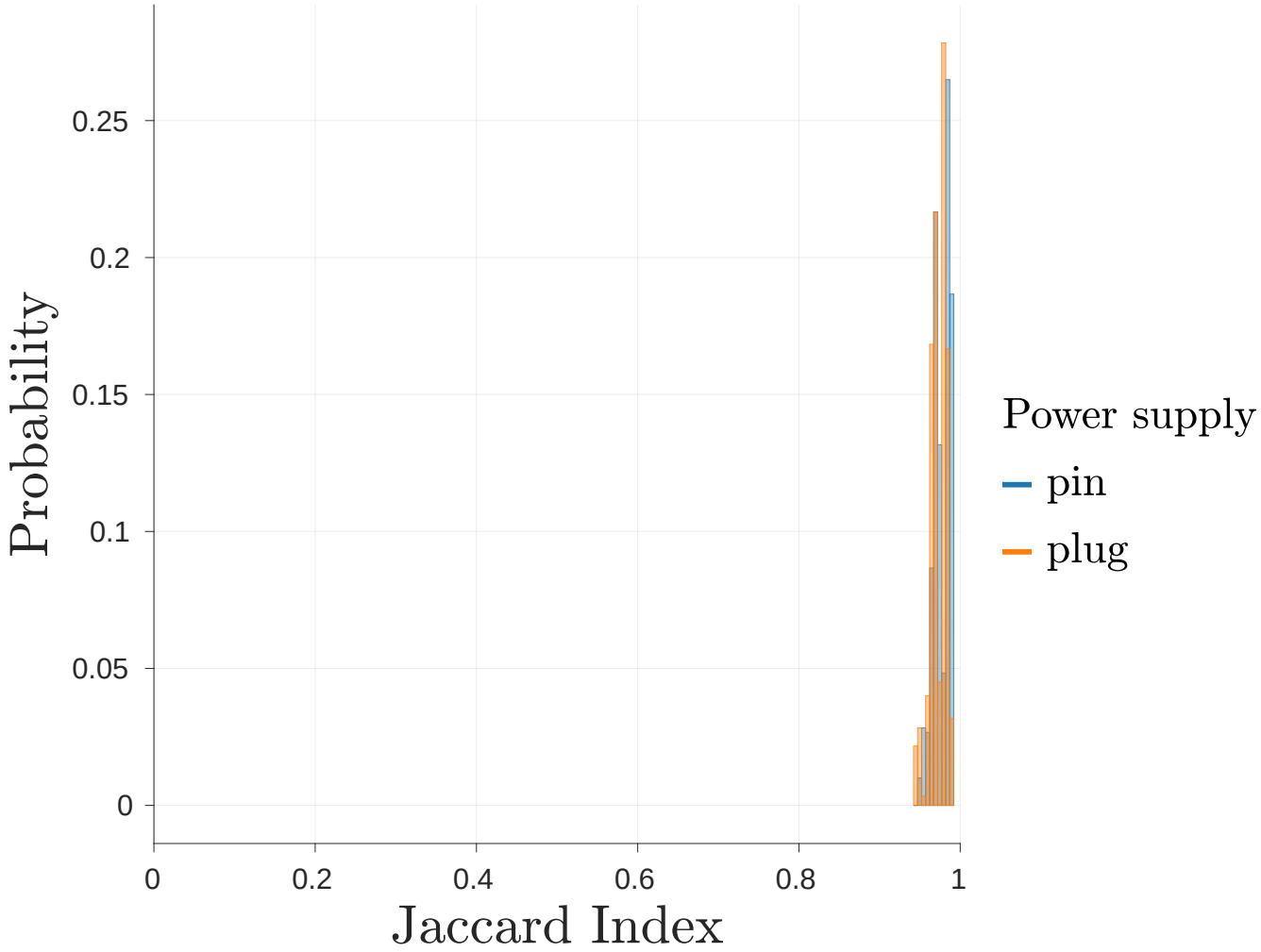


Figure 4.40: Intra-device Jaccard index values for DRAM retention-based PUF responses taken at all the different supply voltages using the same way of supplying power to the Intel Galileo Gen 2 board, either through its power supply pins or its power jack socket at 40°C. The DRAM region being employed as a PUF is 8 KB and the decay time, i.e., the time period during which the DRAM refresh operation has been suspended, is 300s. Initially, the values of the cells utilised in the relevant PUF responses had all been set to logical '0'.

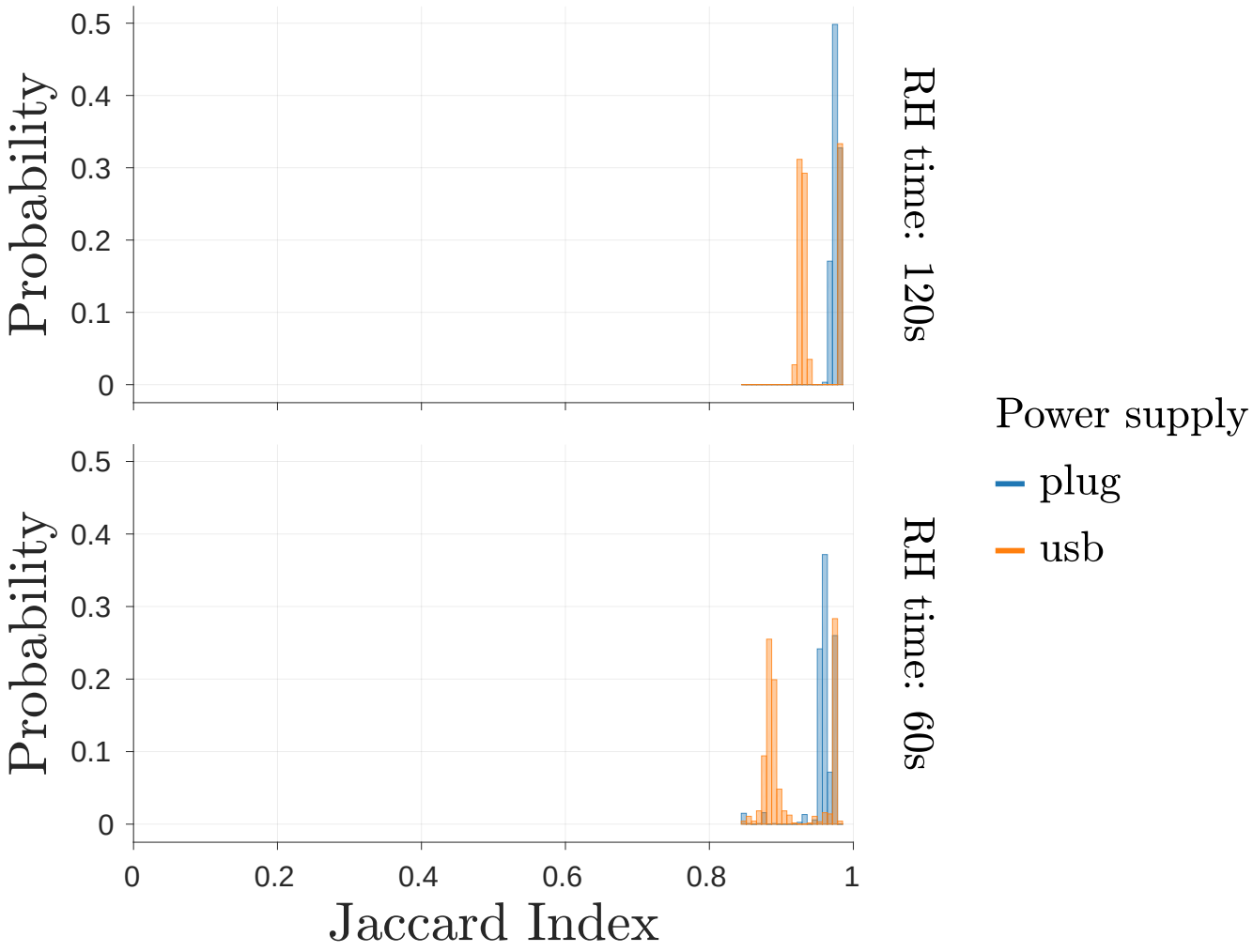


Figure 4.41: Intra-device Jaccard index values for DRAM Row Hammer PUF responses taken at all the different supply voltages using the same way of supplying power to the PandaBoard ES Rev B3, either through its USB port or its power jack socket at 40°C. The DRAM region being employed as a PUF is 128 KB, and two row hammer times, i.e., time periods during which the DRAM refresh operation has been suspended and the relevant hammer rows have been being rapidly and repeatedly accessed, have been examined, 60s and 120s. The rest of the relevant configuration: `RH type=DSRH`, `PUF row IV="0xAA"`, `hammer row IV="0x55"`, `Cache` disabled, according to Table 4.1.

Finally, we have also examined the intra-device Jaccard index values for the responses of the DRAM retention-based PUFs that have been implemented on the Intel Galileo Gen 2 boards, and for the responses of the DRAM Row Hammer PUFs that have been implemented on the PandaBoard ES Rev B3, considering, for each comparison relevant to an intra-device Jaccard index value, only responses stemming from the same board type, which have been collected at each voltage level supplied to the relevant boards, but for different ways of supplying power to them, i.e., through the power supply pins and the power jack socket of the Intel Galileo Gen 2 boards and through the USB port and the power jack socket of the PandaBoard ES Rev B3 devices.

Our results, which are presented in Figure 4.42, for the DRAM retention-based PUFs implemented on the Intel Galileo Gen 2 boards, and in Figure 4.43, for the DRAM Row Hammer PUFs implemented on the PandaBoard ES Rev B3, clearly show that measurements stemming from the same board type,

which have been collected at each voltage level supplied to the relevant boards, but for different ways of supplying power to them, i.e., through the power supply pins and the power jack socket of the Intel Galileo Gen 2 boards and through the USB port and the power jack socket of the PandaBoard ES Rev B3 devices, are highly similar to each other, for each of the two PUF types being tested.

We can, therefore, conclude that variations in the supply voltage provided either to Intel Galileo Gen 2 boards that incorporate DRAM retention-based PUFs, or to PandaBoard ES Rev B3 devices incorporating DRAM Row Hammer PUFs, do not seem to significantly affect the operation of these PUFs or to affect the quality characteristics of their responses. Therefore, both types of PUF seem to be robust to external voltage variations, as long as these variations are not so extreme as to damage permanently their host IoT devices. In any case, however, a way in which the potential effects of supply voltage variations on DRAM-based PUFs could be reduced has already been proposed in the relevant literature [48].

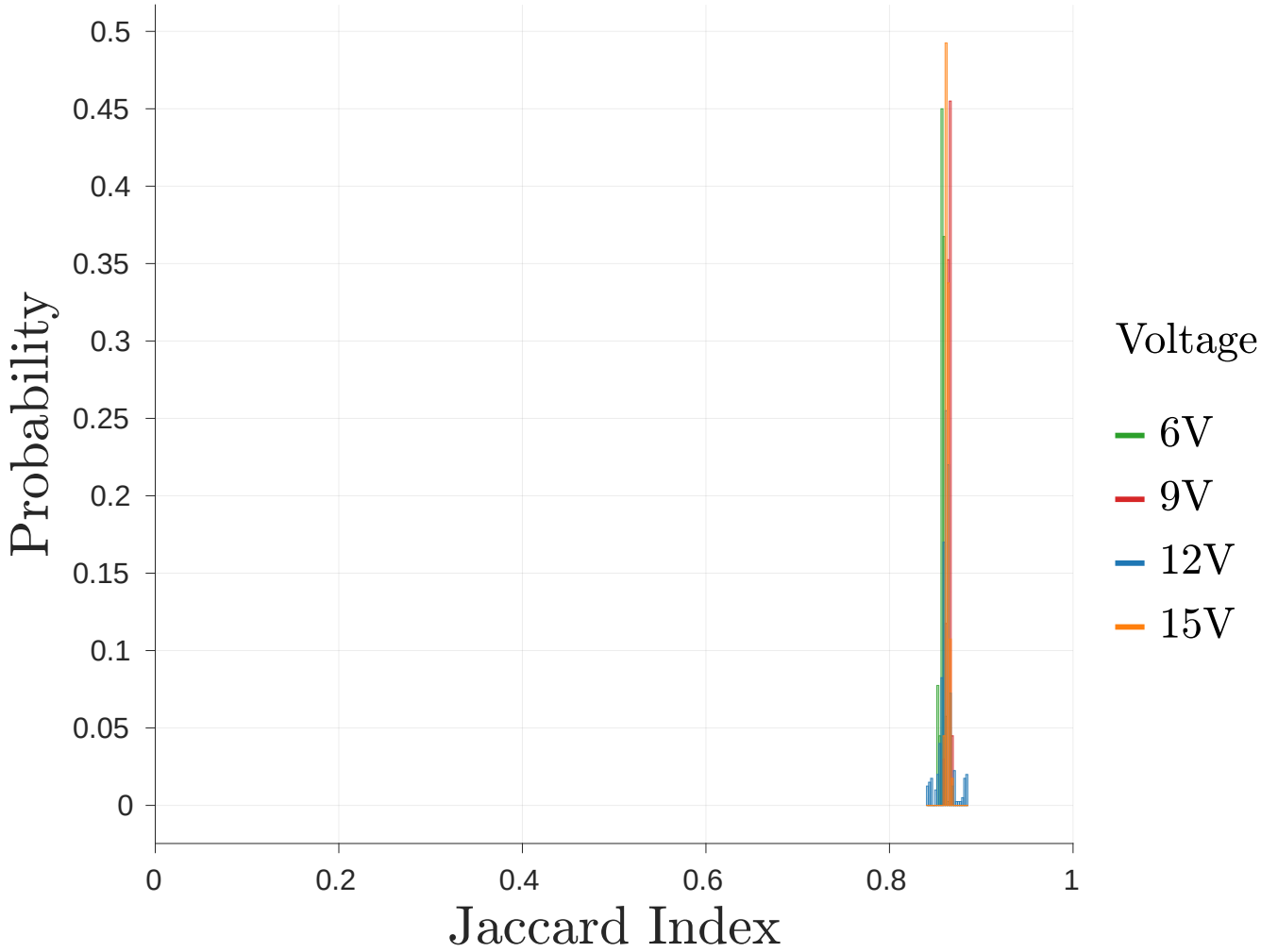


Figure 4.42: Intra-device Jaccard index values for DRAM retention-based PUF responses taken at each supply voltage (6V, 9V, 12V, and 15V) for the two different ways of supplying power to the Intel Galileo Gen 2 board, i.e., through its power supply pins and its power jack socket at 40°C. The DRAM region being employed as a PUF is 8 KB and the decay time, i.e., the time period during which the DRAM refresh operation has been suspended, is 300s. Initially, the values of the cells utilised in the relevant PUF responses had all been set to logical '0'.

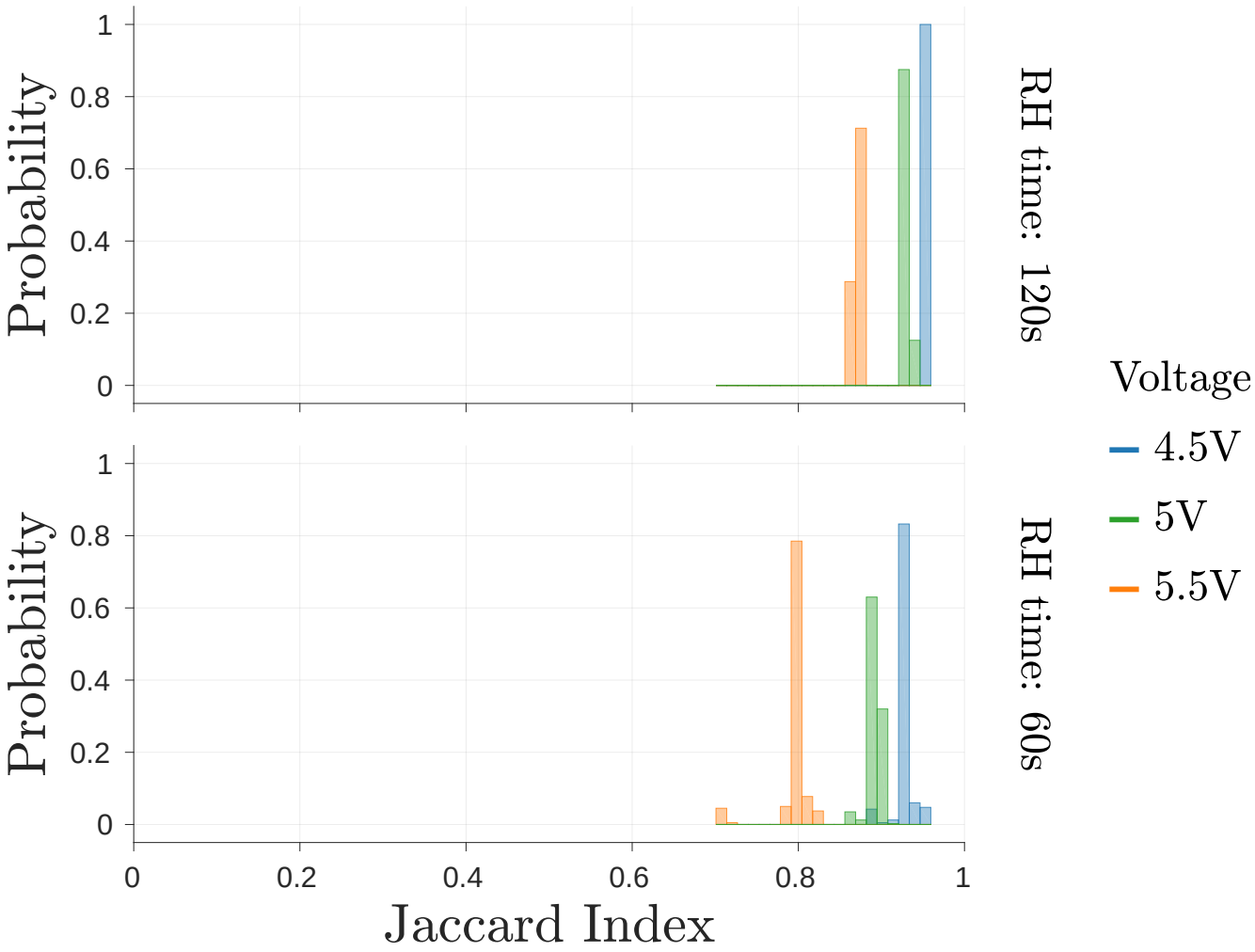


Figure 4.43: Intra-device Jaccard index values for DRAM Row Hammer PUF responses taken at each supply voltage (4.5V, 5V, and 5.5V) for the two different ways of supplying power to the PandaBoard ES Rev B3, i.e., through its USB port and its power jack socket at 40°C. The DRAM region being employed as a PUF is 128 KB, and two row hammer times, i.e., time periods during which the DRAM refresh operation has been suspended and the relevant hammer rows have been being rapidly and repeatedly accessed, have been examined, 60s and 120s. The rest of the relevant configuration: `RH type=DSRH`, `PUF row IV="0xAA"`, `hammer row IV="0x55"`, `Cache` disabled, according to Table 4.1.

4.3.2 A Brief Examination of the Effects of Supply Voltage Variations on NAND-Flash-Memory-Based PUFs Utilising Programming Disturbances

As already mentioned, the relevant literature regarding the effects of supply voltage variations on Flash-memory-based PUFs appears to be rather limited [193]. Nevertheless, supply voltage variations seem to have an effect on such PUFs, with more research being required in order to determine the extent and significance of these effects. For this reason, we will examine here the effects of supply voltage variations on NAND-Flash-memory-based PUF that utilise programming disturbances, utilising multiple instances of the Waveshare NandFlash Board (A), which is a removable external Flash memory module that incorporates a 1-Gbit Samsung K9F1G08U0E NAND Flash memory.

Again, we assume that each individual page of the relevant NAND Flash memory that is affected by the rapidly repeated programming of its nearby pages constitutes a different instance of this PUF, and examine the quality characteristics of the values of the cells of each such page after the operation of this PUF. However, we, once again, note that all such pages found in a single block, or in a particular device, could be considered as a single such PUF. Nevertheless, in this work, we assume that each odd-numbered page [45] of a single block of the Samsung K9F1G08U0E NAND Flash memory found on the Waveshare NandFlash Board (A) constitutes an individual PUF instance. Every even-numbered page of each relevant block is considered a "hammer" page and is to be constantly,i.e., rapidly and repeatedly, (re)programmed during the operation of this PUF. As each memory block of the Samsung K9F1G08U0E NAND Flash memory is made up of 64 pages, 32 of these pages will constitute individual PUF instances, with 31 of them (pages '1', '3', '5', ..., '61') receiving disturbances from both of their adjacent pages (double-side program "hammering"), and one of them (page '63') receiving disturbances from its only adjacent page (single-sided program "hammering", from page '62').

Additionally, we need to again note that the Waveshare NandFlash Board (A) is controlled using an ST Microelectronics STM32F429I Discovery (STM32F429I-DISC1) board, to which this Flash board has been connected using a Waveshare Open429Z-D Standard, an STM32F4 expansion/development board for the ST Microelectronics STM32F429I Discovery (STM32F429I-DISC1) board. A single block was fully erased and, thus, the initial bit pattern of cells utilised in the relevant PUF response was "`0xFF`" (all ones), while the other cells of the same block were rapidly and repeatedly programmed with the bit pattern "`0x00`" (all zeros), for 10,000 programming cycles, or until at least 2040 addresses (corresponding to 1 B each [46]), on each page used as a PUF, have had at least one bit "flip" each [47], whichever condition was fulfilled first.

In order to examine the effects of supply voltage variations on the relevant NAND-Flash-memory-based PUFs, we have modified the supply voltage provided to the overall system through the USB port of the ST Microelectronics STM32F429I Discovery (STM32F429I-DISC1) board, as well as the supply voltage provided by the relevant pins of the ST Microelectronics STM32F429I Discovery (STM32F429I-DISC1) board, which provide either 3.3V or 5V, and which are also connected to the corresponding pins and ports of the relevant Waveshare Open429Z-D Standard, the STM32F4 expansion/development board that provides an interface between the ST Microelectronics STM32F429I Discovery (STM32F429I-DISC1) board and the Waveshare NandFlash Board (A). In this way, we examine whether it is possible for an attacker to modify the supply voltage for the Waveshare NandFlash Board (A), without the need to disconnect it from the relevant Waveshare Open429Z-D Standard board. In general, we assume that an attacker can easily change the supply voltage provided to the overall system through the USB port of the ST Microelectronics STM32F429I Discovery (STM32F429I-DISC1) board without being noticed. Additionally, we also assume that changing the voltage provided through the pins of the ST Microelectronics STM32F429I Discovery (STM32F429I-DISC1) board and the Waveshare Open429Z-D Standard board would also be possible, although, in this case, the attacker would need to have physical access to

[45] With the numbering starting from page '0'.

[46] Byte addressing is used in this memory, as is the case for all the memories examined. Therefore, each address uniquely identifies one byte, which encodes two hexadecimal numbers. The first and the last four bits of such a byte, each encode a single hexadecimal.

[47] Thus, in case this condition holds true, in each of the 2040 bytes of each relevant PUF page, which consists of 2048 bytes in total, there will be at least one bit "flip", i.e., all but one of each PUF page's bytes will contain at least one bit "flip".

either of the two boards, i.e., to the overall system, as the two boards are connected to each other, and the Waveshare Open429Z-D Standard board is also connected to the Waveshare NandFlash Board (A). All the measurements were performed at 25°C, and, for each relevant test case, 20 responses have been received for each page constituting a PUF, of a size of 2 KB each [48], for a total size of 64 KB per block measurement.

We also need to note that we performed our tests within power supply voltage ranges at which the relevant boards would still be operational, and at which the voltage being supplied to them would not cause permanent damages to them, i.e., 3V to 7V for the supply voltage provided to the overall system through the USB port of the ST Microelectronics STM32F429I Discovery (STM32F429I-DISC1) board, whose nominal value would be 5V, 3V to 7V for the supply voltage provided by the pins of the ST Microelectronics STM32F429I Discovery (STM32F429I-DISC1) board and the relevant Waveshare Open429Z-D Standard that would nominally provide 5V, and 2.3V to 4.3V for the supply voltage provided by the pins of the ST Microelectronics STM32F429I Discovery (STM32F429I-DISC1) board and the relevant Waveshare Open429Z-D Standard that would nominally provide 5V. Finally, we need to note that the Waveshare NandFlash Board (A) seems to be powered with 3.3V from the relevant port of the Waveshare Open429Z-D Standard board.

Examining the Effects of Power Supply Voltage Variations on the NAND-Flash-Memory-Based PUFs Implemented on the Waveshare NandFlash Board (A) by Varying the Voltage Supplied by the USB Port of the STM32F429I Discovery (STM32F429I-DISC1) Board to the Overall System of Devices

In order to examine how variations in the voltage supplied by the USB port of the STM32F429I Discovery (STM32F429I-DISC1) board to the overall system affect the quality characteristics of the NAND-Flash-memory-based PUFs implemented on the Waveshare NandFlash Board (A), we first examine the relevant average fractional Hamming weight values for each Flash memory constituting an instance of this PUF. In particular, as Figure 4.44 shows, these values are similar for 5V and 7V, for each page, while the fractional Hamming weight values are noticably lower for 3V. We note that the fractional Hamming weight values for the different pages used as PUF instances are different from each other, in the same way that was noted in Section 4.1.3. Thus, we note that for the power supply voltage values of 5V and 7V, the responses of particular PUF instances consistently correspond to a fractional Hamming weight value either significantly lower than the ideal value of 0.5, such as the responses corresponding to Flash memory pages '25', and '57', or significantly higher than the ideal value of 0.5, such as the responses corresponding to pages '27', and '43'. The same trend is followed also for a power supply voltage value of 3V, but in this case, the fractional Hamming weight values are, for almost all pages, much lower than the relevant values for 5V and 7V. This can be confirmed by Figure 4.45, which demonstrates the overall average fractional Hamming weight value for each level of supply voltage used. The overall fractional Hamming weight value for the responses of the NAND-Flash-memory-based PUFs implemented on the Waveshare NandFlash Board (A) is almost exactly 0.5 for 5V and 7V, and below 0.45 for 3V.

[48] Each relevant page comprises 2 KB of usable addresses and 64 B of spare addresses that are only used instead of segments ("words") of the relevant regular page addresses when these become unusable.

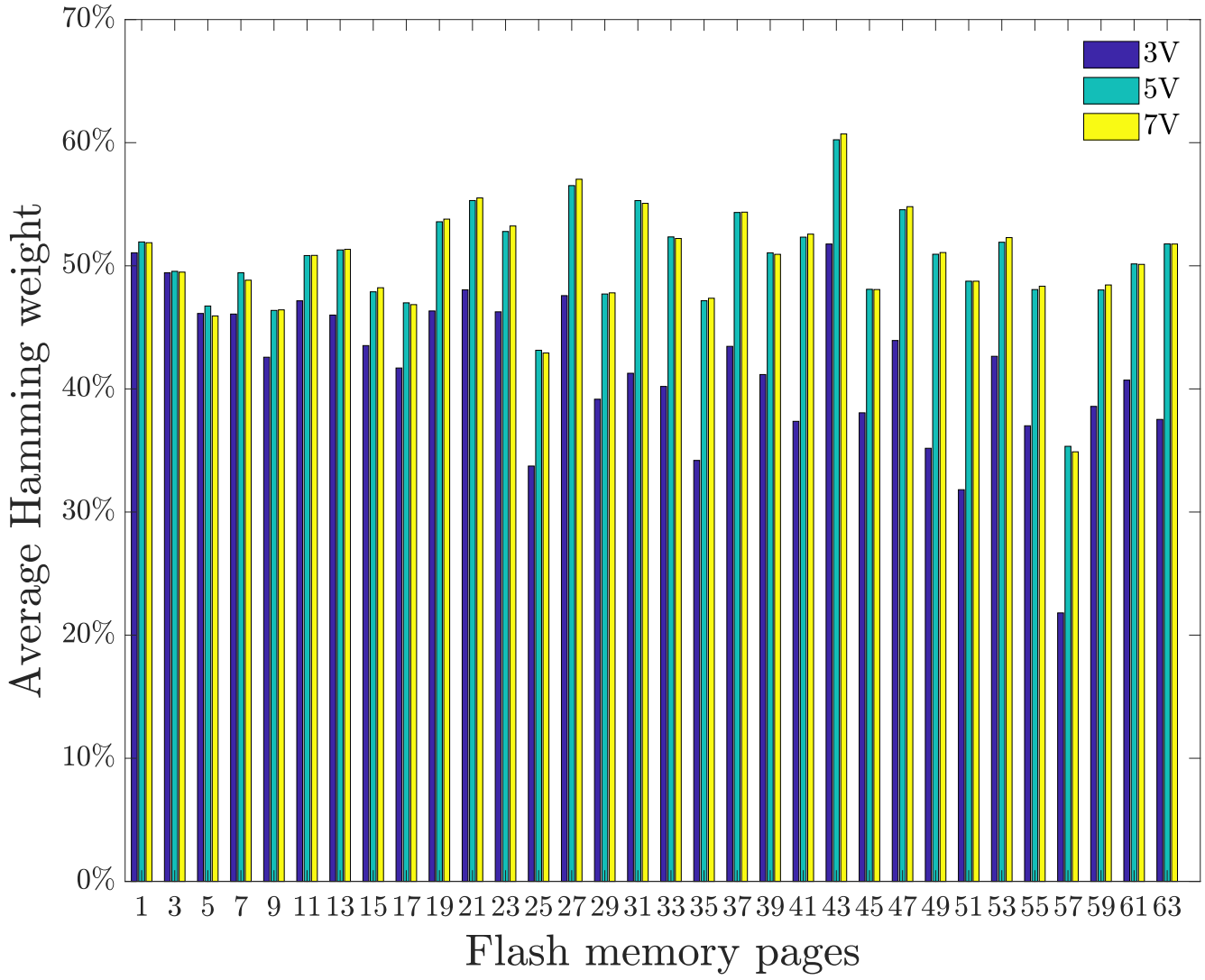


Figure 4.44: Average fractional Hamming weight values per instance of the examined NAND-Flash-memory-based PUFs that utilises programming disturbances, for the different values of power supply voltage provided by the USB port of the STM32F429I Discovery (STM32F429I-DISC1) board to the overall system.

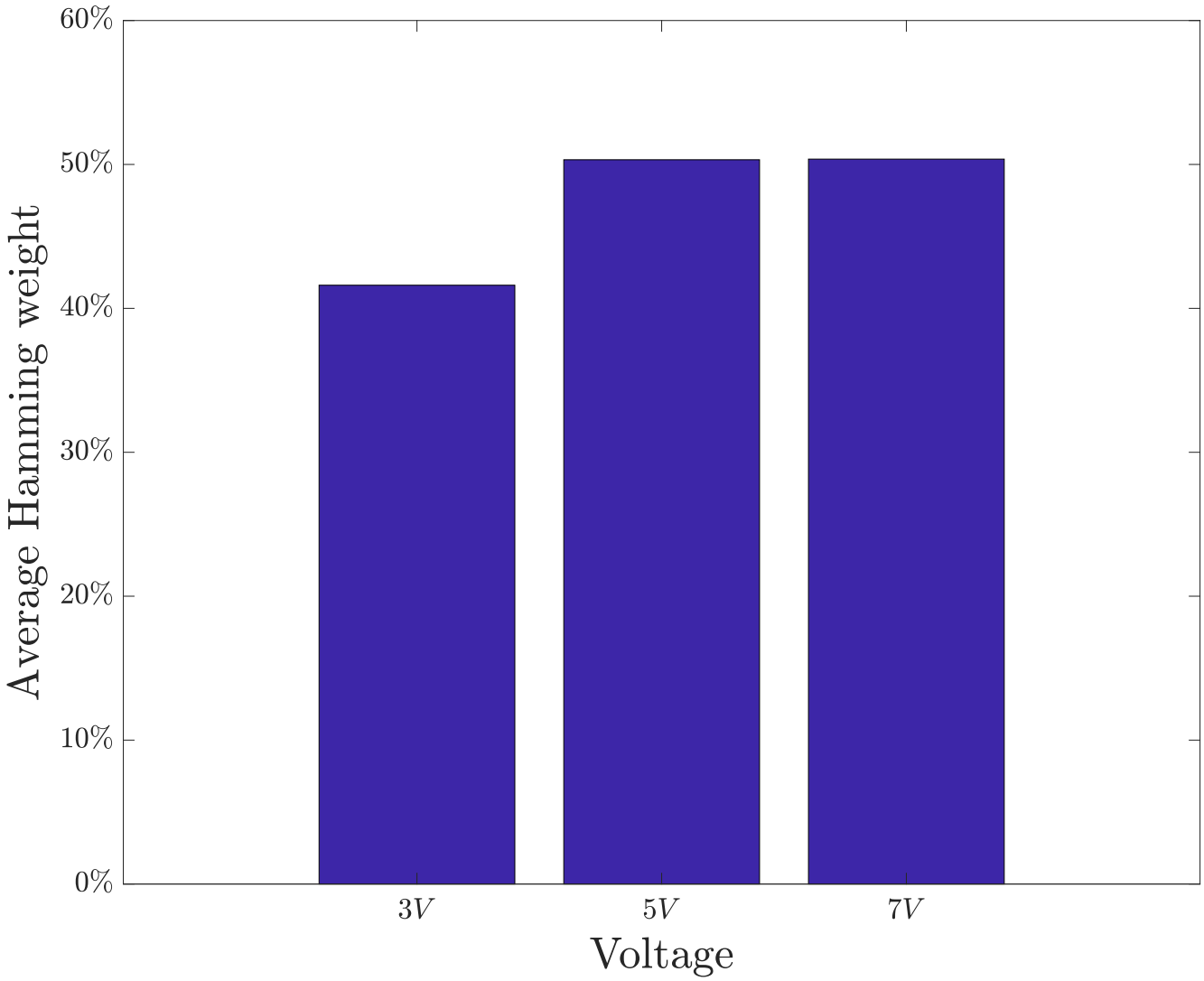


Figure 4.45: Overall average fractional Hamming weight value of the examined NAND-Flash-memory-based PUFs that utilises programming disturbances, for each value of supply voltage provided by the USB port of the STM32F429I Discovery (STM32F429I-DISC1) board to the overall system.

In order to further examine the effects of such supply voltage variations on the relevant NAND-Flash-memory-based PUF, we have also computed the fractional intra-device [49] Hamming distance for all the measurements originating from the same Flash memory page, for all the pages being utilised as PUF instances, first for each pair of PUF responses taken at the same supply voltage level, as shown in Figure 4.46, and, then, for each pair of PUF responses taken at different supply voltage levels, as shown in Figure 4.47. In this way, we can know how different are the PUF responses measured from the same PUF instance at different supply voltage conditions and, in this way, measure their robustness. We note that Figure 4.46 shows that the responses of this PUF become more robust as the supply voltage increases, with the fractional intra-device Hamming distance values for pairs of PUF responses taken at 5V and 7V indicating a high degree of robustness, with almost all of the values being below 0.2, in a fashion similar to the values noted under nominal conditions in Section 4.1.3, and the ones for pairs

[49] Here, the term "device" is used to refer to an instance of the examined Flash-memory-based PUF, i.e., to each relevant page of this memory being utilised as a PUF.

of PUF responses taken at 3V indicating a lower degree of robustness, but with most relevant values again being below 0.2. This is also reflected in Figure 4.47, as the fractional intra-device Hamming distance values for pairs of PUF responses for which one response was measured at 5V and the other at 7V indicate a high degree of robustness, with most of the values being below 0.2, in a fashion similar to the values noted under nominal conditions in Section 4.1.3, while the fractional intra-device Hamming distance values for pairs of PUF responses for which one response was measured at 3V and the other at 5V, and for which one response was measured at 3V and the other at 7V, seem to concentrate between 0.1 and 0.4, with many outliers – in both cases – between 0.4 and 0.6, indicating a rather low degree of robustness, which one may not always be able to address through the employment of a (reverse) fuzzy extractor scheme.

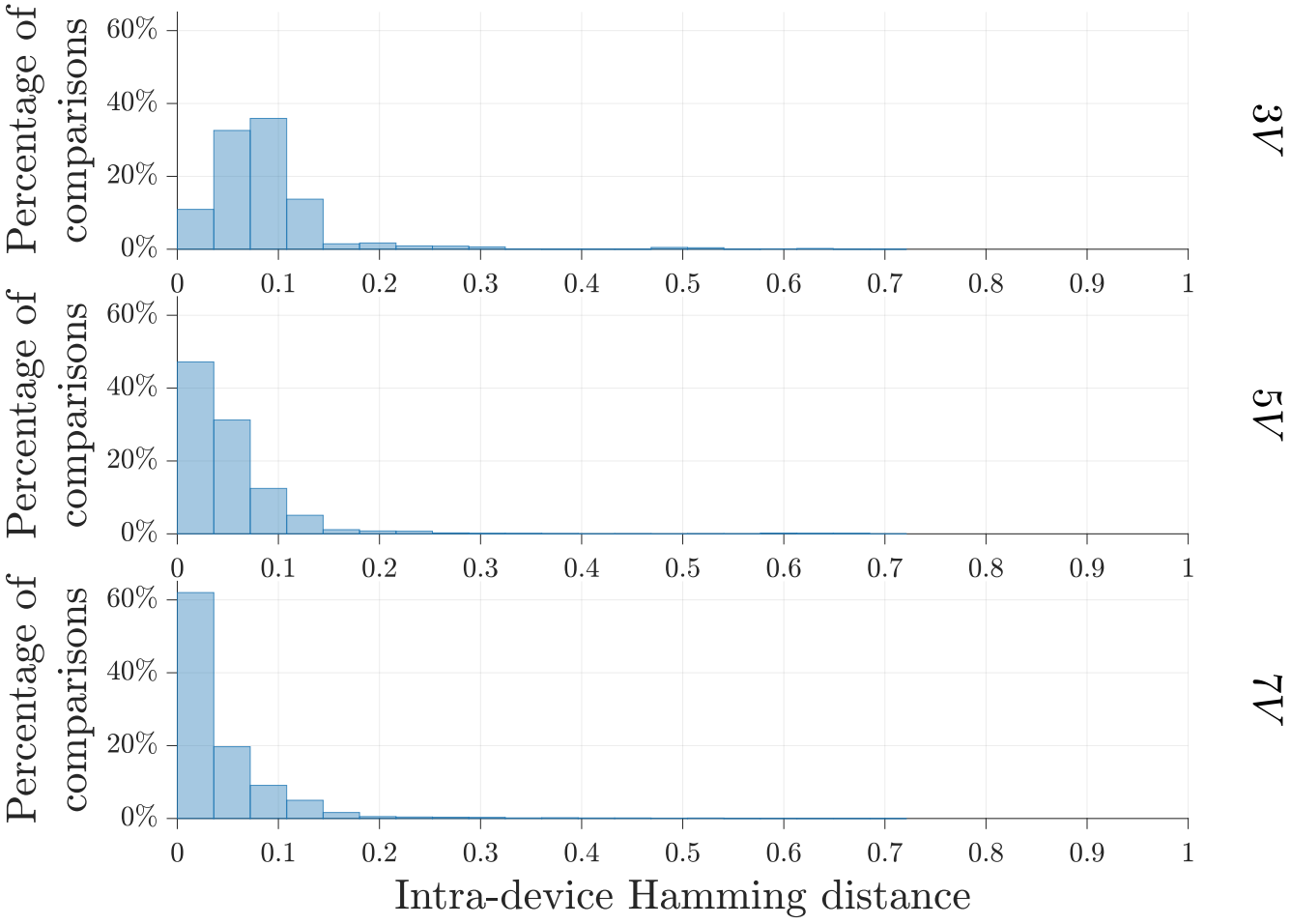


Figure 4.46: Intra-device Hamming distance values for the examined NAND-Flash-memory-based PUFs that utilises programming disturbances, for pairs of PUF responses taken at the same supply voltage level provided by the USB port of the STM32F429I Discovery (STM32F429I-DISC1) board to the overall system.

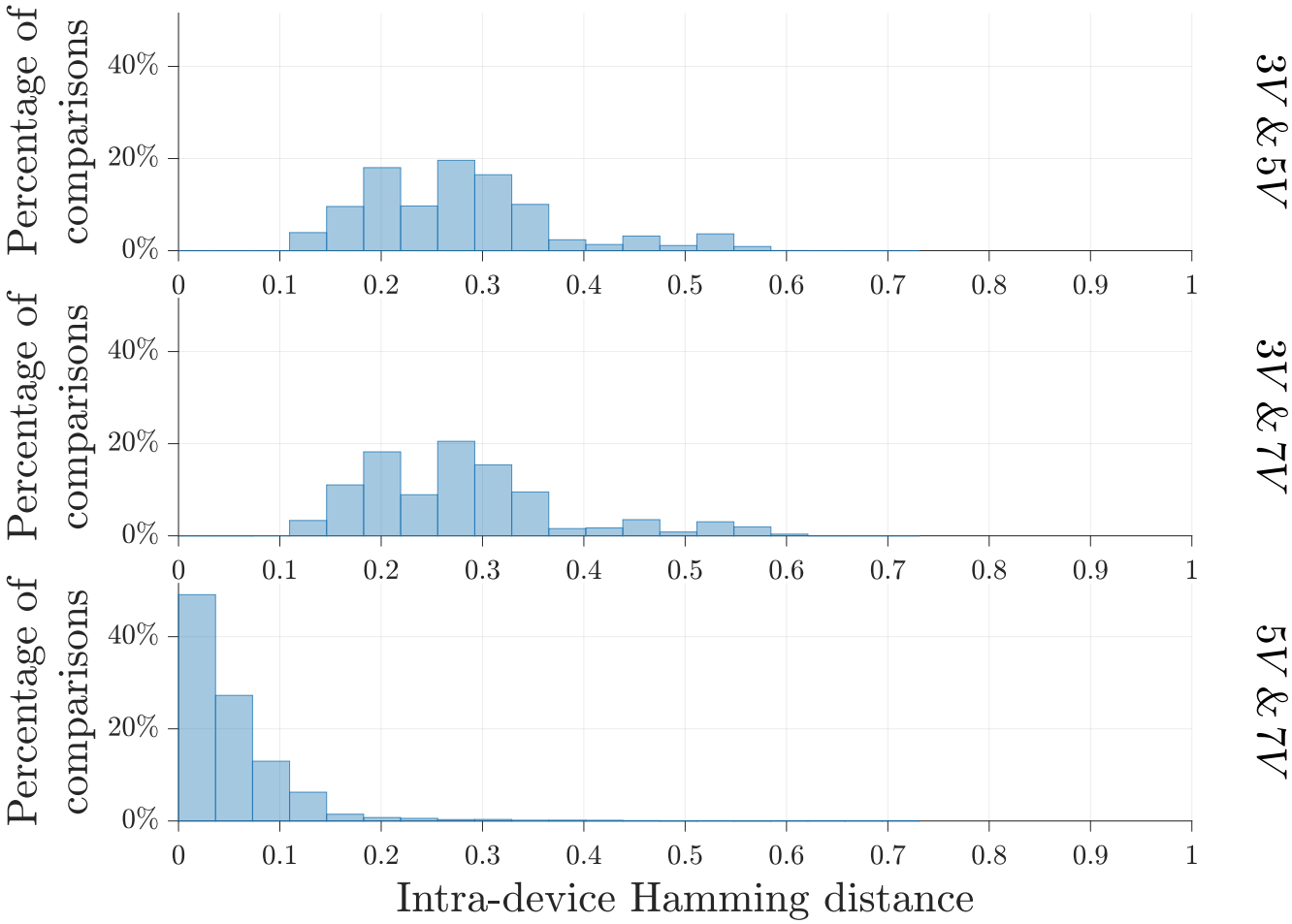


Figure 4.47: Intra-device Hamming distance values for the examined NAND-Flash-memory-based PUFs that utilises programming disturbances, for pairs of PUF responses taken at different supply voltage levels provided by the USB port of the STM32F429I Discovery (STM32F429I-DISC1) board to the overall system.

Nevertheless, we note that, even though supply voltage values that are lower than the nominal supply voltage level that should be supplied by the USB port of the STM32F429I Discovery (STM32F429I-DISC1) board to the overall system of devices seem to have a rather noticable effect on the responses of the examined NAND-Flash-memory-based PUF, this effect cannot be considered as able to preclude the employment of this PUFs as an adequate security mechanism in the context of practical IoT applications, as long as adequate care is taken to ameliorate this effect, e.g., by employing a fuzzy extractor scheme incorporating an ECC that can provide the appropriate level of error correction, or by utilising a bit selection scheme, such as the one proposed in [87]. In general, the effects observed do not seem to be so significant that an attack, other than a DoS attack, based on these effects would seem as probable.

Moreover, such effects can also be addressed by a dedicated security protocol that would provide a way to use the relevant NAND-Flash-memory-based PUF responses in a robust manner, e.g., by modifying the scheme proposed in Section 5.2, in order to address the effects of voltage instead of or in addition to those of temperature, in the case of a NAND-Flash-memory-based PUF instead of a DRAM-based PUF. Finally, we again believe that the exclusion of particular PUF instances from usage, or the consideration of multiple Flash memory pages as a single PUF instance, could provide a lower level of variance for

the fractional intra-device Hamming distance values of this PUF type under supply voltage variations to the power supply voltage provided by the USB port of the STM32F429I Discovery (STM32F429I-DISC1) board to the overall system of devices, and, thus, also a higher level of security in practice, although it would not completely address the relevant effects that supply voltage values that are lower than the nominal level cause on the relevant responses of the examined NAND-Flash-memory-based PUF.

Examining the Effects of Power Supply Voltage Variations on the NAND-Flash-Memory-Based PUFs Implemented on the Waveshare NandFlash Board (A) by Varying the Voltage Supplied by the 5V Pins of the STM32F429I Discovery (STM32F429I-DISC1) Board and the Waveshare Open429Z-D Standard Board

In order to examine how variations in the voltage supplied by the 5V pins of the STM32F429I Discovery (STM32F429I-DISC1) board and the Waveshare Open429Z-D Standard board may affect the quality characteristics of the NAND-Flash-memory-based PUFs implemented on the Waveshare Nand-Flash Board (A), we first examine the relevant average fractional Hamming weight values for each Flash memory constituting an instance of this PUF. In particular, as Figure 4.48 shows, these values are similar for all the supply voltage values tested, i.e., 3V, 5V, and 7V, for each page. This can also be confirmed by Figure 4.49, which demonstrates the overall average fractional Hamming weight value for each level of supply voltage used. The overall fractional Hamming weight value for the responses of the NAND-Flash-memory-based PUFs implemented on the Waveshare NandFlash Board (A) is almost exactly 0.5 for all the supply voltage values tested, i.e., 3V, 5V, and 7V. However, we also note that the fractional Hamming weight values for the different pages used as PUF instances are different from each other, in the same way that was noted in Section 4.1.3. Thus, we note that for all the power supply voltage values tested, the responses of particular PUF instances consistently correspond to a fractional Hamming weight value either significantly lower than the ideal value of 0.5, such as the responses corresponding to Flash memory pages '25', and '57', or significantly higher than the ideal value of 0.5, such as the responses corresponding to pages '27', and '43'.

In order to further examine the effects of such supply voltage variations on the relevant NAND-Flash-memory-based PUF, we have also computed the fractional intra-device [50] Hamming distance for all the measurements originating from the same Flash memory page, for all the pages being utilised as PUF instances, first for each pair of PUF responses taken at the same supply voltage level, as shown in Figure 4.50, and, then, for each pair of PUF responses taken at different supply voltage levels, as shown in Figure 4.51. In this way, we can know how different are the PUF responses measured from the same PUF instance at different supply voltage conditions and, in this way, measure their robustness.

[50] Here, the term "device" is used to refer to an instance of the examined Flash-memory-based PUF, i.e., to each relevant page of this memory being utilised as a PUF.

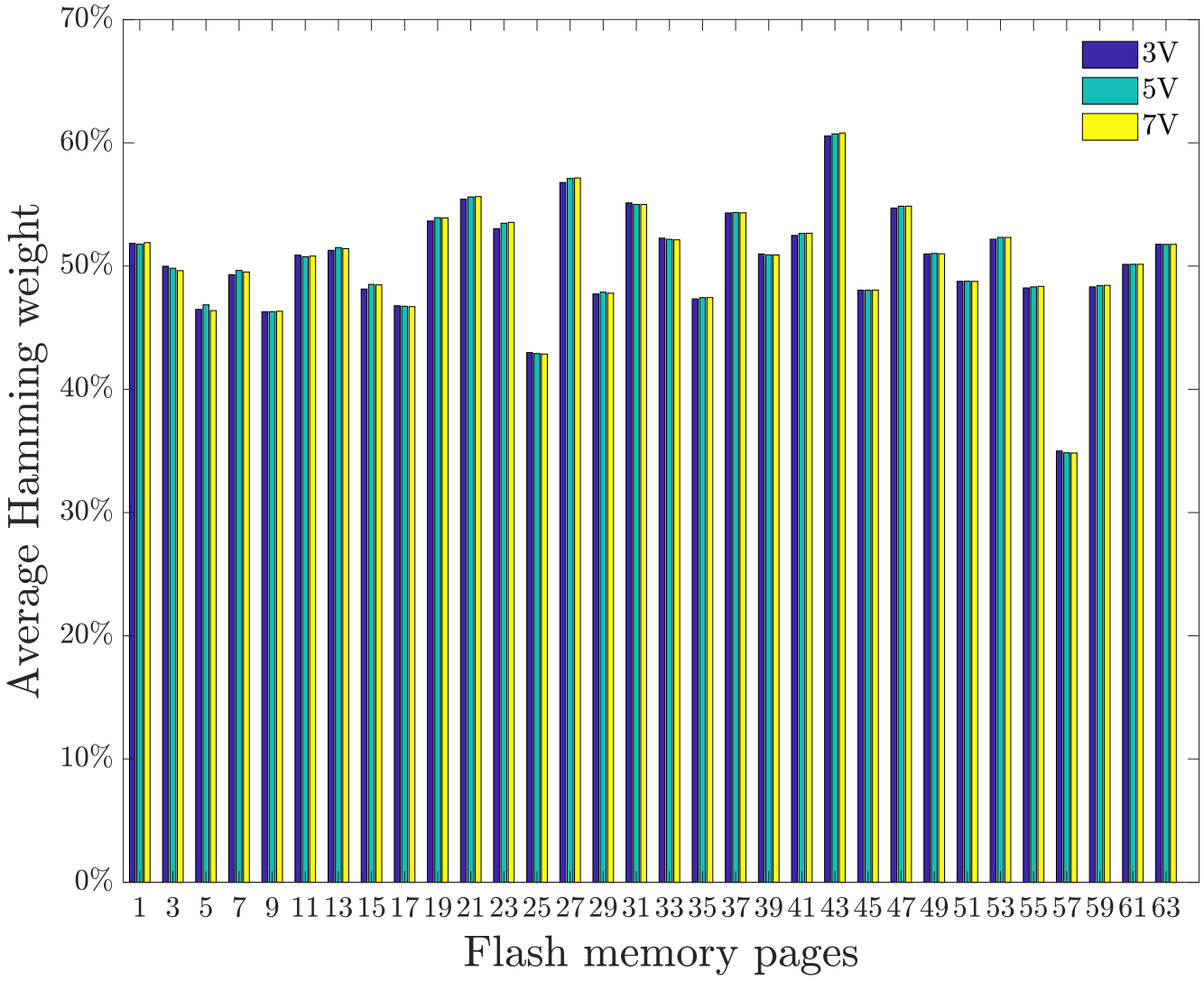


Figure 4.48: Average fractional Hamming weight values per instance of the examined NAND-Flash-memory-based PUFs that utilises programming disturbances, for the different values of power supply voltage provided by the 5V pins of the STM32F429I Discovery (STM32F429I-DISC1) board and the Waveshare Open429Z-D Standard board.

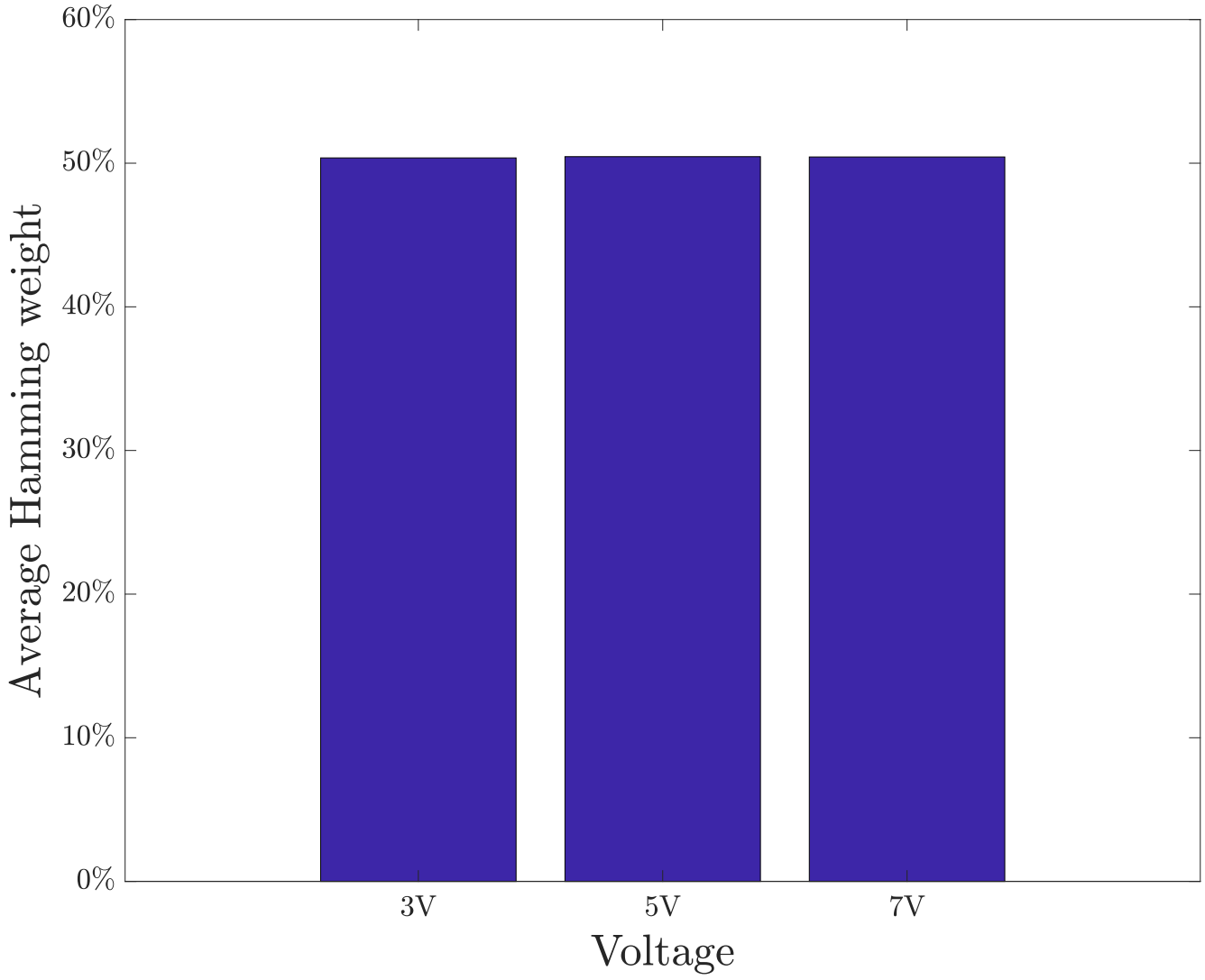


Figure 4.49: Overall average fractional Hamming weight value of the examined NAND-Flash-memory-based PUFs that utilises programming disturbances, for each value of supply voltage provided by the 5V pins of the STM32F429I Discovery (STM32F429I-DISC1) board and the Waveshare Open429Z-D Standard board.

We note that Figure 4.50 shows that the fractional intra-device Hamming distance values for pairs of PUF responses taken at all supply voltage values tested, i.e., 3V, 5V, and 7V, indicate the same degree of robustness as the one noted under nominal conditions in Section 4.1.3. This is also reflected in Figure 4.51, as the fractional intra-device Hamming distance values for pairs of PUF responses for which one response was measured at 5V and the other at 7V, for which one response was measured at 3V and the other at 5V, and for which one response was measured at 3V and the other at 7V, all seem to be similar to the fractional intra-device Hamming distance values of this PUF under nominal conditions, which were presented Section 4.1.3.

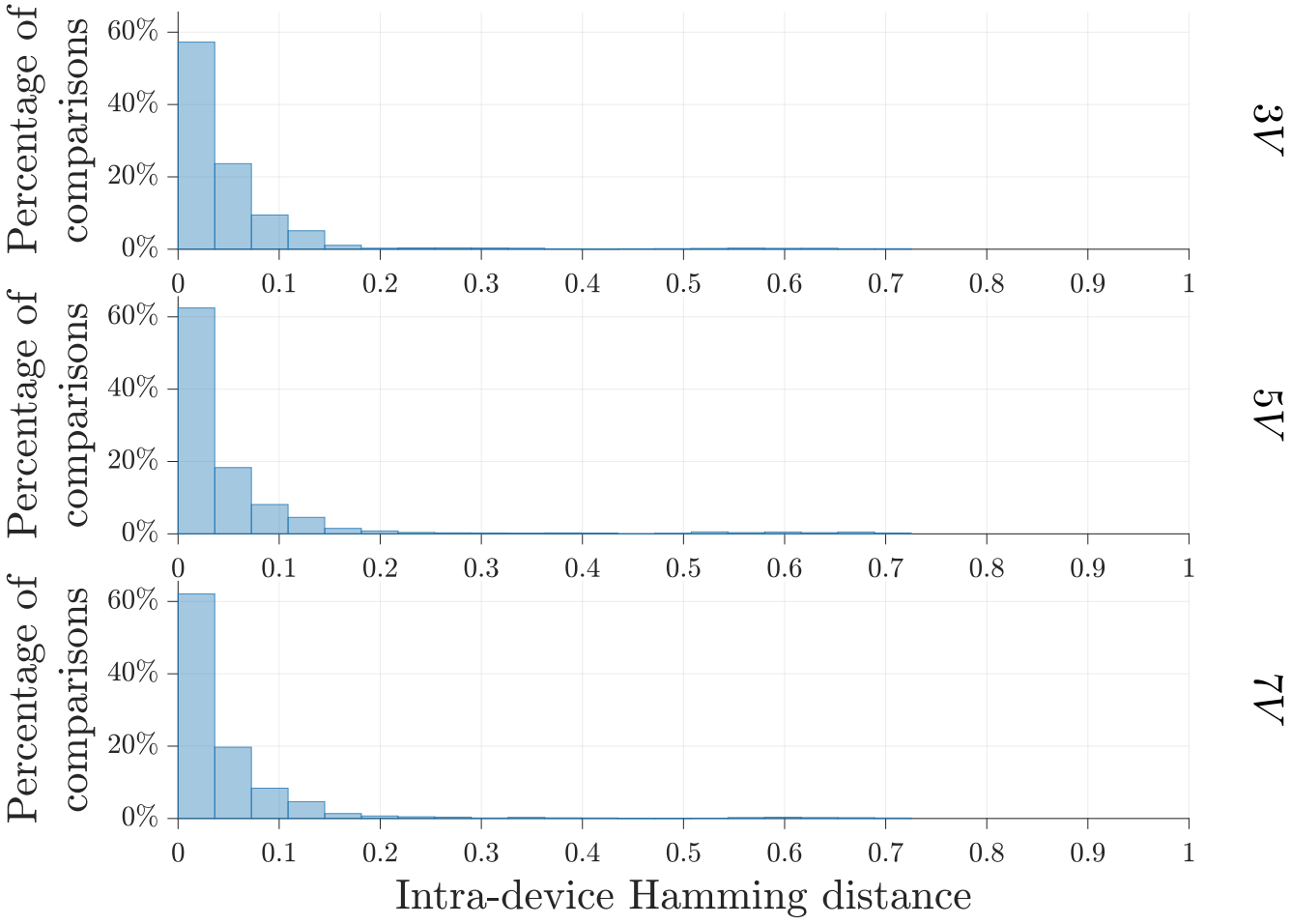


Figure 4.50: Intra-device Hamming distance values for the examined NAND-Flash-memory-based PUFs that utilises programming disturbances, for pairs of PUF responses taken at the same supply voltage level provided by the 5V pins of the STM32F429I Discovery (STM32F429I-DISC1) board and the Waveshare Open429Z-D Standard board.

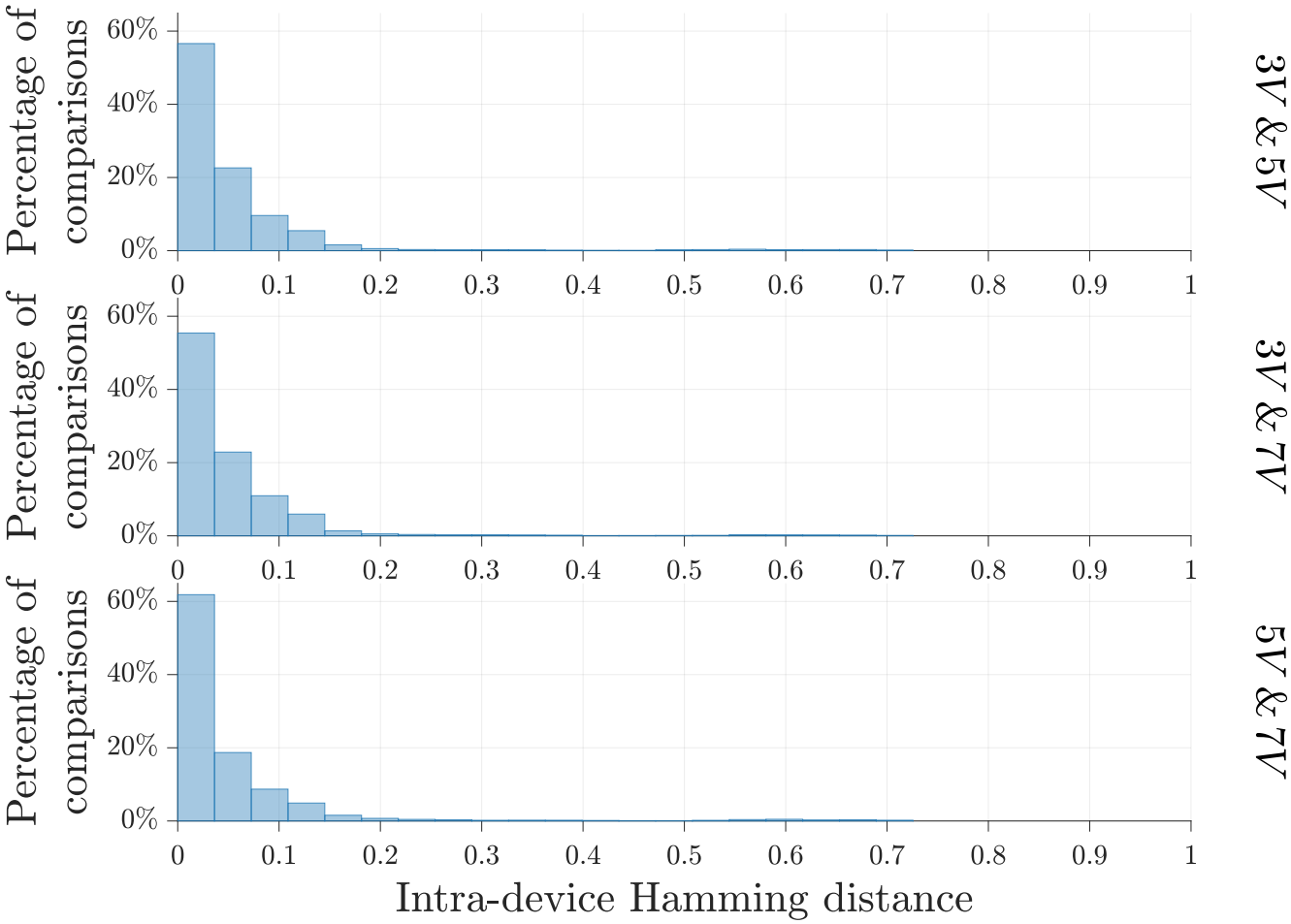


Figure 4.51: Intra-device Hamming distance values for the examined NAND-Flash-memory-based PUFs that utilises programming disturbances, for pairs of PUF responses taken at different supply voltage levels provided by the 5V pins of the STM32F429I Discovery (STM32F429I-DISC1) board and the Waveshare Open429Z-D Standard board.

Therefore, we note that supply voltage variations by varying the voltage supplied by the 5V pins of the STM32F429I Discovery (STM32F429I-DISC1) board and the Waveshare Open429Z-D Standard board do not seem to have an effect on the responses of the examined NAND-Flash-memory-based PUF [51]. Thus, the examined NAND-Flash-memory-based PUF can provide an *acceptable* level of security under such voltage variations. Nevertheless, of course, we note, once again, that the exclusion of particular PUF instances from usage, or the consideration of multiple Flash memory pages as a single PUF instance, could provide a lower level of variance for the fractional intra-device Hamming distance values of this PUF type under supply voltage variations to the power supply voltage provided by the 5V pins of the STM32F429I Discovery (STM32F429I-DISC1) board and the Waveshare Open429Z-D Standard board, and, thus, also a higher level of security in practice.

[51] This could potentially be attributed to the fact that the Waveshare NandFlash Board (A) is powered with 3.3V, which seem to be coming from the relevant 3.3V pins of the STM32F429I Discovery (STM32F429I-DISC1) board and the Waveshare Open429Z-D Standard board.

Examining the Effects of Power Supply Voltage Variations on the NAND-Flash-Memory-Based PUFs Implemented on the Waveshare NandFlash Board (A) by Varying the Voltage Supplied by the 3.3V Pins of the STM32F429I Discovery (STM32F429I-DISC1) Board and the Waveshare Open429Z-D Standard Board

In order to examine how variations in the voltage supplied by the 3.3V pins of the STM32F429I Discovery (STM32F429I-DISC1) board and the Waveshare Open429Z-D Standard board affect the quality characteristics of the NAND-Flash-memory-based PUFs implemented on the Waveshare NandFlash Board (A), we first examine the relevant average fractional Hamming weight values for each Flash memory constituting an instance of this PUF. In particular, as Figure 4.52 shows, these values are similar for all the supply voltage values tested, i.e., 2.3V, 3.3V, and 4.3V, for each page. This can also be confirmed by Figure 4.53, which demonstrates the overall average fractional Hamming weight value for each level of supply voltage used. The overall Hamming weight value for the responses of the

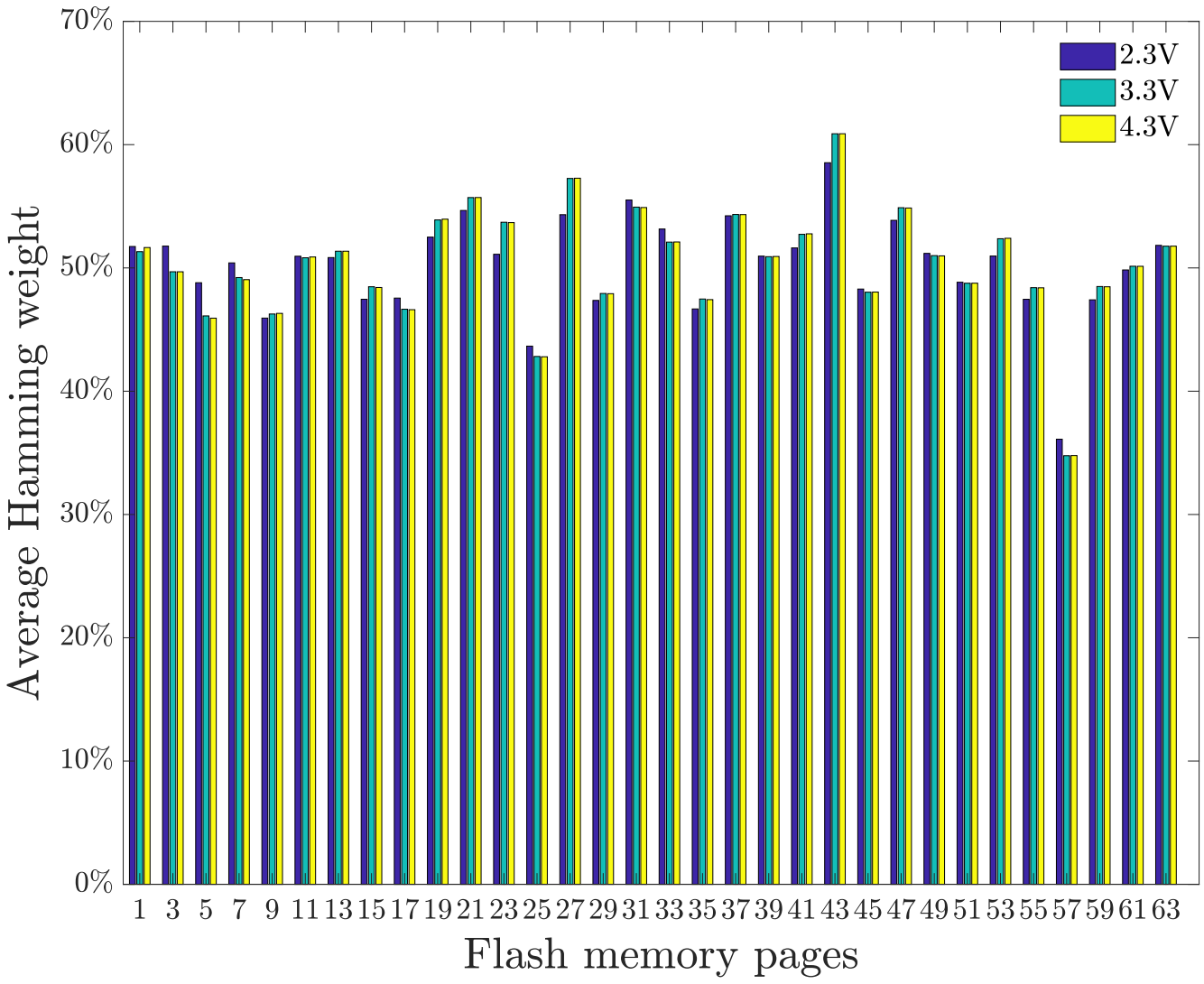


Figure 4.52: Average fractional Hamming weight values per instance of the examined NAND-Flash-memory-based PUFs that utilises programming disturbances, for the different values of power supply voltage provided by the 3.3V pins of the STM32F429I Discovery (STM32F429I-DISC1) board and the Waveshare Open429Z-D Standard board.

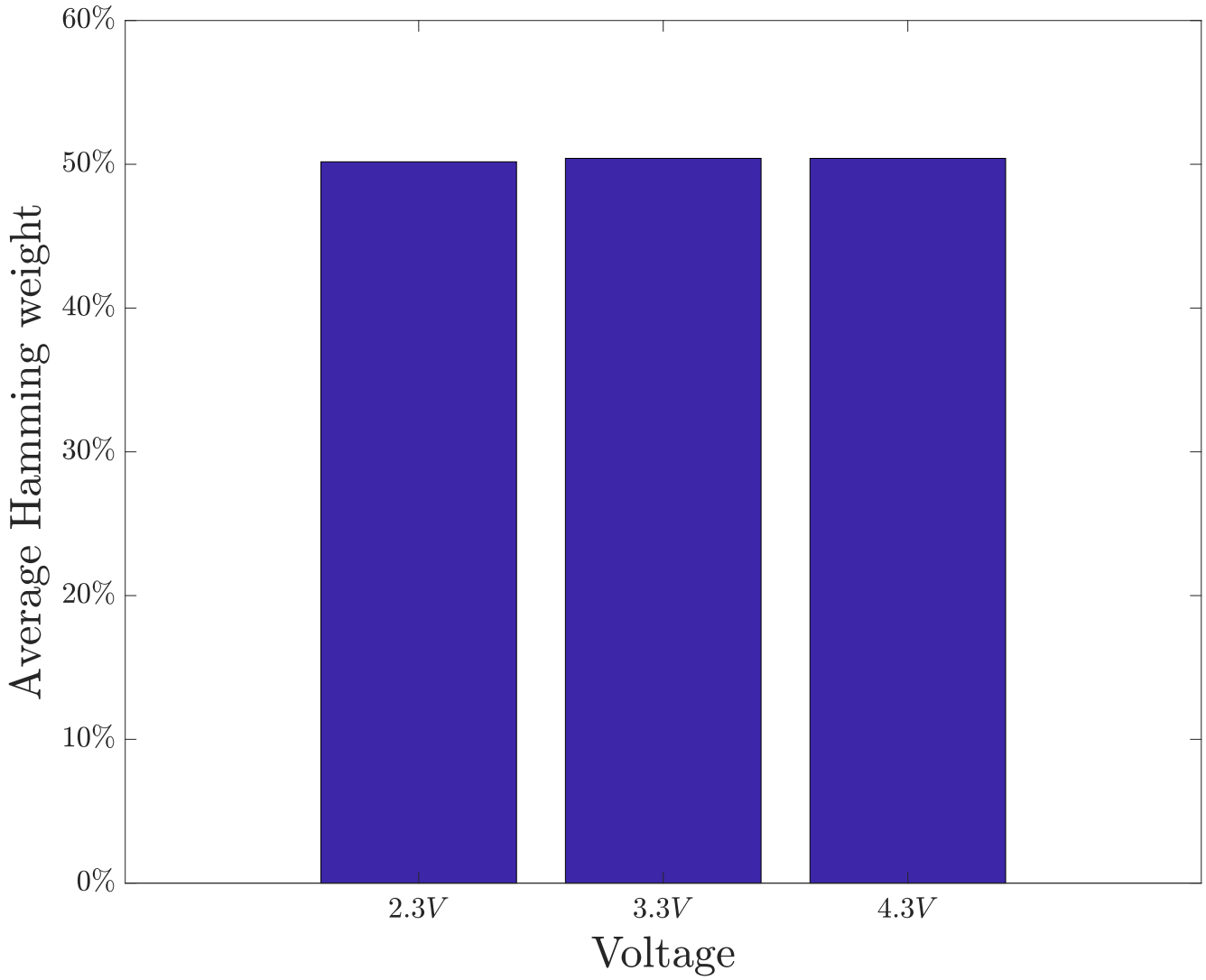


Figure 4.53: Overall average fractional Hamming weight value of the examined NAND-Flash-memory-based PUFs that utilises programming disturbances, for each value of supply voltage provided by the 3.3V pins of the STM32F429I Discovery (STM32F429I-DISC1) board and the Waveshare Open429Z-D Standard board.

NAND-Flash-memory-based PUFs implemented on the Waveshare NandFlash Board (A) is almost exactly 0.5 for all the supply voltage values tested, i.e., 2.3V, 3.3V, and 4.3V. However, we note that the fractional Hamming weight values for the different pages used as PUF instances are different from each other, in the same way that was noted in Section 4.1.3, with the fractional Hamming weight values for a supply voltage value of 2.3V exhibiting a higher degree of variance. In general, we note that for all the power supply voltage values tested, the responses of particular PUF instances consistently correspond to a fractional Hamming weight value either significantly lower than the ideal value of 0.5, such as the responses corresponding to Flash memory pages '25', and '57', or significantly higher than the ideal value of 0.5, such as the responses corresponding to pages '27', and '43'.

In order to further examine the effects of such supply voltage variations on the relevant NAND-Flash-memory-based PUF, we have also computed the fractional intra-device [52] Hamming distance for all the measurements originating from the same Flash memory page, for all the pages being utilised as PUF instances, first for each pair of PUF responses taken at the same supply voltage level, as shown in Figure 4.54, and, then, for each pair of PUF responses taken at different supply voltage levels, as shown in Figure 4.55. In this way, we can know how different are the PUF responses measured from the same PUF instance at different supply voltage conditions and, in this way, measure their robustness. We note that Figure 4.54 shows that the responses of this PUF become more robust as the supply voltage increases, with the fractional intra-device Hamming distance values for pairs of PUF responses taken at 3.3V and 4.3V indicating the same degree of robustness as the one noted under nominal conditions in Section 4.1.3, and the ones for pairs of PUF responses taken at 2.3V indicating a lower degree of robustness, but with most relevant values again being below 0.15.

This is also reflected in Figure 4.55, as the fractional intra-device Hamming distance values for pairs of PUF responses for which one response was measured at 3.3V and the other at 4.3V are similar

[52] Here, the term "device" is used to refer to an instance of the examined Flash-memory-based PUF, i.e., to each relevant page of this memory being utilised as a PUF.

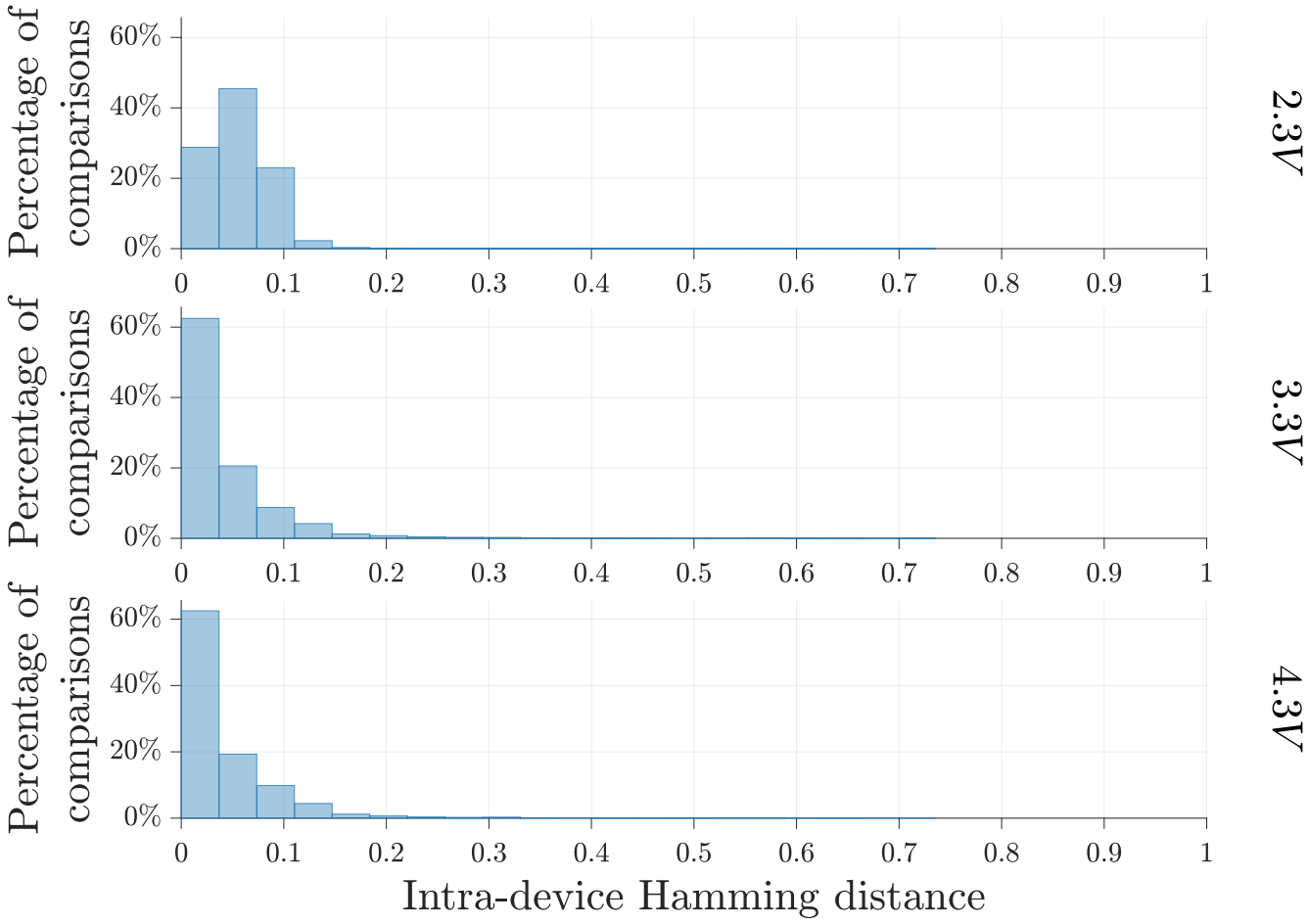


Figure 4.54: Intra-device Hamming distance values for the examined NAND-Flash-memory-based PUFs that utilises programming disturbances, for pairs of PUF responses taken at the same supply voltage level provided by the 3.3V pins of the STM32F429I Discovery (STM32F429I-DISC1) board and the Waveshare Open429Z-D Standard board.

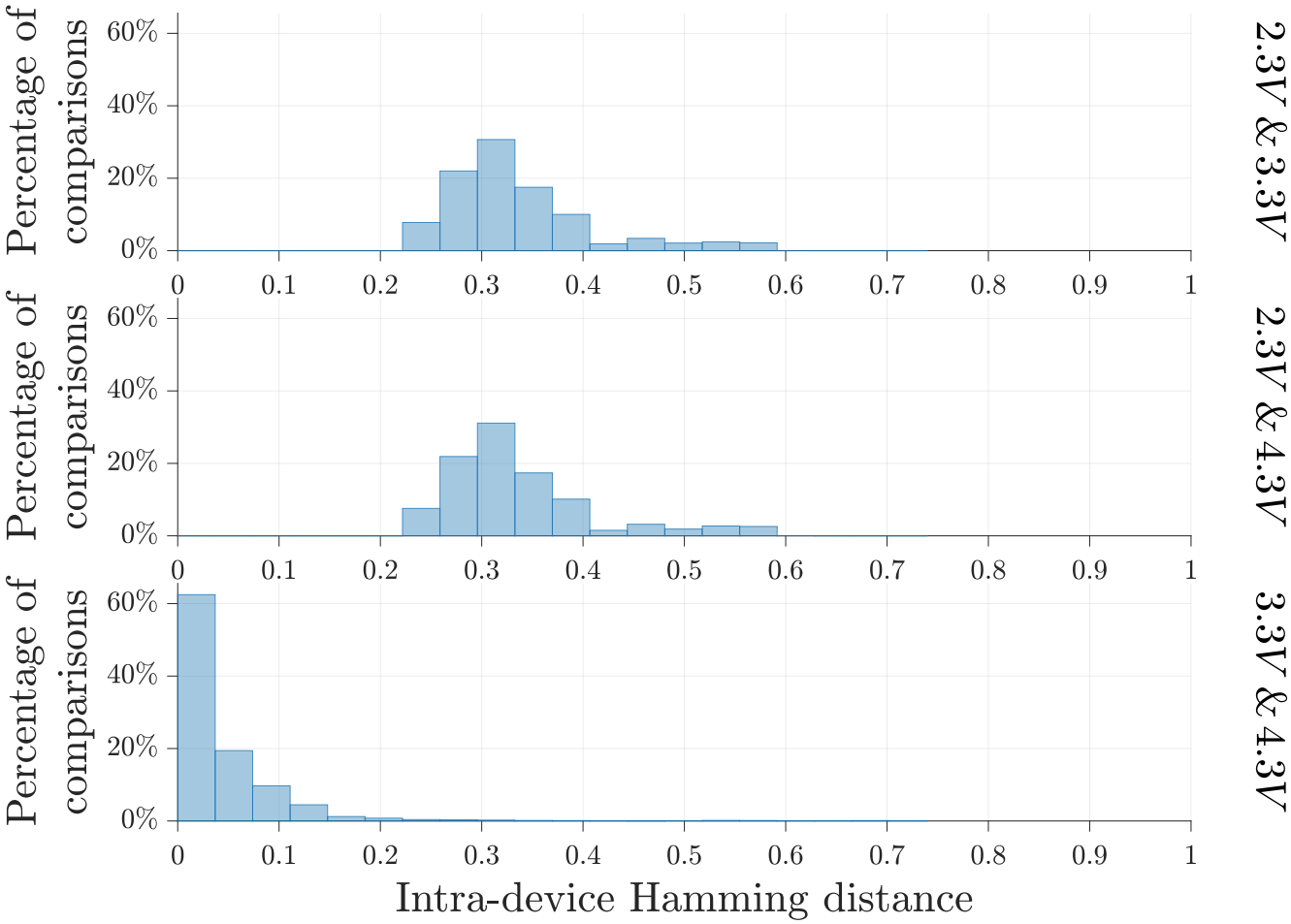


Figure 4.55: Intra-device Hamming distance values for the examined NAND-Flash-memory-based PUFs that utilises programming disturbances, for pairs of PUF responses taken at different supply voltage levels provided by the 3.3V pins of the STM32F429I Discovery (STM32F429I-DISC1) board and the Waveshare Open429Z-D Standard board.

to the fractional intra-device Hamming distance values of this PUF under nominal conditions, which were presented Section 4.1.3, while the fractional intra-device Hamming distance values for pairs of PUF responses for which one response was measured at 2.3V and the other at 3.3V, and for which one response was measured at 2.3V and the other at 4.3V, seem to concentrate between 0.2 and 0.4, with many outliers – in both cases – between 0.4 and 0.6, indicating a rather low degree of robustness, which one may not always be able to address through the employment of a (reverse) fuzzy extractor scheme.

Nevertheless, we note that, even though supply voltage values that are lower than the nominal supply voltage level that should be supplied by the 3.3V pins of the STM32F429I Discovery (STM32F429I-DISC1) board and the Waveshare Open429Z-D Standard board, seem to have a noticable effect on the responses of the examined NAND-Flash-memory-based PUF, this effect cannot be considered as able to preclude the employment of this PUFs as an adequate security mechanism in the context of practical IoT applications, as long as adequate care is taken to ameliorate this effect, e.g., by employing a fuzzy extractor scheme incorporating an ECC that can provide the appropriate level of error correction, or by utilising a bit selection scheme, such as the one proposed in [87]. In general, the effects observed do not

seem to be so significant that an attack, other than a DoS attack, based on these effects would seem as probable.

Moreover, such effects can also be addressed by a dedicated security protocol that would provide a way to use the relevant NAND-Flash-memory-based PUF responses in a robust manner, e.g., by modifying the scheme proposed in Section 5.2, in order to address the effects of voltage instead of or in addition to those of temperature, in the case of a NAND-Flash-memory-based PUF instead of a DRAM-based PUF. Finally, we again believe that the exclusion of particular PUF instances from usage, or the consideration of multiple Flash memory pages as a single PUF instance, could provide a lower level of variance for the fractional intra-device Hamming distance values of this PUF type under supply voltage variations to the 3.3V pins of the STM32F429I Discovery (STM32F429I-DISC1) board and the Waveshare Open429Z-D Standard board, and, thus, also a higher level of security in practice, although it may not be possible to fully address the relevant effects that supply voltage values that are lower than the nominal level cause on the relevant responses of the examined NAND-Flash-memory-based PUF.

4.4 On the Usage of Memory-Based PUFs Under Ionising Radiation

We note that most, if not all, COTS memory modules seem to be significantly resilient to radiation. In particular, as a number of tests that we have performed and the relevant literature indicate, the amount of radiation encountered in near-Earth space does not appear to be able to cause a significant number of errors in contemporary SRAM, DRAM and Flash memory modules. For example, it has been noted that modern COTS Flash memories remain unaffected by radiation as high as 100 gray (10 krad) [220], or even 1000 or 2000 gray (100 krad or 200 krad, respectively) [221], depending on the type of radiation and the scale of integration. In general, components of a larger scale of integration have been found to be more resilient to radiation [220]. Moreover, although DRAM may be susceptible to radiation of 20 gray (2 krad) when its values are not being refreshed [222], most modern DRAMs are refreshed every few milliseconds. Notably, radiation-induced errors, such as bit "flips", would, most probably, serve to actually increase the entropy and, thus, the randomness of the responses of DRAM PUFs that are based on the data decay/retention characteristics of DRAM cells, such as DRAM decay-based PUFs and DRAM Row Hammer PUFs. However, as the positions of such errors are rather random, they would also decrease the robustness of the responses of the relevant PUFs.

In order to test the resilience of memory-based PUFs to radiation, we have performed a number of relevant tests. For this purpose, we have implemented an SRAM and a NAND-Flash-memory-based PUF that utilises programming disturbances on the same ST Microelectronics STM32F407 Discovery (STM32F407G-DISC1) board, and the same two PUF types also on an ST Microelectronics STM32 Nucleo-64 NUCLEO-L152RE board, and tested how low levels of ionising radiation affect their operation. We note that both of these boards can be used in the implementation of various segments of the IoT, and have been used in relevant space applications by the Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR) [147, 223, 224, 225], i.e., the German Aerospace Center. We have used a caesium-137 ($^{137}_{55}$Cs) gamma (and beta) source, a strontium-90 ($^{90}_{38}$Sr) beta source, as well as a hard X-ray radiotherapy source [53], as shown in Table 4.6, in order to irradiate the SRAM and Flash PUFs that were

[53] A LINear ACcelerator (LINAC) was used, which produces X-rays through the rapid deceleration of electrons in a target material, most often a tungsten ($_{74}$W) alloy, that produces "braking radiation" (Bremsstrahlung). This radiation is, then, collimated, in order to achieve the best result, both for radiation therapy and for our experiment.

implemented on each of the two evaluation boards. All experiments were performed under nominal ambient temperature and voltage conditions.

Table 4.6: Radioactive Sources Being Used to Test the Resilience of SRAM and Flash-Memory-Based PUFs to Radiation

Radioactive Source	Emissions	Time Tested	Total Dose Absorbed
$^{137}_{55}$Cs	γ & β^-	10 days	0.024 Gy (2.4 rad)
$^{90}_{38}$Sr	β^-	100 days	105.6 Gy (10.56 krad) [†] 432 Gy (43.2 krad) [‡]
Hard X-ray source	10 MV X-rays	12 minutes	250 Gy (25 krad)

[†] For the ST Microelectronics STM32F407 Discovery (STM32F407G-DISC1) board. The placement of the boards led to different doses being absorbed by each board.

[‡] For the ST Microelectronics STM32 Nucleo-64 NUCLEO-L152RE board. The placement of the boards led to different doses being absorbed by each board.

We do note that none of our experiments resulted in any bit "flips", which would indicate errors. Furthermore, additional simulations for higher doses of radiation from these sources indicated very little interaction with the material of the chips and, therefore, we do not expect that such radiation can have a significant effect on the implemented memory-based PUFs. Our results indicate that relatively high doses of beta radiation or hard X-rays, or low doses of gamma radiation, cannot really affect the operation of the examined intrinsic memory-based PUFs on IoT devices. We note that, although these modules were not decapsulated, the radioactive sources were placed mere centimeters from the bare devices, and no effects were observed. We can, therefore, conclude that, as low levels of ionising radiation do not affect the operation of memory-based PUFs, the potential presence of ionising radiation does not seem to preclude the employment of memory-based PUFs as adequate security mechanisms in the context of practical IoT applications.

Regarding radiation attacks, we need to note that such attacks cannot be considered as low-cost and, thus, as practical, unless they are performed using a radioactive source that, like the sources used in our experiments, is easy to obtain, and they do not require the decapsulation, delayering, and/or removal of the passivation of the device, as such actions may, sooner or later, be noticed. In particular, we observe that unlike potential temperature attacks, which can be easily performed *in situ* using either a cooling spray or a hot air gun, the most effective radiation attacks would rather require the use of advanced equipment, such as a proton, or another particle, e.g., an ion, beam stemming from a particle accelerator, as well as the decapsulation, passivation removal, and/or delayering of the device. In this case, we observe that after the decapsulation, passivation removal, and/or delayering of the device has been performed, it would be rather easier and more cost-effective to perform other types of fault injection attacks against the device, such as optical fault injection ones [179], as these would provide a higher probability of success and rather more focused results, as it would be easier, in this case, to limit the effects of such attacks to a particular area of the relevant device, than if, for example, an ion beam would be used. In general, we can conclude that practical attacks against memory-based PUF utilised in IoT applications appear to be rather unfeasible.

5 Practical PUF-Based Security Solutions for IoT Applications

Partially Based Upon Peer-Reviewed Content Published In

- W. Xiong, A. Schaller, N. A. Anagnostopoulos, M. U. Saleem, S. Gabmeyer, S. Katzenbeisser & J. Szefer, "Run-Time Accessible DRAM PUFs in Commodity Devices", Proceedings of the 18th International Conference on Cryptographic Hardware and Embedded Systems (CHES 2016), vol. 9813 of "Lecture Notes in Computer Science (LNCS)", pp. 432-453, Springer, 2016. DOI: 10.1007/978-3-662-53140-2_21
- A. Schaller, W. Xiong, N. A. Anagnostopoulos, M. U. Saleem, S. Gabmeyer, S. Katzenbeisser & J. Szefer, "Intrinsic Rowhammer PUFs: Leveraging the Rowhammer Effect for Improved Security", Proceedings of the 2017 IEEE International Symposium on Hardware Oriented Security and Trust (HOST 2017), IEEE, 2017. DOI: 10.1109/HST.2017.7951729
- N. A. Anagnostopoulos, S. Katzenbeisser, J. Chandy & F. Tehranipoor, "An Overview of DRAM-Based Security Primitives", Cryptography, vol. 2, iss. 2, MDPI, 2018. DOI: 10.3390/cryptography2020007
- N. A. Anagnostopoulos, T. Arul, Y. Fan, J. Lotichius, C. Hatzfeld, F. Fernandes, R. Sharma, F. Tehranipoor & S. Katzenbeisser, "Securing IoT Devices Using Robust DRAM PUFs", Proceedings of the 2018 Global Information Infrastructure and Networking Symposium (GIIS 2018), IEEE, 2018. DOI: 10.1109/GIIS.2018.8635789
- A. Schaller, W. Xiong, N. A. Anagnostopoulos, M. U. Saleem, S. Gabmeyer, B. Škorić, S. Katzenbeisser & J. Szefer, "Decay-Based DRAM PUFs in Commodity Devices", IEEE Transactions on Dependable and Secure Computing (TDSC), vol. 16, iss. 3, pp. 462-475, IEEE, 2018. DOI: 10.1109/TDSC.2018.2822298
- N. A. Anagnostopoulos, T. Arul, Y. Fan, M. Kumar & S. Katzenbeisser, "AR-PUFs: Advanced Security Primitives for the Internet of Things and Cyber-Physical Systems", Proceedings of the 2019 IEEE International Conference on Consumer Electronics (ICCE 2019), IEEE, 2019. DOI: 10.1109/ICCE.2019.8661840
- W. Xiong, N. A. Anagnostopoulos, A. Schaller, S. Katzenbeisser & J. Szefer, "Spying on Temperature Using DRAM", Proceedings of the 22nd Design, Automation & Test in Europe Conference & Exhibition (DATE 2019), pp. 13-18, IEEE, 2019. DOI: 10.23919/DATE.2019.8714882
- N. A. Anagnostopoulos, Y. Fan, T. Arul, R. Sarangdhar & S. Katzenbeisser, "Lightweight Security Solutions for IoT Implementations in Space", Proceedings of the 2019 IEEE Topical Workshop on Internet of Space (TWIOS 2019), IEEE, 2019. DOI: 10.1109/TWIOS.2019.8771257
- L. Negka, G. Gketsios, N. A. Anagnostopoulos, G. Spathoulas, A. Kakarountas & S. Katzenbeisser, "Employing Blockchain and Physical Unclonable Functions for Counterfeit IoT Devices Detection", Proceedings of the 1st International Conference on Omni-Layer Intelligent Systems (COINS 2019), pp. 172-178, ACM, 2019. DOI: 10.1145/3312614.3312650
- N. A. Anagnostopoulos, S. Ahmad, T. Arul, D. Steinmetzer, M. Hollick & S. Katzenbeisser, "Low-Cost Security for Next-Generation IoT Networks", ACM Transactions on Internet Technology, vol. 20, iss. 3, art. 30, ACM, 2020. DOI: 10.1145/3406280

In this section, we present practical security protocols based on PUFs, and examine how they can be used in IoT applications, in order to secure the heterogeneous devices and diverse networks that form part of the IoT. In particular, we first examine how memory-based PUFs can be utilised in order to secure IoT devices, by allowing for key agreement, secure device identification, and authentication. Then, we examine a PUF-based security solution that would allow us to secure current and next-generation IoT devices and networks. Subsequently, we examine the potential of memory-based PUFs to serve as adequate security mechanisms in IoT applications that would expose them to adverse conditions. More specifically, we examine whether such PUFs can be utilised in the context of space applications. Finally, we examine Advanced Reconfigurable PUFs (AR-PUFs), i.e., a category of reconfigurable PUFs that allow for advanced security applications which can restore security after a successful attack, in a practical and lightweight manner. In this way, we aim to prove beyond reasonable doubt that memory-based PUFs constitute adequate security mechanisms for the IoT in practice.

5.1 Memory-Based PUFs for Key Agreement, Device Identification, and Authentication Protocols

One of the most well-known security applications of PUFs is secure key generation and storage. In general, if a key is generated by a PUF, it can be linked to it (and, therefore, also to the relevant device), as a PUF will always generate the same output when provided the same input under the same conditions, and, therefore, also the same key. Moreover, a key generated from a PUF does not need to remain stored after being used, as it can be easily regenerated every time it is needed. Therefore, PUFs have three main security applications, namely key agreement, identification and authentication. Nevertheless, of course, a large number of different security applications of memory-based PUFs have been examined in the relevant literature. For example, SRAM PUFs have been proposed in a number of works for the implementation of cryptographic solutions [11, 15, 22, 137, 146, 226], such as identification, authentication, attestation, secure boot, and secure key agreement protocols. Additionally, SRAM PUFs can also be used as a source of randomness [22, 137, 227], for the realisation of (true or pseudo)random number generators.

5.1.1 PUF-Based Secure Key Agreement

Regarding secure key agreement, usually, a (reverse) fuzzy extractor scheme [16, 138, 139] needs to be applied, as a PUF often provides slightly noisy responses, which could otherwise affect its reliability [134, 135]. This means that, for a particular challenge, the PUF may not always generate exactly the same response, but may instead generate rather similar responses, which may, however, contain some amount of noise. For this reason, we employ a fuzzy extractor scheme, which incorporates some ECC [76], in order to stabilise the PUF response [13]. PUF-based key agreement consists of two phases, which are shown in Figure 5.1.

In the first phase, the enrollment phase, which is executed entirely on the server's side, a PUF response (which corresponds to a particular random challenge), is combined with the desired key, in order to produce some redundancy bits, called helper data. The server then saves the challenge, the response and the key used, as well as the helper data produced, and an IDentity (ID) regarding the PUF, the CRP and the key used. As Figure 5.1 shows, the same CRP (and PUF) can be used with a different key, which will result in different helper data. If the fuzzy extractor scheme is constructed in a secure

way, such that the helper data do not leak information about the CRP and the key, then the helper data can also be made public.

In the second phase, the reconstruction phase, the PUF has been transferred securely from the server to the client and a reverse procedure is employed in order for the server and the client to agree on a key. The server sends a challenge to the client and the relevant helper data for a specific key. The client uses the PUF to produce the (noisy) response corresponding to the sent challenge, which is then combined with the helper data and corrected, using the same ECC as in the enrollment, in order to produce the key. As the server already has the relevant key from the enrollment phase, the server and the client have agreed to a key, without revealing or transferring a CRP or the key.

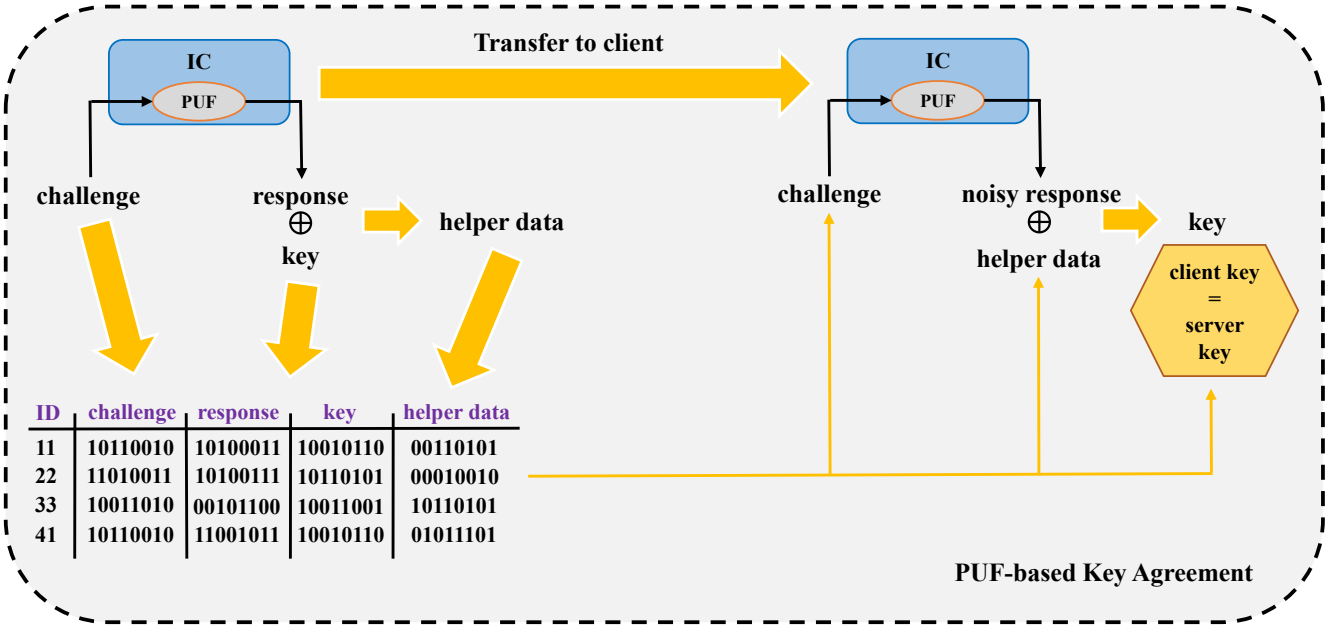


Figure 5.1: PUF-based key agreement.

Regarding the fuzzy extractor scheme and its related ECC, a number of schemes have been proposed for memory-based PUFs, including fuzzy extractor schemes using helper data [86, 163, 164, 165, 166, 167, 173], fuzzy match schemes with or without hashing [169, 170, 171] and fuzzy vault schemes that also use helper data [74]. Additionally, using a universal hash implementation based on the Toeplitz matrix [165, 167], which can be represented by a LFSR, is another way of stabilising memory-based PUF responses. All of these schemes need to be combined with an ECC, which can potentially be based on either Hamming [163, 172, 173], or BCH [164, 165, 167], or Reed–Solomon [74] error correction code.

However, instead of error correction, one can also employ the use of only the most stable cells [48, 79, 85, 87, 141, 174], i.e., the most stable bits of the original PUF response, in order to create a stable response. This approach is quite efficient, especially in applications that do not require perfect stability, such as identification and authentication. In this case, applying an acceptance threshold for stability can be sufficient, as then only the most unstable cells need to be excluded from the response, with no further operation required.

5.1.2 PUF-Based Identification

PUF-based identification works in a similar way to PUF-based key agreement, in the sense that a server queries multiple PUFs in order to create a database of (some of) their CRPs, which are stored according to the PUF from which they came. Then, at any point in time, this server can identify any of these PUFs, without explicitly revealing the stored CRP, by sending challenges to the relevant PUF and observing if the responses produced correspond to the relevant responses stored in its database. In this way, a PUF can be identified if CRPs from it exist on the server's database. Figure 5.2 demonstrates how PUF-based identification works. However, in the case of successful identification, a CRP of the identified PUF may have been completely revealed, and therefore should not be used again. In order to avoid this, a hashing scheme may be employed, so that the CRPs are not transmitted in the clear, or a key can be used for implicit identification, through encryption and decryption of a nonce, a signature, an identifier, etc. Finally, also error correction may again be required, if the PUF responses are noisy.

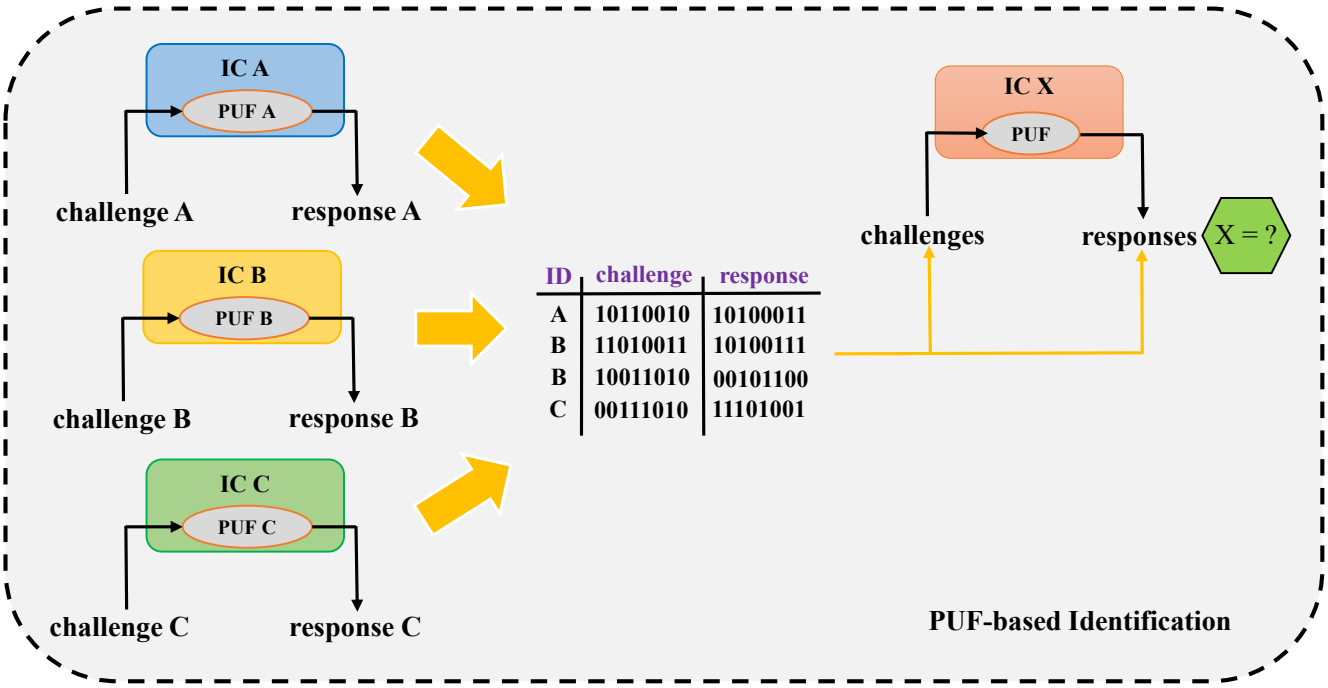


Figure 5.2: PUF-based identification.

5.1.3 PUF-Based Authentication

Finally, PUF-based authentication is also very similar to PUF-based identification, as once again a server creates a database of (some of) the CRPs of a PUF. However, in this case, it is assumed that the PUFs, and their related devices, go through untrusted environments, where they can be substituted by counterfeits. Therefore, if a client wants to ensure the authenticity of such a device, it can do so by sending a request for authentication of the relevant PUF to the server. Then, the server responds with a challenge for the PUF to be identified. If the produced response matches the relevant response saved on the server's database, then the device is authenticated, otherwise not, as shown in Figure 5.3. In this case, authentication is achieved without explicitly revealing any of the stored CRPs. Again, however, in the case of successful authentication, a CRP of the authenticated PUF may be completely revealed, and

therefore should not be used again. In order to avoid this, a hashing scheme may be employed, so that the CRPs are not transmitted in the clear, or a key can be used for implicit authentication, etc. Finally, error correction may also again be required, if the PUF responses are noisy.

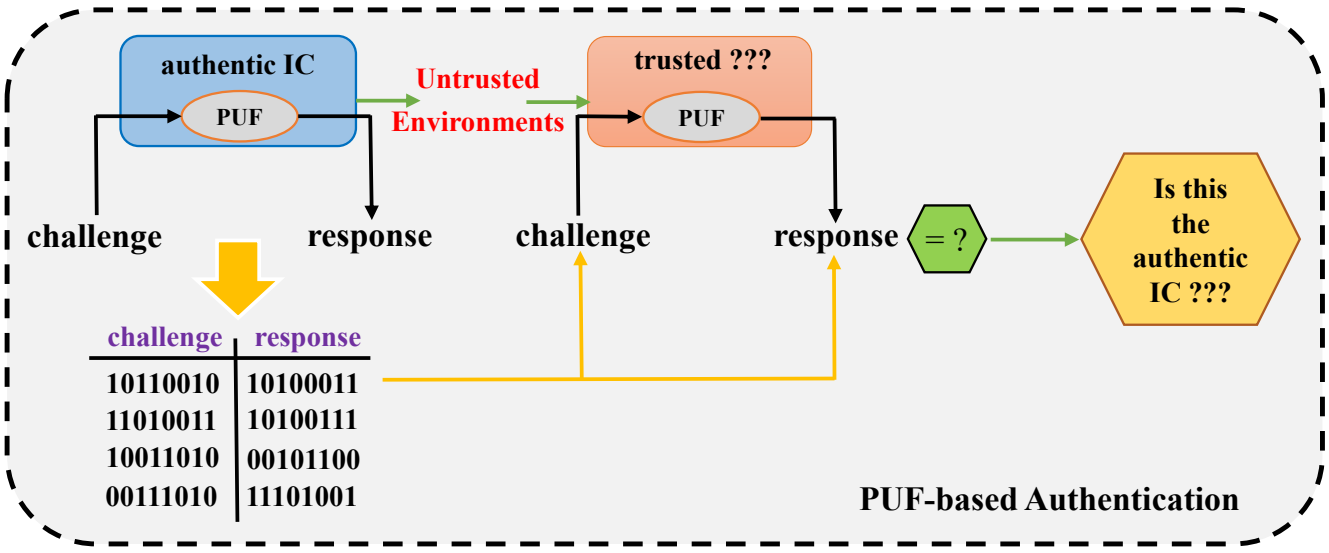


Figure 5.3: PUF-based authentication.

5.1.4 Concluding Remarks and More Advanced PUF-Based Security Protocols

In this subsection, we have described simple key agreement, device identification and authentication protocols for memory-based PUFs. In general, we note that authentication can serve anti-counterfeiting purposes, too, as validating the authenticity of a device containing a PUF can explicitly help to identify counterfeits. In a similar way, identification can also serve as de-anonymisation, as identifying a device containing a PUF will reveal its identity and, thus, de-anonymise it. As we have shown in previous sections, memory-based PUFs can provide a good level of security, as their responses remain highly random, robust, and unique, even under adverse conditions. Essentially, the evaluation of memory-based PUFs presented in Section 4 can guarantee that such PUFs can provide secure key generation and storage, as well as form the basis for the implementation of secure key agreement protocols. Additionally, as such PUFs can be lightweight, flexible, scalable, and cost-efficient, they should be considered as practical security mechanisms that can be used in all segments of the IoT, in the context of simple device authentication, identification, and any other security application that is based on secure key management and agreement.

This has been demonstrated in a number of relevant works, where we have examined and evaluated memory-based PUFs as security mechanisms for IoT applications. In particular, Xiong et al. [86] have proposed a device authentication protocol, as well as a secure channel establishment protocol (essentially a secure key agreement protocol), that are based on DRAM decay-based PUFs. Schaller et al. [87] have also presented a mutual authentication protocol based on the same PUFs. The ability of the Row Hammer PUF, which was introduced in [82], to be used for the implementation of these protocols has been discussed in [143]. Yue et al. [228] have discussed an authentication protocol that is essentially based on the classification of DRAM PUFs using a convolutional neural network. Moreover, a general

overview of protocols that utilise DRAM-based security solutions, including DRAM-based PUFs, has been presented in [229].

We have also proposed and examined a few more advanced security protocols based on memory-based PUFs in a number of works. For example, we have proposed that memory-based PUFs can be utilised for the implementation of an integrity-checking protocol [230] and an authentication protocol [231] in the context of smart, potentially autonomous, vehicles. Moreover, we have noted that the different components of current IoT devices, i.e., microcontrollers, main memories, and Input/Output (I/O) (i.e., communication) units, most often incorporate some memory, which may, for example, act as a small data buffer. We have, therefore, proposed that memory-based PUFs can be implemented on different components of IoT devices, in order to produce a unique token based on the combination of the responses of the PUFs implemented on these components, which would serve to identify and authenticate each IoT device, in order to provide advanced anti-counterfeiting protection [232]. In this way, it would be ensured that not only the device as a whole, but also each of its components, is authentic. Additionally, the produced device token can be stored in a dedicated blockchain, taking advantage of the concept of IoT 2.0, and making sure that such tokens are securely stored, and that the identity, and ownership, of such devices cannot be repudiated, or disputed.

In this work, we focus rather on more advanced security applications of memory-based PUFs, in order to prove their ability to provide state-of-the-art security applications for IoT devices and networks, even in adverse conditions. In particular, we propose and examine a rather generic protocol for robust security applications based on DRAM decay-based PUFs [233], taking into account the dependency of such PUFs on the ambient temperature, which was discussed in Section 4. Moreover, we also investigate a novel SRAM PUF implementation that can be utilised to secure both current and next-generation IoT devices and networks [145], and present a peripheral authentication protocol and a protocol for securing IEEE 802.11ad networks, which utilise this PUF as a security anchor. Additionally, we also examine whether memory-based PUFs can be utilised for the implementation of security protocols in adverse environments, in order to be used for example in space applications [147]. Finally, we also present and discuss a category of reconfigurable PUFs that allow for advanced security applications which can restore security after a successful attack, in a practical and lightweight manner [148].

5.2 Securing IoT Devices at Run-Time Using Robust DRAM PUF-Based Protocols

Cryptography based on DRAM decay-based PUFs (which are also known as DRAM retention-based PUFs) or on Row Hammer PUFs needs to take into account their dependency on temperature variations. Some ways to mitigate the effects of temperature on the responses of these PUFs have already been proposed in the relevant literature [86, 87, 143]. In particular, it has been noted that higher temperatures increase the number of bit "flips" in such a way that the number (and positions) of bit "flips" at a particular temperature T_1 for a particular time t are the same as the number (and positions) of the bit "flips" at a (potentially) different temperature T_2 for a different time, which is referred to as equivalent time, t_e [86, 87]. The relation between times t and t_e is described by the following formula:

$$t_e = \ t \times e^{-\alpha \times (T_2 - T_1)} \ , \tag{5.1}$$

where α is a constant different for each particular device [86, 87]. Additionally, it has been noted that the responses of a DRAM-based PUF could be combined with temperature readings, as modern DRAMs include internal temperature sensors, which are used to define the period of their refresh operation that helps retain the logical values written on their cells [229]. Finally, during PUF enrollment, one could also measure the PUF responses with very high granularity regarding its different operating temperatures, so as to produce a comprehensive mapping between the temperature variations and the PUF responses, for each particular PUF. Taking into account these approaches, we propose a proof-of-concept robust DRAM-based cryptographic protocol (for key agreement), which is illustrated in Figure 5.4.

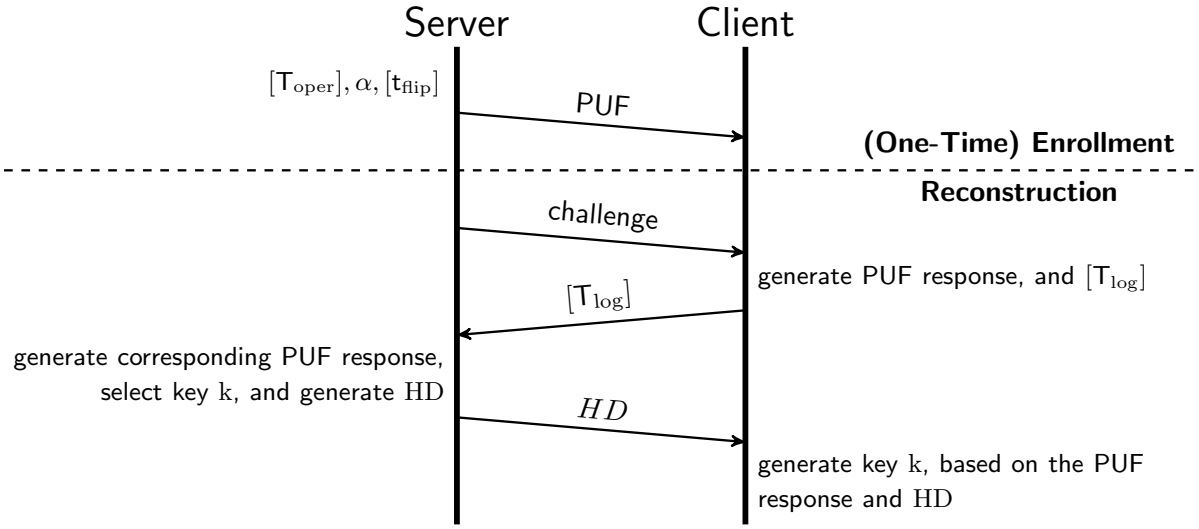


Figure 5.4: Proposed proof-of-concept robust DRAM-based cryptographic protocol (for key agreement).

5.2.1 A Proof-of-Concept Robust Security Protocol Based on a DRAM Decay-Based PUF or on a DRAM Row Hammer PUF

The proof-of-concept protocol consists of two phases; an enrollment phase that needs to be run only once, and a reconstruction phase that is run each time the relevant security function (e.g., key agreement) is to be executed. This protocol is robust to temperature variations, therefore allowing for the utilisation of a DRAM decay-based PUF or a DRAM Row Hammer PUF as the relevant security anchor. The steps of the protocol are as follows:

Enrollment Phase: One-Time Enrollment at the Server Side

1. The Server identifies the operating temperature range of the PUF, $[\mathrm{T_{oper}}]$.
2. Then, it charges all the DRAM cells and it makes a list $[\mathrm{t_{flip}}]$ of their "flipping" (i.e., decay or retention) times at a particular temperature T_{base}, which is in the middle of the operating temperature range of the PUF.
3. It also calculates the variable α used in Equation (5.1) for all other temperatures in the operating temperature range of the PUF, e.g., at temperature increments of 5°C.
4. The Server transfers the PUF to the Client.

Reconstruction Phase: Key agreement

1. In order to establish a common key, the Server sends a challenge time to the Client.

2. The Client uses the challenge time and the DRAM PUF to produce a response, while it logs, on a list $[T_{log}]$, the temperature of the DRAM at regular time intervals, e.g., every 5 seconds.

3. The Client sends $[T_{log}]$ to the Server. From $[T_{log}]$, and assuming that the Server and the Client also agreed on the initial values used for the DRAM cells, the Server can deduce the response of the PUF, i.e., which bit positions have had a value change, a bit "flip", and which have not, using Equation (5.1). If the Server finds temperature values in the log that would affect (the integrity of) the measurement, e.g., too low or too high temperature values, it can abort at this step and restart this Phase of the protocol.

4. At this step, the protocol can differentiate to fit each particular application and use case. In the case of a key agreement protocol, the Server sends helper data HD to the Client. The helper data HD are generated by the Server based on the deduced PUF response and a key chosen by the Server using a fuzzy extractor scheme upon which the Server and the Client have agreed.

5. The Client inputs HD and the generated PUF response into the same fuzzy extractor scheme in order to generate the key chosen by the Server.

It is worth noting that a series of different robust protocols can be constructed by just changing the Reconstruction Phase of the proposed protocol after its third step. Following, for example, the authentication protocol proposed by Schaller et al. [87], at step 4. of the Reconstruction Phase, the Client could send a set made of cell values of both cells that the Server knows that, with high probability, should have "flipped" ("Fast cells" in [87]) and cells that the Server knows that, with high probability, should not have "flipped" ("Slow cells" in [87]), in order to be used by the Server for authenticating the Client. In this way, the original authentication protocol could be made robust to temperature variations, while still being lightweight, practical and flexible. The same follows for attestation, identification, or even more advanced protocols, as the protocol being proposed in this subsection is highly adjustable and flexible. We need to note here that our proposed protocol can also serve to provide a virtual proof of temperature [234], if modified in such a way that it is guaranteed that the Client uses the challenge time to measure the PUF, e.g., by providing very limited additional time for the Client to respond.

It should be, therefore, evident that using the proposed protocol and its modifications, DRAM retention-based PUFs and Row Hammer PUFs can provide a cost-efficient, lightweight, flexible and practical solution to significantly enhance the security of resource-constrained IoT devices, without the need for hardware additions, since DRAMs are inherent components of most contemporary computer systems.

5.2.2 Concluding Remarks

In this subsection, we have presented a novel way of securing IoT devices, using robust DRAM PUFs. In particular, we have demonstrated that, although DRAM retention-based PUFs and of Row Hammer PUFs are highly affected by temperature variations, there are ways in which robust security protocols can be constructed using these security primitives. In this way, we have proven that IoT devices can be

secured in a lightweight, practical, flexible, and cost-efficient manner, through the usage of DRAM-based PUFs, without the need for any hardware change. Finally, we have also stressed the variety of security protocols that can be implemented using these PUFs and the fact that they can be used at run-time and provide multiple security tokens, in contrast to other memory-based PUFs, such as SRAM PUFs.

5.3 PUFs as a Security Primitive for Next-Generation IoT Devices and Networks

As already mentioned, IoT applications are slowly becoming ubiquitous, with the number of interconnected devices being estimated to be between 20 billion and 100 billion by the current year (2020) [99, 100]. It is worth noting that, nowadays, the IoT can include device segments ranging from space satellites, autonomous vehicles, and energy grids to remotely controlled robots, and heterogeneous sensors and actuators. Thus, the IoT is no longer comprised only of low-cost devices, but rather includes devices that range from high-end infrastructure servers to low-end sensors, which are, in turn, utilised in very diverse applications.

Additionally, wireless technology is also constantly evolving and wireless protocols are slowly being unified within common standards and architectures. The previously existing large diversity of protocols and interfaces is not being eliminated, but, slowly, interconnection starts to become guaranteed and common interfaces are being used much more often. For example, the ongoing commercialisation of the newest generation of wireless cellular standards, the fifth generation of such standards, which is commonly known as 5G, is not limited to cell phones and relevant network devices, but will also find application in different IoT segments, e.g., in autonomous vehicles for the use case of Vehicle-to-Everything (V2X) communication. Moreover, also in-vehicle communication is slowly moving towards a common standard for communications in the form of the automotive Ethernet protocol. Similarly, IoT networks and architectures are also evolving. However, due to the highly diverse nature of the IoT networks and the devices that are connected to them, backward compatibility will certainly be required in order to ensure a smooth transition to new standards and technologies, and to not inhibit further growth.

Within this context, the issue of securing IoT devices and networks is not one that can be addressed in a simple way. As the relevant literature indicates, security solutions for the IoT need to be low-cost, lightweight, adequate and practical, exactly due to the large diversity of devices and networks that comprise the IoT. We have already discussed that PUFs have been proposed in the literature as a potential solution for this problem, due to their lightweight and cost-efficient nature. However, we have also observed that PUF-based solutions cannot easily scale, as they are based on symmetric cryptography between each pair of devices that want to communicate. In general, when taking into account the billions of IoT devices that are operational, it is very difficult to find any potential solution that can fit to such a scale of communication.

Having these facts in mind, we propose, in this subsection, a much more scalable solution based on the implementation of a PUF on an inherent network device, a router (see Section 3.1.1), rather than on each IoT node device. In particular, the security of communications within a network cluster could be based on only the PUF of the relevant network router, instead of utilising PUFs found on each individual node device. In this way, security in the communication between nodes could be outsourced to the network components themselves, rather than be a separate task for each individual device. However, in this case, potential attacks against these network devices could put into danger not only the security of such

devices themselves, but also of large segments of the IoT network. Nevertheless, we note that, in either case, attacks against the network devices most often put in danger the whole network segment that these devices serve. Additionally, our implementation [145] has also proven that network modules, even in individual IoT devices, can also potentially serve as inherent security modules on their own. This rather obvious conclusion of our work could potentially revolutionise the industry, as each connected device may no longer require dedicated hardware or complex software algorithms, in order to be provided with security primitives and mechanisms, to secure the messages that its network module exchanges with such modules of other connected devices.

It is also worth mentioning that modern peripherals, such as the Qualcomm Atheros QCA9500 wireless communication module on which we have implemented our PUF, tend to resemble more their own (sub)systems, rather than fully dependent modules, as they feature their own memory and processor (in order to store and run the firmware supplied by the host device), device interface (to the host device), as well as their own input and output (from and to other devices, respectively) and power interfaces, as Figure 5.5 shows. Therefore, it may be possible to implement an intrinsic (potentially memory-based) PUF on most modern peripherals, including the various sensors and actuators utilised in the IoT, and therefore provide an inherent security solution for them.

Finally, we need to note that the wireless communication module on which we have implemented the PUF that this subsection examines is exactly a next-generation network module that implements the IEEE 802.11ad 60 GHz beamforming protocol. Based on this fact, we can easily conclude that our work can very easily be applied to current and next-generation IoT networks, in general.

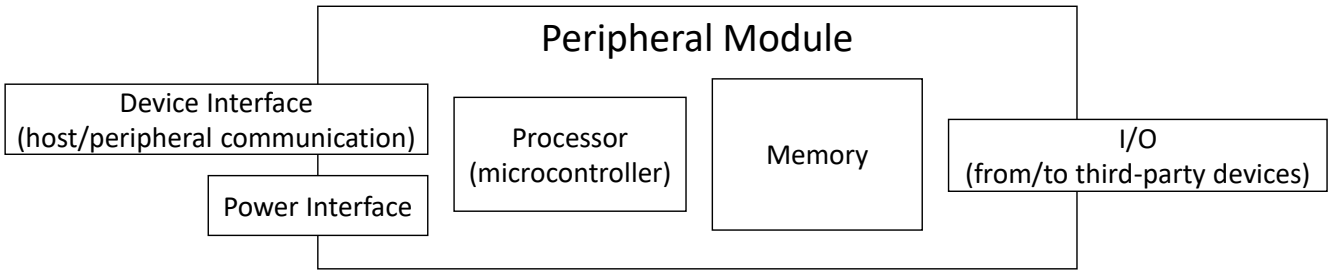


Figure 5.5: A typical modern peripheral and its components. Modern peripherals have their own processor, memory, and device interface, as well as their own input/output and power interfaces. Most often, SRAM is used in peripherals to form their memory, due to its fast speed.

5.3.1 Threat Model and Assumptions

For the purposes of this subsection, we assume an attacker with full access to the network, but very limited access to the different network devices. As the security provided by our solution is dependent upon the secrecy of the PUF response that constitutes the security anchor of our solution, an attacker with full access to the network should not be able to gain any advantage, whereas an attacker with access to a network node must not be able to gain access to the secret. Therefore, we assume that the secret is saved in the network devices in memory that requires privileged access, and that an attacker can eavesdrop, modify, inject, and replay any and all network traffic, e.g., generate and inject forged packets, but can only gain unprivileged access to any network device. We note that as our PUF resides on the memory of a peripheral module of a router, privileged access is indeed required in order to gain

access to it through the main module of the device. Furthermore, we also note that an attacker can potentially intercept, jam or deflect network packets and/or damage the network devices, which would constitute a successful DoS attack. However, taking into account the large variety of applications, use cases and environments in which IoT devices are employed, we believe that it is difficult to provide a counter-measure that would be effective against such DoS attacks, in general.

Moreover, we assume that an attacker cannot gain physical access to the memory storing the secret, the relevant device buses, and other potential side channels that may be relevant to the secret. We also assume that an attacker can gain general knowledge about the network devices and their software, but cannot utilise this knowledge in order to gain privileged access to these devices. Additionally, we assume that all cryptographic primitives employed, e.g.hashes, encoding and decoding algorithms, fuzzy extractor schemes, etc., as well as the software and firmware of the network devices, are secure. Finally, we also assume that an attacker can power-cycle any device.

5.3.2 PUF-Based Security Protocols for Next-Generation Networks and Devices

We present two security protocols based on our PUF implementation, in order to demonstrate how this PUF can be used to secure current and next-generation networks and devices (Figure 5.6). The first protocol concerns the hardware authentication of peripheral modules (Figure 5.6a), and the second deals with the authentication of the sector sweep in the 60 GHz millimetre-wave beamforming IEEE 802.11ad protocol (Figure 5.6b).

A PUF-Based Protocol for the Hardware Authentication of Peripherals

We first present a simple hardware authentication protocol that is based on our SRAM PUF implementation on the Qualcomm Atheros QCA9500 wireless communication module. We note that, in general, the security of peripherals is an important topic of current research, as a lot more devices are nowadays operating unattended than in the recent past. Attackers can, therefore, potentially replace the peripherals of such devices, in an effort to gain access to the host device [235]. Adversaries can potentially also inject malicious code in the firmware updates of the peripherals [236].

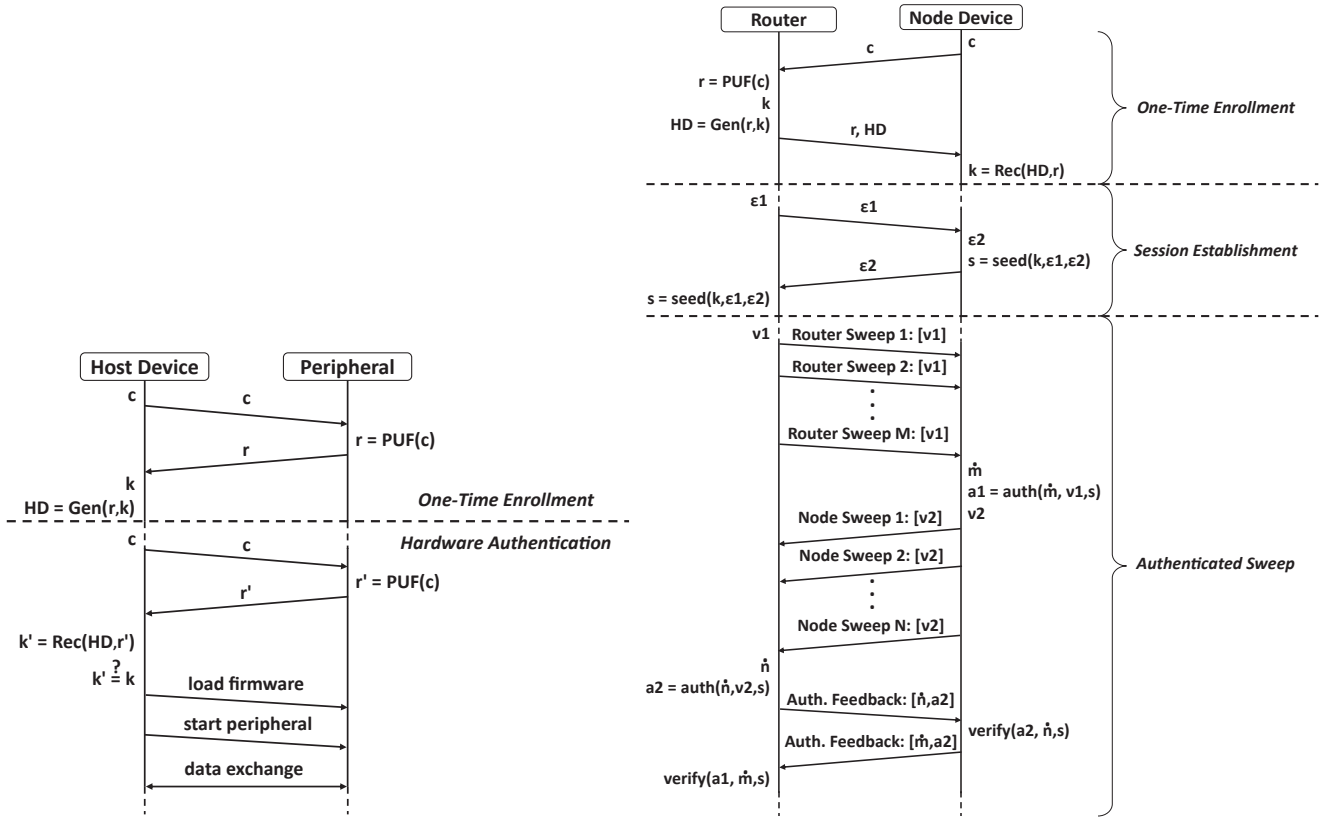
A number of works have looked into potential ways of authenticating the peripherals of a device, mostly based on software schemes [237, 238, 239, 240]. Additionally, Thibadeau [241] presented and discussed different ways in which peripherals can be attacked, and proposed a number of methods and schemes in which trust could be established between the host device and its peripherals. Nevertheless, such schemes require the addition of hardware, and/or the implementation of complicated and computationally heavy algorithms in order to function properly.

In this subsection, we demonstrate that hardware peripherals can be authenticated without the need for hardware additions or significant modification of their software. Our proposed hardware authentication protocol is based on the existence of a PUF on the peripheral that needs to be authenticated, which is the exact case that this subsection examines.

Our protocol includes an enrollment phase, which needs to be executed only once, in order for the host device to gain knowledge of the PUF response r to a challenge c, e.g., by recording the start-up values of the peripheral's SRAM. Then, this response is combined with a selected secret k, in the context

of a (reverse) fuzzy extractor scheme, to generate the helper data HD through function $Gen()$. HD and k are stored in the host device to authenticate the peripheral in the future.

In order to authenticate the peripheral module, the host device power-cycles it, so that access can be gained to the start-up values of its SRAM, assuming that it incorporates an SRAM PUF. Then, the authentication process works as shown in Figure 5.6a. The PUF response, i.e., the start-up values of the SRAM, are sent to the host device and are combined with HD, in the context of a fuzzy extractor scheme, in order to produce k' through function $Rec()$. If k' is equal to the secret k, then the peripheral is considered as authentic and is trusted by the host device. Firmware is loaded on it and normal communication is established. Otherwise, the host device assumes that a counterfeit replaced the authentic peripheral and ceases all communication with it, as it may also be malicious.



(a) A PUF-based protocol for the hardware authentication of peripherals.

(b) A PUF-based protocol to secure 60 GHz beamforming network communication.

Figure 5.6: PUF-based security protocols for next-generation networks and devices.

A PUF-Based Protocol to Secure 60 GHz Network Communication

Steinmetzer et al. [242] demonstrated in 2018 that the 60 GHz beamforming IEEE 802.11ad protocol is vulnerable to beam-stealing, as attackers may remotely convince devices to steer their antenna signals to arbitrary directions. In order to address this issue, Steinmetzer et al. [243] presented a security protocol that authenticates the sector sweep in IEEE 802.11ad networks. This protocol is based on the agreement of a secret s between the 802.11ad network router and each network node device. This secret is based on asymmetric cryptography and a PUF-based variant of it has been examined in detail

by Ahmad in [244]. Nevertheless, asymmetric cryptography can be computationally heavy and comes at a relatively high cost.

For this reason, in this work, we present a PUF-based variant of this security protocol, which is based on a symmetric key agreement between the 802.11ad router device and each network node device. In this way, we demonstrate a truly lightweight variant of this protocol, which utilises only the PUF examined in this subsection as its security anchor.

Our protocol is shown in Figure 5.6b and includes three phases. The first phase is an enrollment phase (similar to that of Section 5.3.2) that needs to be run only once and which essentially constitutes a PUF-based key agreement phase. The second phase utilises the agreed key and two nonces ($\varepsilon 1$ and $\varepsilon 2$, one selected by the router and one by the node, respectively) in order to establish the session secret s. In the third phase, an authenticated sector sweep takes place, with each party transmitting a nonce ($\nu 1$ for the router and $\nu 2$ for the node) within its sector sweep. Each party then selects the best sector ($\dot{m}$ for the router sweep and $\dot{n}$ for the node sweep) and computes an authenticator based on this sector, the nonce received and the secret s, using a truncated hash function $auth()$. Finally, after the establishment of the connection, the authenticator and the sector selected by each party are sent to the other party in order to be verified. In this way, the whole sector sweep can be authenticated. As Steinmetzer et al. [243] note, using the SHA-256 hash algorithm and utilising 4-byte nonces and 8-byte authenticators, the overhead, compared to the original sector sweep, is only 7.31%. Nevertheless, our variant of the protocol is significantly more lightweight, as it does not require the establishment of the key using asymmetric cryptography that may also involve a certification authority, but rather depends on a very simple one-time key agreement phase which is based only on the PUF of the router examined in this subsection, as illustrated in Figure 5.6b.

5.3.3 A Brief Assessment of the Security Provided

It is evident that the PUF that this subsection examines can be utilised for a large number of different security applications. However, it is also important to examine the level of security that it can provide. In particular, we note that we have assumed that an attacker cannot have physical access to any network devices, or to any relevant side channels, as well as that an attacker does not have privileged access to the memories of the devices. Such assumptions seem to be necessary in all cases in which a cryptographic token, such as a key, is stored in memory. In general, a (symmetric) secret among two or more parties needs to somehow be established and then verified. Therefore, an attacker may be able to gain knowledge of it, while it is being shared, stored, or accessed for verification. Even in the case of the one-time pad encryption technique, the secret used needs to be pre-shared and kept in the memory, until it is utilised. In particular, we again allude to the observations of Canetti and Fischlin regarding the impossibility of achieving secure cryptographic protocols without any setup assumptions [178]. Therefore, we also need to assume that an attacker cannot gain access to the SRAM PUF response during the one-off enrollment phase of the relevant protocols, which constitutes their setup phase, during which the secret is being shared. In general, it is important to note here that it has been observed that perfect security does not truly exist and that security is rather dependent on its cost, as well as the cost of the relevant attacks [179]. Therefore, as we note in Section 3.3, it is important to examine whether memory-based PUFs can provide an *acceptable* level of security that can be considered as adequate and practical.

We also again acknowledge that the formal security analysis of weak memory-based PUFs remains an open research topic, which has not yet been extensively addressed and which has attracted the attention of recent works [181, 182, 183]. Nevertheless, within the constraints of our assumed attacker model, we can briefly examine the potential of our work to provide a low-cost solution that can truly ensure an acceptable level of security. For this purpose, we observe again that the relevant literature has proven that attacks against SRAM PUFs are successful only when an attacker has physical access to the device or to a relevant side channel, or can directly access the SRAM [1, 2, 3, 4, 5, 6, 7, 24]. Therefore, we believe that our PUF implementation can be considered as secure within the context of the provided threat model.

Furthermore, we also observe that, while easy-to-implement attacks have been described against SRAM PUFs that are implemented in IoT devices [3, 4], our PUF implementation is rather more protected against software-based attacks than most other implementations, as its operation inherently requires privileged access. In particular, as our PUF is implemented on the SRAM of a peripheral module, it is only accessible through modifications of the relevant firmware and of the operating system residing in the main module of the device. Additionally, if an attacker tries to replace or steal the peripheral, this would probably be immediately noticed, exactly because of the peripheral's PUF response, which is its unique signature in the system.

Nevertheless, considering the increasing need for real-world cryptography, we do acknowledge that our PUF implementation can be successfully attacked, and proceed to propose a number of ways in which such a successful attack could potentially be addressed. An attacker may gain access to the PUF secret either through a network node or through the router device itself. Depending on whether the attacker has gained access to the PUF response itself or to a cryptographic token produced by it, e.g., a key, different remedies may be considered. For example, if only a key has been exposed, but not the relevant helper data, a new key can be agreed upon by modifying the helper data being used. However, in case the PUF response has been exposed, then our PUF implementation needs to be modified, potentially in such a way as to constitute a reconfigurable PUF based on the examined PUF and on (other) PUFs implemented on the memories of the main module of the router (host) device. Alternatively, the memories of network node devices can also be utilised as PUFs in order to restore secure communication, or the peripheral module unit could also be replaced, resulting in a new SRAM PUF instance. Depending on the risk of a successful attack, as well as its potential damages, more resources could also be invested in physically protecting the devices, or in installing tamper-resistant and/or tamper-proof components on them. We do note that all remedies would come at an increased cost, but would also provide a higher level of security.

Finally, we do also note the existence of works that consider the security of the fuzzy extractor scheme and the relevant helper data being produced, which call attention to their potential vulnerabilities [245, 246]. This is one of the reasons why we assumed that all the employed fuzzy extractor schemes, as well as all the other relevant software and firmware, are secure.

In general, we note that our security solution utilises only thePUF of a network router, in order to establish security in communications between devices in the IoT network cluster that this router serves. In this way, our solution is extremely flexible and scalable, as it requires minimal resources and power, but, at the same time, may also be less secure than other security solutions for the IoT that utilise the

PUF of every device within a network cluster. Nevertheless, we believe that the examined PUF provides an efficient and reliable low-cost security solution for current and next-generation IoT networks and devices, as it exhibits good security characteristics, even under adverse conditions, and it can be used as an adequate security anchor for the realisation of a large number of diverse cryptographic protocols at a minimal cost.

5.3.4 Concluding Remarks

In this subsection, we have taken note of the pervasive nature of the IoT and the evolving characteristics of wireless networks. Additionally, we have noted the diverse nature and varying applications of IoT devices and networks, and the huge number of such devices, which ranges in the scale of billions. For this reason, we have proposed a low-cost security solution that can be easily applied to both current and next-generation IoT networks and devices. In particular, we have implemented an SRAM PUF using the memory of the 60 GHz network module of a tri-band network router, which can support current and next-generation wireless communication protocols. In this way, we have demonstrated not only that the network modules of IoT devices can potentially serve to enhance their security, but also that it is no longer required for each device to have its own PUF for security to be established on IoT networks, as the PUFs of inherent network components, such as network routers, can be used to secure the network cluster(s) that they serve. In this way, our work provides a flexible, lightweight and cost-efficient security solution for current and next-generation IoT networks. Furthermore, we have also presented and discussed two security protocols based on this PUF implementation, in order to demonstrate how it can be used to secure current and next-generation IoT networks and devices. The first protocol concerns the hardware authentication of peripheral modules, and the second deals with the authentication of the sector sweep in the 60 GHz millimetre-wave beamforming IEEE 802.11ad protocol. We, therefore, conclude that our PUF implementation indeed constitutes a reliable, practical and efficient low-cost security solution for current and next-generation IoT networks and devices.

5.4 On the Potential of Memory-Based PUFs to Serve as Security Mechanisms in Space Applications

In recent years, COTS devices are being used in space applications more and more often. These devices have proven to be particularly suitable for near-Earth space applications, as they are produced under economies of scale, which keep their price relatively low, and can operate under adverse temperature and radiation conditions. In particular, COTS devices have been shown to be able to operate correctly even at ambient temperatures as low as −110°C [3, 4] and as high as 80°C [143]. Additionally, the effects of radiation on COTS devices and their components have also been studied in depth. For example, such components as Flash memories have been proven to remain unaffected by heavy-ion radiation as high as 2000 gray (200 krad) [221].

As already mentioned, the IoT is not limited only to terrestrial applications, but has also started to be slowly adopted in applications and devices being sent to space, such as satellites and planetary landers, in the context of cyber-physical systems and special-purpose IoT devices. Additionally, the space segment of the IoT and its terrestrial segments are bound to become more and more interconnected, with information delivered from satellite sensors directly to actuators found on our planet, ensuring in this way that immediate action can be taken when needed. One can very easily imagine the sensors of a

number of interconnected satellites detecting a fire in a particularly remote location on our planet, or an emergency situation on a boat in our planet's seas, and instructing autonomous firefighting, or rescue, vehicles and means to rush to the appropriate spot and take adequate action.

5.4.1 Memory-Based PUFs as Security Primitives in the Context of the Space Segment of the Internet of Things (IoT)

It becomes, therefore, quickly apparent that IoT technology calls for adequate security in IoT devices, applications and networks. In the aforementioned example, we would need to ensure that the satellites will only report real events, and that the terrestrial vehicles and other means will move towards the right location, and will only perform the appropriate actions. Nevertheless, as we have already noted, in order to minimise the cost of their adoption, IoT devices tend to be based on cost-efficient low-end designs, which are usually rather resource-constrained. Thus, most often, they do not incorporate complex security mechanisms, such as TPMs or other security (co-)processors. Moreover, the addition of any hardware component comes at an increased cost, which would be detrimental to the adoption of IoT devices. Nevertheless, as most such devices commonly include one or more memory components, it is not difficult to implement intrinsic memory-based PUFs based on the unique characteristics of each of these memory components. These PUFs can then act as security primitives for the implementation of cryptographic applications. In this way, such PUFs can ensure the security of not only the device itself, but also of the communication between such devices.

Nevertheless, environmental conditions in space may be significantly different from the usual conditions occurring on our planet. In particular, IoT devices used in space may be subjected to significantly higher or lower ambient temperatures, and to a much higher amount of radiation than on Earth. However, we assume that such devices are shielded by an aluminium ($_{13}$Al) alloy housing, as is most often the case for satellites and, therefore, the radiation and ambient temperatures to which they are exposed are not too extreme.

Moreover, our examination in Section 4 has demonstrated that memory-based PUFs are rather resilient to radiation, but may be heavily dependent on temperature. However, we have provided a way to produce robust protocols even when temperature-dependent memory-based PUFs are used, in Section 5.2. We can, therefore, conclude that ambient temperature is not a factor that can preclude the use of intrinsic memory-based PUFs for enhancing the security of IoT devices being employed in space applications.

Regarding radiation, we need to note that the experiments and simulations presented in Section 4.4 were conducted by irradiating the bare boards, while we assume that in space IoT devices will be protected by both the usual aluminium ($_{13}$Al) alloy satellite housing and, in the case of near-Earth orbits, by the Earth's magnetic field, which stops charged particles at the Van Allen belts. Similarly, the magnetic fields of other planets can also protect satellites from radiation, but could may also potentially interfere with their electronics. Therefore, future work should also investigate the effects of strong magnetic fields on memory-based PUFs implemented on IoT devices.

In general, we note that non-permanent *soft* errors affecting the memory used as a PUF can be removed by rebooting, erasing, or overwriting the memory module before using it as a PUF. Additionally, through the employment of a fuzzy extractor scheme, which is a very common post-processing scheme

for PUFs, we can make sure that even a few permanent *hard* errors cannot affect the correct operation of a PUF, as the fuzzy extractor scheme will always correct up to a particular number of errors, even if these persist. Moreover, each PUF can be queried a few times with the same challenge, in order to come up with a final response (corresponding to that challenge) that contains as few errors as possible, through the comparison (and further utilisation) of the (noisy) PUF responses received for that challenge. We can, therefore, conclude that even relatively high levels of radiation cannot truly inhibit the use of intrinsic memory-based PUFs for enhancing the security of IoT devices being employed in space applications, at least in near-Earth space orbits.

Finally, in the context of IoT space applications, the usage of SRAM and Flash PUFs seems rather more advantageous than the use of DRAM-based PUFs, as certain categories of the latter appear to exhibit a far greater dependency on ambient temperature variations. In particular, our tests regarding SRAM and Flash PUFs on both an ST Microelectronics STM32F407 Discovery (STM32F407G-DISC1) board and an ST Microelectronics STM32 Nucleo-64 NUCLEO-L152RE board [54] have shown that such PUFs can be considered as suitable for the implementation of IoT security applications, e.g., key agreement, device identification, and authentication, even under adverse conditions, such as the ones expected in space applications.

5.4.2 Concluding Remarks

In this subsection, we have noted the inherent need for security in IoT applications in extra-terrestrial environments. We have also noted that, while most IoT devices do not include dedicated security components, such as TPMs, they most often incorporate memory modules, which can be used for the implementation of memory-based PUFs. Such PUFs can act as intrinsic security primitives and serve as a basis for the implementation of cryptographic applications, in order to enhance the security of IoT devices. However, in order to be used in space, their operation must be resilient to radiation and ambient temperature variations. In this subsection, we briefly discuss the effects of ambient temperature variations on memory-based PUFs, and note that robust cryptographic protocols can be constructed even with the use of temperature-dependent memory-based PUFs. We also demonstrate that memory-based PUFs do not seem to be significantly affected by radiation and note that ways exist to recover soft errors and correct a number of hard errors. We conclude, therefore, that memory-based PUFs constitute adequate lightweight, practical, flexible and cost-efficient security solutions for IoT implementations in space.

5.5 Reconfigurable PUFs With Advanced Applications

In this subsection, we propose a number of Advanced Reconfigurable PUF (AR-PUF) implementations that are based on intrinsic memory-based PUFs found inherently on IoT and CPS devices. Such implementations can provide a significantly increased amount of entropy in comparison to the standalone memory-based PUFs that they incorporate. Since the reconfigurable PUF (rPUF) implementations we propose are based only on customised software components and the inherent hardware of the devices, we can construct particularly lightweight, flexible, and practical security primitives that can be used to secure IoT and CPS devices and networks, through the implementation of a number of cryptographic

[54] We note that both of these boards can be used in the implementation of IoT applications, and are used in space applications by the German Aerospace Center (Deutsches Zentrum für Luft- und Raumfahrt e.V. – DLR).

applications. Additionally, we show that such *advanced* rPUFs can also be used to implement a number of protocols that allow the security of IoT and CPS devices to be restored after such devices have been compromised.

5.5.1 Advanced Reconfigurable PUF (AR-PUF) Designs

In this subsection, we very briefly present implementations of AR-PUFs that take advantage of either a software or a hardware RM. In this way, we prove that AR-PUFs are flexible, as they allow for different realisations, which can be used in very different use cases and applications.

AR-PUFs With a Software Reconfiguration Module

In general, we note that software RMs are easy to implement, cost-efficient, flexible and practical, but also easier to attack than hardware, as they are easier to access. We present two AR-PUF implementations that utilise a software RM, one that uses a reshuffling tree structure as its RM and one that is based on sequential hashing. Both implementations are expected to be able to provide a similar degree of security and are extremely lightweight.

- A Reshuffling Tree as a Software RM

  As Figure 5.7 shows, the data structure of a tree that reshuffles every time the rPUF is queried can serve as an AR-PUF. In particular, the tree contains both fakes and challenges. Every time the server sends a challenge, the branch that contains this challenge in the rightmost and bottom-most position is selected, with a preference for paths on the right. A history of challenges is saved and the response of the rPUF consists of the concatenation of the contents of the selected branch, on which challenges are replaced by the relevant responses of the standalone (memory-based) PUF.

- Sequential Hashing as a Software RM

  In a similar way, a sequential hashing scheme can also be used as an RM for an AR-PUF. In this case, every hash result (which constitutes the response of this AR-PUF) is produced by hashing the concatenation of the previous hash result and the response of the standalone (memory-based) PUF to the provided challenge. The first hash is produced by concatenating the response of the standalone PUF with a chosen salt value and then hashing this concatenation. In this way, a chain of hashes is produced, which can be traced back to the history of challenges, which is again saved. Different hash algorithms can be employed, as long as they are considered secure, especially in terms of lack of collisions among the hash results, and “one-wayness”, i.e., the inability of an attacker to acquire the hash input from the hash output.

AR-PUFs With a Hardware Reconfiguration Module

Hardware RMs are harder to attack, but may be less flexible and practical, unless replacements are made easily available, perhaps in a scheme similar to SIM cards used in cell phones. The AR-PUFs modules that utilise a hardware RM are inspired by the Combination PUF proposed by Sutar et al. [130], which combines an SRAM PUF with a DRAM PUF, and the rPUF proposed by Eichhorn et al.[118], which uses an NVM as an RM for logical reconfiguration of the relevant standalone PUF.

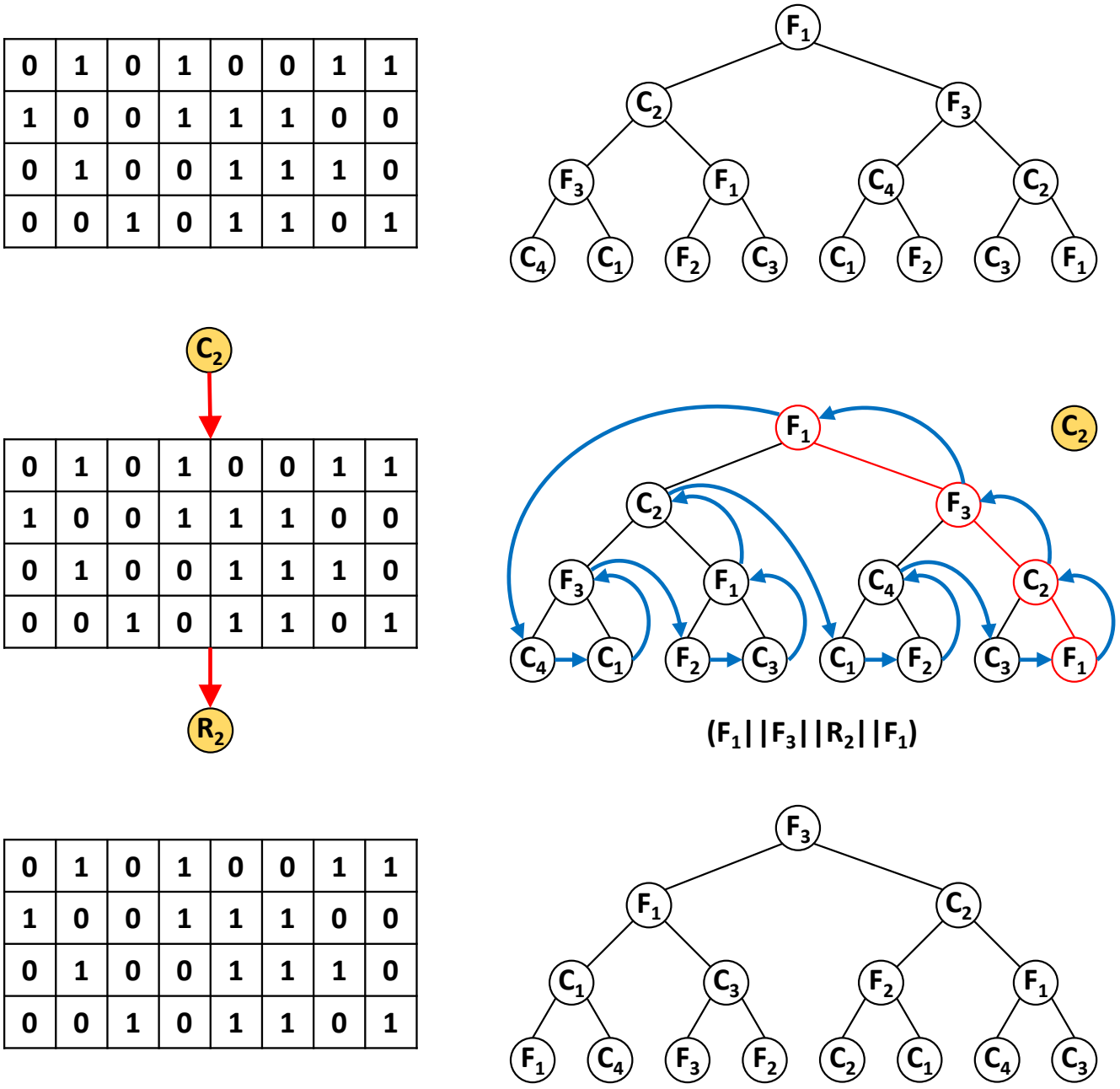


Figure 5.7: Example of an AR-PUF using a reshuffling tree as an RM. The PUF module is represented as a memory-based PUF, which can provide normal challenge-response pairs (left). The reshuffling tree acts as the RM (right), by providing the path sequence (red) that contains the challenge received (yellow), with a preference for nodes on the right and at the bottom. The path contains fakes, as well as actual challenges that correspond to responses of the memory-based PUF. The response of the AR-PUF is a concatenation of the contents of the selected path sequence, where challenges have been replaced by the relevant responses. After a path sequence has been used, the tree reshuffles according to some pre-determined algorithm (blue arrows). The challenge used each time can be saved, in order to keep a history log.

- A Removable DIMM DRAM as a Hardware RM

  A PUF based on a removable Dual In-line Memory Module (DIMM) DRAM module can easily be used as an RM for an AR-PUF, as DRAM PUFs can produce multiple CRPs [48] and removable

DRAMs allow for flexibility, while being inherent system components. Another inherent memory-based PUF can be used in combination with this RM, thus making this solution lightweight and cost-efficient. Such a concept is somehow discussed in [130], which, however, fails to identify the relevant *advanced* security applications of this scheme. In relevance to such applications, we again require that a history of challenges be saved.

- A Removable Flash Memory as a Hardware RM

  Similarly to the case of the removable DRAM module, a PUF based on a removable Flash memory module can also be used in combination with another standalone memory-based PUF, in order to create an AR-PUF. A removable external Flash memory connected through a Serial Peripheral Interface (SPI) bus can be used. A Flash-memory-based PUF can produce multiple CRPs, while the removable nature of the relevant Flash memory allows for flexibility, while still being an inherent system component. Again, another inherent memory-based PUF can be used in combination with this RM, thus making this solution lightweight and cost-efficient. In order to allow for advanced security applications, again it is required that a history of the challenges used is saved.

5.5.2 Advanced Security Applications of AR-PUFs

AR-PUFs can significantly enhance the security of IoT and CPS devices and networks, as they can be used as primitives in common cryptographic applications, such as key agreement, device identification, authentication, (remote) attestation, etc., providing a higher level of security than the standalone PUFs incorporated into such devices, due to their higher inherent entropy. Furthermore, they can also be used to implement two novel security functions that allow the security of a partially or fully compromised device to be restored. Both of these security protocols can be implemented using any one of the previously described AR-PUFs. A presentation of the two relevant protocols follows.

Recovery of the Legitimate State Based on the History

This protocol, an example use case of which is depicted in Figure 5.8, allows the server to potentially prove that the AR-PUF has been logically tampered with and to recover communication with the client possessing this AR-PUF by reverting to a previous state of its RM. It is important to note here that our protocol assumes that a one-off enrollment phase has taken place before its start, during which the server has made a model of the AR-PUF, i.e., has noted the initial state of the AR-PUF, has made a list of relevant CRPs that the standalone PUF produces, and has observed how the RM works, so that it can predict the AR-PUF's CRPs correctly. It is also assumed that at the end of this enrollment phase, the AR-PUF is in the possession of the client.

As Figure 5.8 shows, at some point, while the server and the client are using the AR-PUF to secure their communication, an attacker may appear and try to read a large amount of CRPs of the AR-PUF, in order to try to gain some advantage in correctly predicting future CRPs (i.e., to attempt a modelling or machine learning attack). This attack is quite commonly used against "strong" PUF implementations, as they have proven to be vulnerable to machine learning and modelling attacks [17]. However, each CRP exchange between the attacker and the client will alter the state of the RM of the AR-PUF.

Our protocol demonstrates that if the server then tries to use a new CRP in the communication with the client, the client will fail to authenticate, as the client's and the server's configuration states do not

match. At this point, the server could potentially invoke the history of CRP exchanges [55] being kept by the client and compare it to the history of CRP exchanges kept by itself, thus not only finding out about the attack, but also being able to recover secure communication with the client, by rolling back the history of both devices to the last configuration state that they share [56].

In order to prevent the attacker from using this recovery option to its own benefit, communication during the recovery phase could be encrypted based on a pre-determined configuration state of the AR-PUF. Nevertheless, even by simply disallowing the very first configuration state from being exchanged during the recovery phase, we make sure that the attacker will never be able to gain access to all possible future CRPs within a limited timeframe and without being revealed (as the client and the server can always just return to the initial state of the respective AR-PUF device and model and start over, while the attacker cannot do so).

[55] Essentially, only a history of challenges having been used is needed.

[56] This is possible, as the client and the server can get their AR-PUF device and model, respectively, into its initial state and then re-apply to them the list of challenges that have been identified as legitimate.

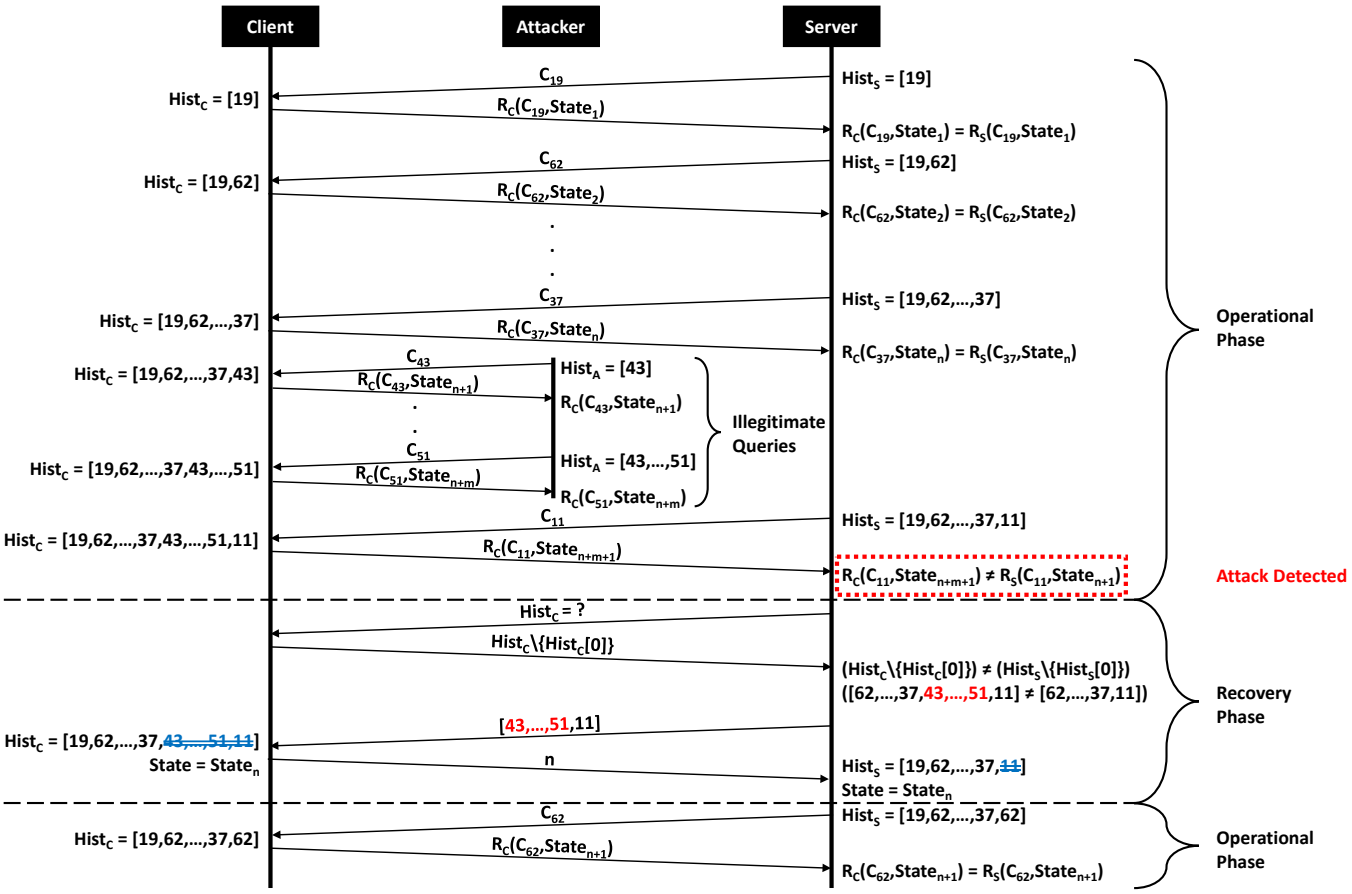


Figure 5.8: Example use case of the recovery protocol, based on the history functionality.

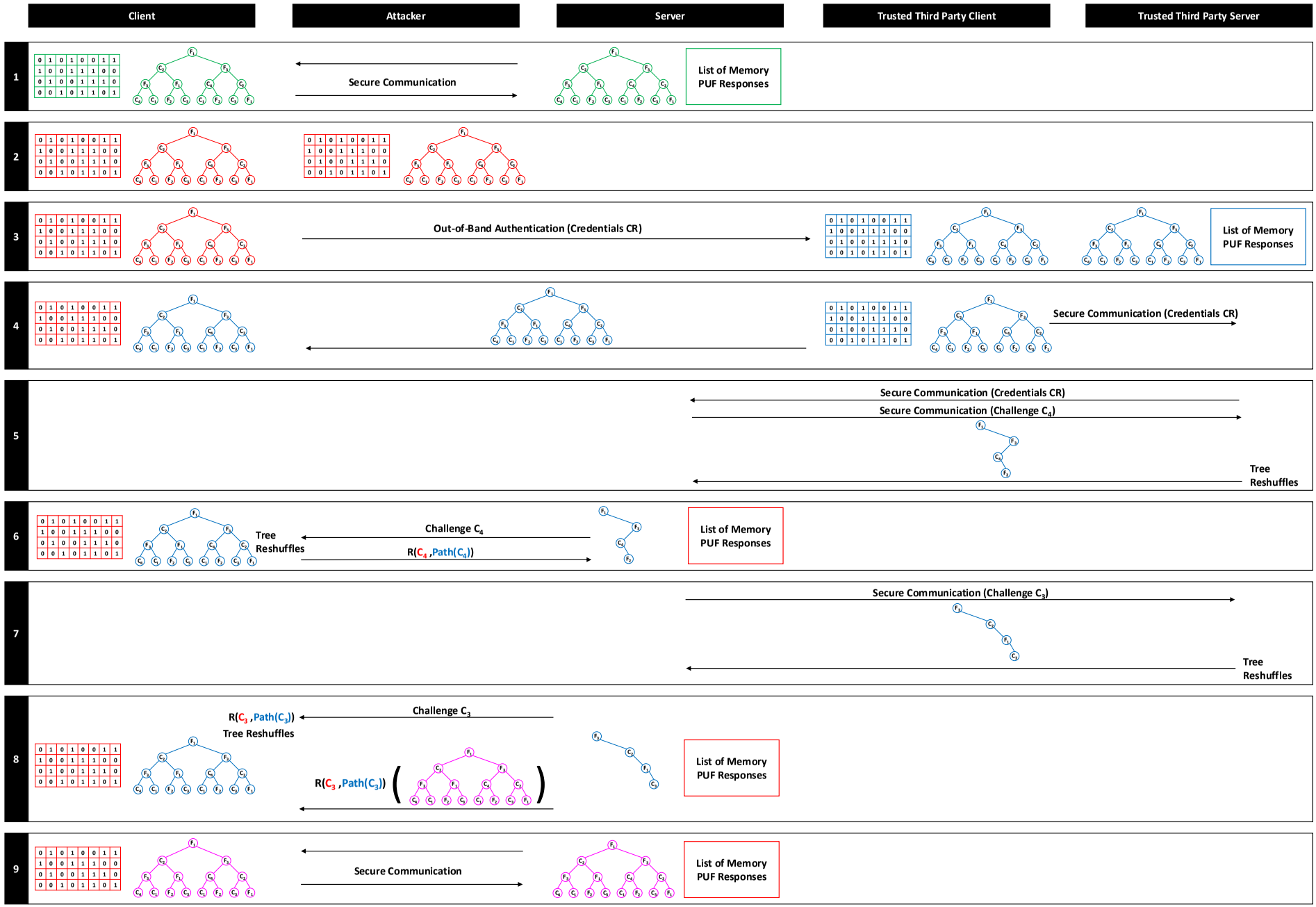


Figure 5.9: Example use case of the partial renewal protocol, using a (semi-)trusted third party.

Partial Renewal of a Fully Compromised AR-PUF Through a Third Party

The second protocol we have implemented allows, through the help of a third party, for the partial renewal of an AR-PUF that has been fully compromised, as shown in Figure 5.9. It is important to note here that this protocol also assumes that a one-time enrollment phase has taken place before its start, during which the relevant server has made a model of the AR-PUF, i.e., has noted the initial state of the AR-PUF, has made a list of relevant CRPs that the standalone (memory-based) PUF produces, and has observed how the RM works, so that it can predict the AR-PUF's CRPs correctly. It is also assumed that at the end of this enrollment phase, the AR-PUF is in the possession of the relevant client.

In the first step of this protocol, we assume that regular secure communication, using the AR-PUF, is taking place between the client and the relevant server, S_c. We can, for example, assume that, in the first step of our protocol, the client and the server S_c have used an AR-PUF, which may e.g., utilise a reshuffling tree as an RM, to implement a key agreement protocol. However, our protocol can be implemented using any of the AR-PUFs described in this subsection. Of course, in the case of AR-PUFs with a hardware module, such a module should be readily available to be purchased from the third party, as would be the case, for example, with a SIM card for a cell phone.

In the second step of the protocol, it is assumed that an attacker ATT has completely compromised the AR-PUF and therefore is able to predict all future CRPs, even if the client and the server S_c rewind the relevant CRP history, as discussed previously. However, the client still has the AR-PUF, as our attacker wants to gain access to the secret communication between the server S_c and the client. In case the client no longer had that the AR-PUF, then the client and the server S_c would need to use a different AR-PUF, after a relevant one-off enrollment phase, and, at the same time, would be certain that an attack has taken place. This is not the case that this protocol addresses. Our protocol address a case in which the client either suspects or knows that the security of the AR-PUF has been compromised, but still has possession of it. An example of a relevant use case would be that the client has left the AR-PUF unattended in an insecure environment for some time [57].

In our protocol, security is restored with the help of a third party, which, as we will explain, does not need to be fully trusted, but should also rather not be malevolent [58]. In this case, this third party should have its own AR-PUF, as shown in the third step of our protocol, which, of course, is used by the third party in order to communicate to its relevant server, S_{tp}. In this step, an out-of-band authentication takes place between the client and the third party, as the third party needs to make sure through the credentials CR of the client, that the "client" is indeed the legitimate owner of its own AR-PUF and not some attacker.

---

[57] This is a very realistic scenario in practice, which is relevant also to other security primitives and secrets. For example, if one forgets, for a couple of hours or days, the keys to one's home and home safe at one's office, to which cleaning staff and others may have access, and one has stored 500 million euros in one's home safe, then one will obviously attempt to renew the security of these keys (which may be regular – entirely physical – keys) in some way, even if one is rather confident that their security has *probably* not been compromised.

[58] In the relevant example of a key, one may buy new keylocks and keys, without fully trusting the third party from which these are purchased, but assuming that such a third party does not have full knowledge of the relevant keylocks, keys, *and* one's home and home safe, where they will be installed. Both in the case of a compromised key and in the case of a compromised AR-PUF, a relative abundance of relevant third parties, e.g., entities selling keys or supporting AR-PUFs, and a widespread use of the relevant primitives, keys and AR-PUFs, would make the assumption of a *semi*-trusted third party entirely realistic.

In the fourth step of the protocol, the third party provides the authenticated client with access to the RM of the third party's AR-PUF, and sends the credentials CR of the client to its own (i.e., the third party's) server, S_{tp}. Now, a new AR-PUF instance has been formed, consisting of the standalone PUF of the client and the third party's RM. In the fifth step, the third party's server, S_{tp}, sends the credentials CR of the client to the server relevant to the client, S_c. In this way, S_c can know which client may have an issue and, thus, also which AR-PUF, as well as check the relevant credentials CR. Then, S_c sends a challenge C_α to S_{tp}, and S_{tp} replies to S_c with the relevant information regarding the state of the RM of the third party's AR-PUF [59], for which it has a model. The challenge C_α is also used in order to make the state stored at the server S_{tp} for the RM of the third party's AR-PUF change accordingly, but the response of the third party's AR-PUF, $R_{tp\ [\alpha]} = \text{AR-PUF}_{tp}(C_\alpha)$, that may be produced at S_{tp} will not be used further. Moreover, it is assumed, in general, that communication between S_c and S_{tp}, as well as between S_{tp} and the third party, is secure.

In the sixth step, the server of the client, S_c, uses the acquired information regarding the state of the third party's AR-PUF and the model it has for the client's AR-PUF in order to produce the response relevant to the composite AR-PUF that the client currently has, which is comprised of the client's standalone (memory-based) PUF and the third party's RM (as discussed in the fourth step). Then, S_c sends challenge C_α to the client, who replies with the response of the composite AR-PUF, $R_{(PUF_{client}+RM_{tp})\ [\alpha]} = \text{AR-PUF}_{(PUF_{client}+RM_{tp})}(C_\alpha)$. In this way, the server can confirm that the client has a composite AR-PUF composed of the client's standalone PUF and the third party's RM. Essentially, now, secure communication has been restored between the client and the relevant server S_c, as neither the attacker ATT nor the third party (nor S_{tp}) can predict the composite AR-PUF's CRPs, as the attacker lacks information on the third party's RM and the third party (as well as S_{tp}) lack(s) information on the client's standalone PUF. Additionally, any other entity lacks information regarding both the client's standalone PUF and the third party's RM. Moreover, now, the state of the third party's RM has been synchronised with the state stored at S_{tp}.

In the seventh step, S_c sends a challenge C_β to S_{tp}, and S_{tp} replies to S_c with the relevant information regarding the current state of the RM of the third party's AR-PUF, for which it has a model. The challenge C_β is also used in order to make the state stored at the server S_{tp} for the RM of the third party's AR-PUF change accordingly, but the response of the third party's AR-PUF, $R_{tp\ [\beta]} = \text{AR-PUF}_{tp}(C_\beta)$, that may be produced at S_{tp} will not be used further. We need to note that between the sixth and the seventh step, the third party and S_{tp} may use the third party's AR-PUF, including its RM, to communicate, which will change the state of the relevant RM significantly. This is discouraged, as it may delay the execution of our protocol, but not forbidden. At the end of this step, S_c has acquired access to another CRP of the composite AR-PUF that consists of the client's standalone PUF and the third party's RM.

In the eighth step, S_c sends challenge C_β to the client, who computes the relevant response of the composite AR-PUF, $R_{(PUF_{client}+RM_{tp})\ [\beta]} = \text{AR-PUF}_{(PUF_{client}+RM_{tp})}(C_\beta)$. Again, now, the state of the third party's RM has been synchronised with the state stored at S_{tp}. Additionally, S_c sends a new RM encrypted using $R_{(PUF_{client}+RM_{tp})\ [\beta]}$. Since the client has computed $R_{(PUF_{client}+RM_{tp})\ [\beta]}$, it can decrypt the new RM and start using it instead of its previous RM about which the attacker ATT has knowledge. Since

[59] For example, which path of the tree will be followed for this challenge C_α, if the RM is a reshuffling tree, or what is the current salt to be used and with which hash algorithm, if the AR-PUF is based on sequential hashing, or, if a PUF is used as a hardware RM, what its response would be, etc.

the third party and S_{tp} do not have knowledge of the client's standalone PUF, they cannot compute $R_{(PUF_{client}+RM_{tp})\ [\beta]}$ and gain knowledge of the new RM. Any other entity does not have any information regarding the client's standalone PUF or the third party's RM. At this point, the third party, stops giving the client access to its RM. Throughout the protocol's run, the client or S_c cannot compromise the third party's AR-PUF as they have no information regarding the third party's standalone PUF.

We need to note here that while new software RMs can be directly sent to the client through network communication, new hardware RMs should be readily available at the site of the third party, in the same way that SIM cards are extremely easy to acquire. Then, in the same way as SIM cards need to be registered or unlocked, before they can be used, also such a new hardware RM would require the information sent by S_c regarding it in order to be usable, e.g., by needing to be unlocked or "registered" with S_c before the client could use them. For example, such information could be the order in which memory cells are to be concatenated, or which memory regions are to be used, in case the new RM is a memory-based PUF. In this case, however, S_c would not send a new RM *per se*, but rather only information regarding how it should be used in order to establish secure communication. Nevertheless, without this information, no other entity apart from the client, could use the new RM correctly, and therefore such a hardware RM cannot be considered as compromised even if the third party knows the general characteristics of it, because the third party still would not be able to gain full knowledge on how it is to be used, and, even more importantly, the third party does not have knowledge of the client's standalone PUF. Obviously, in case the new RM is software-based, this can be sent encrypted by S_c, e.g., in the form of a new reshuffling tree, or a hash function and a new *initial* salt, and the third party can gain no knowledge on it. One interesting result of our protocol is that the client can change the form of the RM being used for its AR-PUF using our protocol, e.g., if the original RM was a PUF, S_c can even choose to send a reshuffling tree as a new RM, or a DRAM PUF may substitute sequential hashing, or a Flash PUF, as the relevant RM, as long as the device incorporating the client's AR-PUF allows for it.

In the ninth step, secure communication has been fully restored between the client and S_c, as they both have the needed information in order to communicate using an AR-PUF that consists of the client's standalone PUF and the new RM sent by S_c to the client in the previous protocol step. This partially renewed AR-PUF is not as secure as the original uncompromised AR-PUF that the client possessed, but cannot be compromised by attacker ATT or any other entity, without having the new RM (and for entities other than ATT, also the client's standalone PUF) compromised. Our protocol has, therefore, succeeded in restoring completely security using the capabilities of AR-PUFs, even after a completely successful attack against an AR-PUF instance that was completely compromised. We also note that this was done at a low cost, as only the RM, which may potentially be only a software source of entropy, was renewed.

Regarding the third party, it is evident that it does not to be a fully trusted authority, but, at the same time, needs to be a semi-honest semi-trusted party, as the protocol is not secure if the third party (or S_{tp}) collude with attacker ATT. Examples of such a third party may include entities such as the post office or a bank, which can verify that the client is legitimate, and not an attacker, before allowing a composite AR-PUF to be produced, composed of the client's standalone PUF and the third party's RM, and which are not expected to collude with attackers. Depending on the identity of the expected attacker, however, the potential of different entities to serve as semi-honest semi-trusted parties may change.

5.5.3 Concluding Remarks

In this subsection, we have presented very briefly a number of novel Advanced implementations of Reconfigurable PUFs (AR-PUFs), which utilise either a software or a hardware reconfiguration module, while being based on inherent components found in most IoT and CPS devices. Additionally, we have presented two novel security protocols that can be implemented based on AR-PUFs. The first of these protocols can allow such PUFs to potentially be logically tamper-proof and to quickly recover after an attacker has tried to gather a, potentially significant, number of their CRPs. The second protocol presented allows for a compromised AR-PUF to be partially renewed, through the help of a semi-trusted third party. Security is restored, without revealing any secrets to the third party, and while also preventing the attacker from gaining any further information. We have, therefore, proven that AR-PUFs can be implemented on most IoT and CPS devices, providing a flexible, lightweight, and cost-efficient security solution that can significantly enhance their security, in practice. Finally, we note that we have successfully examined and tested proof-of-concept implementations of all of the AR-PUFs and the security protocols described in this subsection. AR-PUFs have the potential to serve in different segments and applications related to the security not only of IoT and CPS devices and networks, but also of such varying applications as smart transportation, smart cards, and telecommunications.

6 Conclusion

In this work, we investigated whether PUFs constitute truly *practical* security mechanisms in the context of the IoT. In particular, we first proposed a new definition for PUFs that addressed the shortcomings of existing ones, and discussed why memory-based PUFs constitute suitable security mechanisms for the IoT. Then, we briefly examined the literature relevant to PUFs in general, as well as to their role as security solutions in the framework of the IoT, in particular. Subsequently, we presented the most relevant designs of memory-based PUFs, introduced the most important metrics to assess their security, and discussed the relation between security and cost, especially in the context of practical applications. In this context, we tested and evaluated instances of memory-based PUF designs under both nominal and adverse environmental conditions, in order to identify potential dependencies of such PUFs that an attacker could extremely easily exploit. Finally, we presented and examined a number of practical security solutions for the IoT, which utilised the memory-based PUFs examined in this work, in such a way as to address their shortcomings, exploit their advantages, and provide even advanced security applications in practice. In this way, we proved that memory-based PUFs can indeed provide lightweight practical security in the context of IoT applications.

More specifically, we examined, in Section 1.1, what can be considered as a PUF through existing definitions and proposed a new one that addresses the shortcomings of the pre-existing proposals, through the concept of "Physical *Unique* Functions". In this way, we were able to provide an answer to the first research question addressed by this work: "What can be considered as a PUF?". We, then, discussed, in Section 1.2, the nature of PUFs, their operation principles and their essential properties, in order to investigate, in Section 1.3, whether PUFs could be utilised as security mechanisms in the framework of practical IoT applications. In this way, we were able to address the second research question that our work answers: "Why do PUFs appear to be suitable for enhancing the security of the IoT?".

As we identified, in Section 1.3, that memory-based PUFs constitute one of the most fitting security solutions for the IoT [60], we then proceeded to present and briefly discuss the relevant existing literature regarding PUFs in general, including memory-based PUFs, in Section 2.1, and regarding PUFs as a security mechanism for the IoT, in Section 2.2. We also introduced and briefly considered existing works that concerned reconfigurable PUFs, in Section 2.3, as such PUFs allow for advanced security applications and can potentially be implemented through the combination of multiple memory-based PUFs together.

Subsequently, in order to address the next research question: "Can memory-based PUFs serve as adequate and acceptable security solutions for the IoT?", we, first, presented the most relevant designs of memory-based PUFs in Section 3.1, discussed the most important metrics utilised to assess their characteristics in Section 3.2, and the relation between security and cost in Section 3.3, in order to, then, be able to examine and assess, in Section 4, whether these PUFs can provide adequate security under both nominal and adverse environmental conditions. In particular, we evaluate the ability of the responses of memory-based PUFs to provide adequate security not only under nominal conditions, in Section 4.1, but also under ambient temperature and supply voltage variations, in Section 4.2 and Section 4.3 respectively. We also examine the effects of ionising radiation on such PUFs, in Section 4.4.

[60] In this way, we addressed another research question relevant to our work: "Which PUF types are the (most) suitable to serve as security mechanisms for the IoT?".

In this way, memory-based PUFs are examined as security solutions for IoT applications both in nominal conditions and in adverse environments.

Finally, in order to address the final research question: "Can memory-based PUFs be utilised to provide security in realistic applications and use case scenarios, in the context of the IoT?", we presented and examined, in Section 5, a number of practical security protocols and schemes that utilise memory-based PUFs in such a manner as to not only exploit their advantages, but also to address their shortcomings, and allow them to provide even rather advanced security applications, in the framework of realistic use case scenarios for the IoT. In particular, generic key agreement, as well as device identification and authentication, protocols that utilise memory-based PUFs were introduced, examined, and discussed in Section 5.1. Then, a protocol that allows for run-time security applications and, at the same time, addresses the potential dependencies of the relevant DRAM-based PUFs on the ambient temperature was introduced, examined and discussed in Section 5.2. Subsequently, a protocol based on an SRAM PUF was proposed for the security of current and next-generation IoT devices and networks, as well as examined, and discussed in Section 5.3. The potential of memory-based PUFs to serve as security mechanisms in space applications was explored in Section 5.4, and, finally, the ability of memory-based reconfigurable PUFs to provide advanced security applications was investigated in Section 5.5.

By managing to adequately address the relevant research questions, we believe that we were able to prove that memory-based PUFs can indeed constitute adequate and acceptable lightweight security mechanisms for practical IoT applications, and that, in this way, we were able to answer the main research question that our work examined: "Can memory-based PUFs be utilised in order to provide security for IoT devices, networks, and applications in a practical and lightweight manner?", in the affirmative.

In general, it should be further noted that some of the most important contributions of this work include:

- Providing a new definition for PUFs. Our work addresses the proverbial "elephant in the room". We provide a realistic definition of PUFs as physical inanimate objects that realise particular functions (with errors) in a rather unique manner. In this way, we clarify that PUFs are physical objects, rather than actual mathematical functions. Additionally, we refrain from defining PUFs as objects or functions that have specific security properties, such as unclonability or tamper-evidence, as previous definitions did. In this way, we avoid the fallacy of *security by definition*, which, in practice, seems to have significantly hindered the adoption of PUFs as security mechanisms. By comparing the nature and characteristics of PUFs to that of biometrics, we reveal the true potential of PUFs for practical security applications, in a manner similar to biometrics.

- Considering the relation between security and cost in the context of PUFs. Our work is one of the few that observes and discusses the relation between security and cost in the context of potential attacks and the framework of *acceptable* risk. In this way, we attempt to provide a framework regarding what can truly be considered as *practical* security in the context of real-world engineering. To this end, we observe that most works about security tend to make unrealistic assumptions, and that systems and services that are, in general, considered as secure, may not provide a high, or even an *acceptable* level of security in practice. This does not mean that the relevant security systems are not secure up to a certain level, provided that the assumptions made for them hold, but rather

that security is a *relative* concept, and as *perfect* security is infeasible to exist, the merits and shortcomings of each security solution need to be assessed for each different use case and application in a rather individual manner, and by taking into account the relevant levels of *acceptable* cost and *acceptable* risk, in the context of potential attacks, and the gains and/or damages associated with them. Only in this way, the relevant level of *acceptable* security can be established, in relation to the relevant costs, and a particular security solution can be considered as practical or not.

- Examining and addressing the dependency of DRAM-based PUFs on ambient temperature variations. Previous works tended to either underestimate or exaggerate the dependencies of certain DRAM-based PUFs on ambient temperature variations, depending on whether they were focusing on the role of such PUFs as security mechanisms or were proposing relevant temperature-based attacks against these PUFs (or were trying to suggest that a different type of PUFs or another security mechanism would be better than the relevant DRAM PUFs). In this work, we examine in detail the dependency of the DRAM decay-based PUF and the DRAM Row Hammer PUF on ambient temperature variations, as well as the effects of other environmental variations on these and other memory-based PUFs, and propose and examine a security protocol that can overcome this dependency in order to provide robust security, even at the presence of ambient temperature variations. In general, we note that it may be possible to provide robust protocols in a similar manner to the protocol that we present in this work, in order to address other potential dependencies of memory-based PUFs and/or other security mechanisms on external factors, such as environmental changes and variations.

- Providing a lightweight and practical PUF-based solution to secure current and next-generation IoT devices and networks. Our work proposes and examines a lightweight PUF-based solution, which utilises the SRAM of the network peripheral module of a (potentially, local) router, i.e., of the network (routing) node of the relevant (sub)network, in order to provide security both for the network communication and for the relevant routing device, as well as for the network peripheral module itself. We also propose that the other node devices of the relevant network can be secured through the implementation of memory-based PUFs on each of them, when this is feasible. However, the employment of these PUFs (each of which would be found on each of the individual nodes of the network), could potentially increase significantly the relevant communication overheads and costs. Therefore, we propose that such PUFs are only used to secure each of the relevant node devices, and that only the memory-based PUF of the network node is used to secure the communication between the different node devices, including the relevant network (routing) node. In this way, we can secure current and next-generation IoT devices and networks, as our implementation of such a PUF and the relevant protocol on a next-generation tri-band network router, which supports wireless communication at 2.4 GHz, 5 GHz and 60 GHz, and implements the IEEE 802.11ad 60 GHz beamforming protocol, proves.

- Proving that PUFs can be utilised as security mechanisms in the context of IoT space applications. To the best of our knowledge, our work is the first that considers the utilisation of PUFs for securing the space segment of the IoT, and manages to prove that memory-based PUFs could be employed as a lightweight and practical security mechanism in the context of IoT space applications. To this

end, we examine the effects of ionising radiation and low temperatures on such PUF, and prove that none of these environmental factors could truly preclude the use of intrinsic memory-based PUF for enhancing the security of IoT devices that are employed in space applications, at least in near-Earth space orbits.

- Introducing AR-PUFs and examining their advanced security applications. Our work proposes a new category of (reconfigurable) PUFs, namely Advanced Reconfigurable PUFs (AR-PUFs). These PUFs are based on the combination of a memory-based PUF and a software-based source of entropy, or on the combination of multiple memory-based PUFs together, in order to provide advanced security applications. As, in either of these cases, the relevant components, i.e., the memory-based PUFs and the software potentially being required, most often form inherent components of the same system, these PUFs are not only highly flexible, and scalable, due to their ability to be reconfigured, but are also rather lightweight, and cost-efficient. In this way, AR-PUFs can be utilised for regular security applications in practice. Additionally, their ability to offer a number of advanced security applications, e.g., being able to return to a legitimate state after being illegitimately utilised, and being able to recover their level of confidentiality after being fully compromised by a successful attack, makes their use as security mechanisms rather desirable, not only in the context of the IoT, but perhaps also even more broadly.

By making these contributions, our work aims to have a potentially high impact on the adoption of memory-based PUFs in practice, and to significantly contribute to rejuvenating the relevant scientific field.

Moreover, we note that, as the IoT and its applications become more and more pervasive and ubiquitous, separate IoT (sub)systems, (sub)networks, or even entire IoT segments, of a widely heterogeneous nature, are consolidated into larger systems of systems and networks of networks, sometimes even in a temporary ad hoc manner. Additionally, a new generation of the IoT, i.e., IoT 2.0, is currently emerging, taking advantage of the currently evolving technologies of artificial intelligence, the blockchain, and machine learning. Therefore, future works should address the role of PUFs as security mechanisms in the context of systems of systems and networks of networks, especially in the context of the emerging IoT 2.0 concept. To this end, we have published some preliminary works in order to define and explore this novel scientific field [247, 248]. Finally, future works should also examine the effects of aging on the quality of the responses of memory-based PUFs, and investigate the potential for the implementation of relevant low-cost attacks based on such potential effects of aging.

Referenced Bibliography

[1] C. Helfmeier, C. Boit, D. Nedospasov, and J. Seifert. **"Cloning Physically Unclonable Functions"**. In: *2013 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*. IEEE, 2013, pp. 1–6. DOI: 10.1109/HST.2013.6581556 (cit. on pp. 1, 2, 80, 180).

[2] A. Roelke and M. R. Stan. **"Attacking an SRAM-Based PUF through Wearout"**. In: *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 2016, pp. 206–211. DOI: 10.1109/ISVLSI.2016.68 (cit. on pp. 1, 80, 180).

[3] N. A. Anagnostopoulos, T. Arul, M. Rosenstihl, A. Schaller, S. Gabmeyer, and S. Katzenbeisser. **"Low-Temperature Data Remanence Attacks Against Intrinsic SRAM PUFs"**. In: *2018 21st Euromicro Conference on Digital System Design (DSD)*. IEEE, 2018, pp. 581–585. DOI: 10.1109/DSD.2018.00102 (cit. on pp. 1, 40, 80, 84, 85, 104, 105, 109, 125, 180, 181).

[4] N. A. Anagnostopoulos, T. Arul, M. Rosenstihl, A. Schaller, S. Gabmeyer, and S. Katzenbeisser. **"Attacking SRAM PUFs Using Very-Low-Temperature Data Remanence"**. In: *Microprocessors and Microsystems* 71 (2019). ISSN: 0141-9331. DOI: 10.1016/j.micpro.2019.102864. URL: http://www.sciencedirect.com/science/article/pii/S0141933118305076 (cit. on pp. 1, 40, 80, 84, 85, 104, 105, 109, 125, 180, 181).

[5] S. Zeitouni, Y. Oren, C. Wachsmann, P. Koeberl, and A. Sadeghi. **"Remanence Decay Side-Channel: The PUF Case"**. In: *IEEE Transactions on Information Forensics and Security* 11.6 (2016), pp. 1106–1116. ISSN: 1556-6021. DOI: 10.1109/TIFS.2015.2512534 (cit. on pp. 1, 2, 80, 109, 180).

[6] Y. Oren, A.-R. Sadeghi, and C. Wachsmann. **"On the Effectiveness of the Remanence Decay Side-Channel to Clone Memory-Based PUFs"**. In: *Cryptographic Hardware and Embedded Systems – CHES 2013*. Ed. by G. Bertoni and J.-S. Coron. Vol. 8086. Lecture Notes in Computer Science (LNCS). Springer, 2013, pp. 107–125. ISBN: 9783642403491. DOI: 10.1007/978-3-642-40349-1_7 (cit. on pp. 1, 2, 80, 109, 180).

[7] C. Helfmeier, C. Boit, D. Nedospasov, S. Tajik, and J. Seifert. **"Physical Vulnerabilities of Physically Unclonable Functions"**. In: *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2014, pp. 1–4. DOI: 10.7873/DATE.2014.363 (cit. on pp. 1, 2, 80, 180).

[8] S. Tajik, E. Dietz, S. Frohmann, H. Dittrich, D. Nedospasov, C. Helfmeier, J.-P. Seifert, C. Boit, and H.-W. Hübers. **"Photonic Side-Channel Analysis of Arbiter PUFs"**. In: *Journal of Cryptology* 30.2 (2017), pp. 550–571. ISSN: 1432-1378. DOI: 10.1007/s00145-016-9228-6. URL: https://doi.org/10.1007/s00145-016-9228-6 (cit. on pp. 1, 2, 28).

[9] F. Ganji, S. Tajik, and J.-P. Seifert. **"Let Me Prove It to You: RO PUFs Are Provably Learnable"**. In: *Information Security and Cryptology – ICISC 2015*. Ed. by S. Kwon and A. Yun. Vol. 9558. Lecture Notes in Computer Science (LNCS). Springer, 2016, pp. 345–358. ISBN: 9783319308401. DOI: 10.1007/978-3-319-30840-1_22 (cit. on pp. 1, 2).

[10] U. Rührmair, S. Devadas, and F. Koushanfar. **"Security Based on Physical Unclonability and Disorder"**. In: *Introduction to Hardware Security and Trust*. Ed. by M. Tehranipoor and C. Wang. Springer, 2012, pp. 65–102. ISBN: 9781441980809. DOI: 10.1007/978-1-4419-8080-9_4. URL: https://doi.org/10.1007/978-1-4419-8080-9_4 (cit. on p. 2).

[11] 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: Foundations and Practice*. Ed. by A.-R. Sadeghi and D. Naccache. Springer, 2010, pp. 3–37. ISBN: 9783642144523. DOI: 10.1007/978-3-642-14452-3_1 (cit. on pp. 2, 168).

[12] R. Maes, P. Tuyls, and I. Verbauwhede. **"Low-Overhead Implementation of a Soft Decision Helper Data Algorithm for SRAM PUFs"**. In: *Cryptographic Hardware and Embedded Systems – CHES 2009*. Ed. by C. Clavier and K. Gaj. Vol. 5747. Lecture Notes in Computer Science (LNCS). Springer, 2009, pp. 332–347. ISBN: 9783642041389. DOI: 10.1007/978-3-642-04138-9_24 (cit. on pp. 2, 3).

[13] R. Maes, P. Tuyls, and I. Verbauwhede. **"A Soft Decision Helper Data Algorithm for SRAM PUFs"**. In: *2009 IEEE International Symposium on Information Theory (ISIT 2009)*. IEEE, 2009, pp. 2101–2105. DOI: 10.1109/ISIT.2009.5205263 (cit. on pp. 2, 3, 168).

[14] A. Das, Ü. Kocabaş, A.-R. Sadeghi, and I. Verbauwhede. **"PUF-Based Secure Test Wrapper Design for Cryptographic SoC Testing"**. In: *2012 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2012, pp. 866–869. DOI: 10.1109/DATE.2012.6176618 (cit. on pp. 2, 3).

[15] J. Guajardo, S. S. Kumar, G.-J. Schrijen, and P. Tuyls. **"FPGA Intrinsic PUFs and Their Use for IP Protection"**. In: *Cryptographic Hardware and Embedded Systems – CHES 2007*. Ed. by P. Paillier and I. Verbauwhede. Vol. 4727. Lecture Notes in Computer Science (LNCS). Springer, 2007, pp. 63–80. ISBN: 9783540747352. DOI: 10.1007/978-3-540-74735-2_5 (cit. on pp. 2, 3, 24, 25, 34, 168).

[16] A. Van Herrewege, S. Katzenbeisser, R. Maes, R. Peeters, A.-R. Sadeghi, I. Verbauwhede, and C. Wachsmann. **"Reverse Fuzzy Extractors: Enabling Lightweight Mutual Authentication for PUF-Enabled RFIDs"**. In: *Financial Cryptography and Data Security – FC 2012*. Ed. by A. D. Keromytis. Vol. 7397. Lecture Notes in Computer Science (LNCS). Springer, 2012, pp. 374–389. ISBN: 9783642329463. DOI: 10.1007/978-3-642-32946-3_27 (cit. on pp. 2, 3, 36, 87, 99, 103, 120, 168).

[17] S. Katzenbeisser, Ü. Kocabaş, V. van der Leest, A.-R. Sadeghi, G.-J. Schrijen, H. Schröder, and C. Wachsmann. **"Recyclable PUFs: Logically Reconfigurable PUFs"**. In: *Cryptographic Hardware and Embedded Systems – CHES 2011*. Ed. by B. Preneel and T. Takagi. Vol. 6917. Lecture Notes in Computer Science (LNCS). Springer, 2011, pp. 374–389. ISBN: 9783642239519. DOI: 10.1007/978-3-642-23951-9_25 (cit. on pp. 2, 3, 30, 186).

[18] P. Koeberl, Ü. Kocabaş, and A. Sadeghi. **"Memristor PUFs: A New Generation of Memory-Based Physically Unclonable Functions"**. In: *2013 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2013, pp. 428–431. DOI: 10.7873/DATE.2013.096 (cit. on pp. 2, 25).

[19] C. Bösch, J. Guajardo, A.-R. Sadeghi, J. Shokrollahi, and P. Tuyls. **"Efficient Helper Data Key Extractor on FPGAs"**. In: *Cryptographic Hardware and Embedded Systems – CHES 2008*. Ed. by E. Oswald and P. Rohatgi. Vol. 5154. Lecture Notes in Computer Science (LNCS). Springer, 2008, pp. 181–197. ISBN: 9783540850533. DOI: 10.1007/978-3-540-85053-3_12 (cit. on pp. 2, 3).

[20] F. Armknecht, R. Maes, A.-R. Sadeghi, B. Sunar, and P. Tuyls. **"Memory Leakage-Resilient Encryption Based on Physically Unclonable Functions"**. In: *Advances in Cryptology – ASIACRYPT 2009*. Ed. by M. Matsui. Vol. 5912. Lecture Notes in Computer Science (LNCS). Springer, 2009, pp. 685–702. ISBN: 9783642103667. DOI: 10.1007/978-3-642-10366-7_40 (cit. on pp. 2, 3).

[21] C. Wachsmann and A.-R. Sadeghi. **"Physically Unclonable Functions (PUFs): Applications, Models, and Future Directions"**. Ed. by E. Bertino and R. Sadhu. 1st Edition. Vol. 12. Synthesis Lectures on Information Security, Privacy, and Trust. Morgan & Claypool, 2014. ISBN: 9781627055093. DOI: 10.2200/S00622ED1V01Y201412SPT012. URL: https://doi.org/10.2200/S00622ED1V01Y201412SPT012 (cit. on pp. 2, 3).

[22] H. Handschuh, G.-J. Schrijen, and P. Tuyls. **"Hardware Intrinsic Security From Physically Unclonable Functions"**. In: *Towards Hardware-Intrinsic Security: Foundations and Practice*. Ed. by A.-R. Sadeghi and D. Naccache. Springer, 2010, pp. 39–53. ISBN: 9783642144523. DOI: 10.1007/978-3-642-14452-3_2 (cit. on pp. 2, 3, 168).

[23] F. Armknecht, R. Maes, A.-R. Sadeghi, B. Sunar, and P. Tuyls. **"Memory Leakage-Resilient Encryption Based on Physically Unclonable Functions"**. In: *Towards Hardware-Intrinsic Security: Foundations and Practice*. Ed. by A.-R. Sadeghi and D. Naccache. Springer, 2010, pp. 135–164. ISBN: 9783642144523. DOI: 10.1007/978-3-642-14452-3_6. URL: https://doi.org/10.1007/978-3-642-14452-3_6 (cit. on pp. 2, 3).

[24] D. Nedospasov, J. Seifert, C. Helfmeier, and C. Boit. **"Invasive PUF Analysis"**. In: *2013 Workshop on Fault Diagnosis and Tolerance in Cryptography*. IEEE, 2013, pp. 30–38. DOI: 10.1109/FDTC.2013.19 (cit. on pp. 2, 80, 180).

[25] X. Xu and W. Burleson. **"Hybrid Side-Channel/Machine-Learning Attacks on PUFs: A New Threat?"** In: *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2014, pp. 1–6. DOI: 10.7873/DATE.2014.362 (cit. on pp. 2, 18).

[26] G. Hospodar, R. Maes, and I. Verbauwhede. **"Machine Learning Attacks on 65nm Arbiter PUFs: Accurate Modeling Poses Strict Bounds on Usability"**. In: *2012 IEEE International Workshop on Information Forensics and Security (WIFS)*. IEEE, 2012, pp. 37–42. DOI: 10.1109/WIFS.2012.6412622 (cit. on pp. 2, 18).

[27] 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*. CCS '10. Association for Computing Machinery, 2010, pp. 237–249. ISBN: 9781450302456. DOI: 10.1145/1866307.1866335. URL: http://doi.acm.org/10.1145/1866307.1866335 (cit. on pp. 2, 18, 19, 24).

[28] 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"**. In: *IEEE Transactions on Information Forensics and Security* 8.11 (2013), pp. 1876–1891. ISSN: 1556-6021. DOI: 10.1109/TIFS.2013.2279798 (cit. on pp. 2, 18, 24).

[29] A. Cherkaoui, L. Bossuet, L. Seitz, G. Selander, and R. Borgaonkar. **"New Paradigms for Access Control in Constrained Environments"**. In: *2014 9th International Symposium on Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC)*. IEEE, 2014, pp. 1–4. DOI: 10.1109/ReCoSoC.2014.6861362 (cit. on p. 3).

[30] U. Rührmair, H. Busch, and S. Katzenbeisser. **"Strong PUFs: Models, Constructions, and Security Proofs"**. In: *Towards Hardware-Intrinsic Security: Foundations and Practice*. Ed. by A.-R. Sadeghi and D. Naccache. Springer, 2010, pp. 79–96. ISBN: 9783642144523. DOI: 10.1007/978-3-642-14452-3_4. URL: https://doi.org/10.1007/978-3-642-14452-3_4 (cit. on p. 3).

[31] F. Armknecht, R. Maes, A.-R. Sadeghi, F.-X. Standaert, and C. Wachsmann. **"A Formalization of the Security Features of Physical Functions"**. In: *Proceedings of the 2011 IEEE Symposium on Security and Privacy*. IEEE, 2011, pp. 397–412. DOI: 10.1109/SP.2011.10 (cit. on p. 3).

[32] P. Koeberl, J. Li, A. Rajan, C. Vishik, and W. Wu. **"A Practical Device Authentication Scheme Using SRAM PUFs"**. In: *Trust and Trustworthy Computing – Trust 2011*. Ed. by J. M. McCune, B. Balacheff, A. Perrig, A.-R. Sadeghi, A. Sasse, and Y. Beres. Vol. 6740. Lecture Notes in Computer Science (LNCS). Springer, 2011, pp. 63–77. ISBN: 9783642215995. DOI: 10.1007/978-3-642-21599-5_5 (cit. on p. 3).

[33] K. Kursawe, A.-R. Sadeghi, D. Schellekens, B. Škorić, and P. Tuyls. **"Reconfigurable Physical Unclonable Functions — Enabling Technology for Tamper-Resistant Storage"**. In: *2009 IEEE International Workshop on Hardware-Oriented Security and Trust (HOST)*. IEEE, 2009, pp. 22–29. DOI: 10.1109/HST.2009.5225058 (cit. on pp. 7, 29, 30).

[34] R. S. Pappu. **"Physical One-Way Functions"**. Ph.D. dissertation. Massachusetts Institute of Technology, 2001. URL: https://dspace.mit.edu/handle/1721.1/45499 (cit. on pp. 8, 9, 23, 25, 30).

[35] R. Pappu, B. Recht, J. Taylor, and N. Gershenfeld. **"Physical One-Way Functions"**. In: *Science* 297.5589 (2002), pp. 2026–2030. ISSN: 0036-8075. DOI: 10.1126/science.1074376. eprint: https://science.sciencemag.org/content/297/5589/2026.full.pdf. URL: https://science.sciencemag.org/content/297/5589/2026 (cit. on pp. 8, 23, 25, 30).

[36] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas. **"Silicon Physical Random Functions"**. In: *Proceedings of the 9th ACM Conference on Computer and Communications Security*. CCS '02. Association for Computing Machinery, 2002, pp. 148–160. ISBN: 1581136129. DOI: 10.1145/586110.586132. URL: https://doi.org/10.1145/586110.586132 (cit. on pp. 8, 9, 23, 25, 29).

[37] B. L. P. Gassend. **"Physical Random Functions"**. M.Sc. thesis. Massachusetts Institute of Technology, 2003. URL: https://dspace.mit.edu/handle/1721.1/37606 (cit. on pp. 8, 9, 23, 25).

[38] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas. **"Controlled Physical Random Functions"**. In: *Proceedings of the 18th Annual Computer Security Applications Conference (CSAC 2002)*. IEEE, 2002, pp. 149–160. DOI: 10.1109/CSAC.2002.1176287 (cit. on pp. 8, 9, 29).

[39] B. Gassend, M. van Dijk, D. Clarke, and S. Devadas. **"Controlled Physical Random Functions"**. In: *Security with Noisy Data: On Private Biometrics, Secure Key Storage and Anti-Counterfeiting*. Ed. by P. Tuyls, B. Škorić, and T. Kevenaar. Springer, 2007, pp. 235–253. ISBN: 9781846289842. DOI: 10.1007/978-1-84628-984-2_14. URL: https://doi.org/10.1007/978-1-84628-984-2_14 (cit. on pp. 8, 9, 29).

[40] B. Gassend, M. van Dijk, D. Clarke, E. Torlak, S. Devadas, and P. Tuyls. **"Controlled Physical Random Functions and Applications"**. In: *ACM Transactions on Information and System Security* 10.4 (2008). ISSN: 1094-9224. DOI: 10.1145/1284680.1284683. URL: https://doi.org/10.1145/1284680.1284683 (cit. on pp. 8, 9, 29).

[41] P. Bulens, F.-X. Standaert, and J.-J. Quisquater. **"How to Strongly Link Data and Its Medium: The Paper Case"**. In: *IET Information Security* 4 (3 2010), pp. 125–136. ISSN: 1751-8709. DOI: 10.1049/iet-ifs.2009.0032. URL: https://digital-library.theiet.org/content/journals/10.1049/iet-ifs.2009.0032 (cit. on p. 9).

[42] J. D. R. Buchanan, R. P. Cowburn, A.-V. Jausovec, D. Petit, P. Seem, G. Xiong, D. Atkinson, K. Fenton, D. A. Allwood, and M. T. Bryan. **"'Fingerprinting' Documents and Packaging"**. In: *Nature* 436.7050 (2005), pp. 475–475. ISSN: 1476-4687. DOI: 10.1038/436475a. URL: https://doi.org/10.1038/436475a (cit. on p. 9).

[43] National Research Council, Division on Engineering and Physical Sciences, National Materials Advisory Board, Commission on Engineering and Technical Systems, Committee on Next-Generation Currency Design. **"Counterfeit Deterrent Features for the Next-Generation Currency Design"**. National Academy Press, 1993. ISBN: 9780309050289. Publication NMAB-472 (cit. on pp. 9, 23).

[44] C.-W. Wong and M. Wu. **"A Study on PUF Characteristics for Counterfeit Detection"**. In: *2015 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2015, pp. 1643–1647. DOI: 10.1109/ICIP.2015.7351079 (cit. on p. 9).

[45] S. Voloshynovskiy, M. Diephuis, F. Beekhof, O. Koval, and B. Keel. **"Towards Reproducible Results in Authentication Based on Physical Non-Cloneable Functions: The Forensic Authentication Microstructure Optical Set (FAMOS)"**. In: *2012 IEEE International Workshop on Information Forensics and Security (WIFS)*. IEEE, 2012, pp. 43–48. DOI: 10.1109/WIFS.2012.6412623 (cit. on p. 9).

[46] W. Clarkson, T. Weyrich, A. Finkelstein, N. Heninger, J. A. Halderman, and E. W. Felten. **"Fingerprinting Blank Paper Using Commodity Scanners"**. In: *2009 30th IEEE Symposium on Security and Privacy*. IEEE, 2009, pp. 301–314. DOI: 10.1109/SP.2009.7 (cit. on p. 9).

[47] M. Diephuis, S. Voloshynovskiy, T. Holotyak, N. Stendardo, and B. Keel. **"A Framework for Fast and Secure Packaging Identification on Mobile Phones"**. In: *Media Watermarking, Security, and Forensics 2014*. Ed. by A. M. Alattar, N. D. Memon, and C. D. Heitzenrater. Vol. 9028. Proceedings of SPIE. SPIE – The International Society for Optics and Photonics, 2014, pp. 296–305. DOI: 10.1117/12.2039638 (cit. on p. 9).

[48] Q. Tang, C. Zhou, W. Choi, G. Kang, J. Park, K. K. Parhi, and C. H. Kim. **"A DRAM Based Physical Unclonable Function Capable of Generating $>10^{32}$ Challenge Response Pairs Per 1Kbit Array for Secure Chip Authentication"**. In: *2017 IEEE Custom Integrated Circuits Conference (CICC)*. IEEE, 2017, pp. 1–4. DOI: 10.1109/CICC.2017.7993610 (cit. on pp. 14, 19, 30, 71, 126, 136, 142, 143, 148, 169, 185).

[49] Cisco. **"White Paper: The Internet of Things Reference Model"**. 2014. URL: http://cdn.iotwf.com/resources/71/IoT_Reference_Model_White_Paper_June_4_2014.pdf (Accessed in 2021). *Draft – Controlled Distribution.* (Cit. on pp. 14, 15).

[50] Intrinsic ID. **"SRAM PUF Technology"**. 2021. URL: https://www.intrinsic-id.com/sram-puf/ (Accessed in 2021) (cit. on p. 15).

[51] C. Herder, M. D. Yu, F. Koushanfar, and S. Devadas. **"Physical Unclonable Functions and Applications: A Tutorial"**. In: *Proceedings of the IEEE* 102.8 (2014), pp. 1126–1141. ISSN: 0018-9219. DOI: 10.1109/JPROC.2014.2320516 (cit. on pp. 18, 19, 24).

[52] W. Yan, C. Jin, F. Tehranipoor, and J. A. Chandy. **"Phase Calibrated Ring Oscillator PUF Design and Implementation on FPGAs"**. In: *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 2017, pp. 1–8. DOI: 10.23919/FPL.2017.8056859 (cit. on pp. 18, 24).

[53] U. Rührmair, C. Jaeger, and M. Algasinger. **"An Attack on PUF-Based Session Key Exchange and a Hardware-Based Countermeasure: Erasable PUFs"**. In: *Financial Cryptography and Data Security – FC 2011*. Ed. by G. Danezis. Vol. 7035. Lecture Notes in Computer Science (LNCS). Springer, 2011, pp. 190–204. ISBN: 9783642275760. DOI: 10.1007/978-3-642-27576-0_16 (cit. on pp. 18, 30).

[54] D. W. Bauder. **"An Anti-Counterfeiting Concept for Currency Systems"**. Tech. rep. PTK-11990. Sandia National Labs, Albuquerque, New Mexico, USA, 1983 (cit. on p. 23).

[55] G. J. Simmons. **"A System for Verifying User Identity and Authorization at the Point-of Sale or Access"**. In: *Cryptologia* 8.1 (1984), pp. 1–21. DOI: 10.1080/0161-118491858737 (cit. on p. 23).

[56] S. N. Graybeal and P. B. McFate. **"Getting Out of the STARTing Block"**. In: *Scientific American* 261.6 (1989), pp. 61–67. URL: https://www.jstor.org/stable/24987511 (cit. on p. 23).

[57] G. J. Simmons. **"Identification of Data, Devices, Documents and Individuals"**. In: *Proceedings of the 25th Annual IEEE International Carnahan Conference on Security Technology*. IEEE, 1991, pp. 197–218. DOI: 10.1109/CCST.1991.202215 (cit. on p. 23).

[58] Defense Policy Panel of the House Committee on Armed Services of the United States Congress. **"Breakout, Verification, and Force Structure: Dealing with the Full Implications of START"**. Report of the Defense Policy Panel of the Committee on Armed Services. House of Representatives, One Hundredth Congress, Second Session. U.S. Government Printing Office, 1988 (cit. on p. 23).

[59] D. E. Holcomb, W. P. Burleson, and K. Fu. **"Initial SRAM State as a Fingerprint and Source of True Random Numbers for RFID Tags"**. In: *Proceedings of the Conference on RFID Security 2007*. 2007. URL: http://www.rfidsec07.etsit.uma.es/slides/papers/paper-12.pdf (cit. on pp. 24, 25).

[60] 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: *Digest of Technical Papers of the 2004 Symposium on VLSI Circuits (VLSIC)*. IEEE, 2004, pp. 176–179. DOI: 10.1109/VLSIC.2004.1346548 (cit. on pp. 24, 25).

[61] M. Majzoobi, F. Koushanfar, and M. Potkonjak. **"Testing Techniques for Hardware Security"**. In: *2008 IEEE International Test Conference*. IEEE, 2008, pp. 1–10. DOI: 10.1109/TEST.2008.4700636 (cit. on p. 24).

[62] T. McGrath, I. E. Bagci, Z. M. Wang, U. Roedig, and R. J. Young. **"A PUF Taxonomy"**. In: *Applied Physics Reviews* 6.1 (2019). DOI: 10.1063/1.5079407. URL: https://doi.org/10.1063/1.5079407 (cit. on p. 24).

[63] P. Tuyls, G.-J. Schrijen, B. Škorić, J. van Geloven, N. Verhaegh, and R. Wolters. **"Read-Proof Hardware from Protective Coatings"**. In: *Cryptographic Hardware and Embedded Systems – CHES 2006*. Ed. by L. Goubin and M. Matsui. Vol. 4249. Lecture Notes in Computer Science (LNCS). Springer, 2006, pp. 369–383. ISBN: 9783540465614. DOI: 10.1007/11894063_29 (cit. on p. 25).

[64] G. E. Suh and S. Devadas. **"Physical Unclonable Functions for Device Authentication and Secret Key Generation"**. In: *Proceedings of the 44th Annual Design Automation Conference*. DAC '07. Association for Computing Machinery, 2007, pp. 9–14. ISBN: 9781595936271. DOI: 10.1145/1278480.1278484. URL: https://doi.org/10.1145/1278480.1278484 (cit. on p. 25).

[65] Y. Su, J. Holleman, and B. Otis. **"A 1.6 pJ/bit 96% Stable Chip-ID Generating Circuit Using Process Variations"**. In: *Digest of Technical Papers of the 2007 IEEE International Solid-State Circuits Conference (ISSCC 2007)*. IEEE, 2007, pp. 406–611. DOI: 10.1109/ISSCC.2007.373466 (cit. on p. 25).

[66] M. Majzoobi, F. Koushanfar, and M. Potkonjak. **"Lightweight Secure PUFs"**. In: *2008 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2008, pp. 670–673. DOI: 10.1109/ICCAD.2008.4681648 (cit. on p. 25).

[67] S. S. Kumar, J. Guajardo, R. Maes, G.-J. Schrijen, and P. Tuyls. **"Extended Abstract: The Butterfly PUF Protecting IP on Every FPGA"**. In: *2008 IEEE International Workshop on Hardware-Oriented Security and Trust (HOST)*. IEEE, 2008, pp. 67–70. DOI: 10.1109/HST.2008.4559053 (cit. on p. 25).

[68] R. Maes, P. Tuyls, and I. Verbauwhede. **"Intrinsic PUFs from Flip-Flops on Reconfigurable Devices"**. In: *3rd Benelux Workshop on Information and System Security (WISSec2008)*. 2008 (cit. on p. 25).

[69] D. Suzuki and K. Shimizu. **"The Glitch PUF: A New Delay-PUF Architecture Exploiting Glitch Shapes"**. In: *Cryptographic Hardware and Embedded Systems – CHES 2010*. Ed. by S. Mangard and F.-X. Standaert. Vol. 6225. Lecture Notes in Computer Science (LNCS). Springer, 2010, pp. 366–382. ISBN: 9783642150319. DOI: 10.1007/978-3-642-15031-9_25 (cit. on p. 25).

[70] P. Prabhu, A. Akel, L. M. Grupp, W.-K. S. Yu, G. E. Suh, E. Kan, and S. Swanson. **"Extracting Device Fingerprints from Flash Memory by Exploiting Physical Variations"**. In: *Trust and Trustworthy Computing – Trust 2011*. Ed. by J. M. McCune, B. Balacheff, A. Perrig, A.-R. Sadeghi, A. Sasse, and Y. Beres. Vol. 6740. Lecture Notes in Computer Science (LNCS). Springer, 2011, pp. 188–201. ISBN: 9783642215995. DOI: 10.1007/978-3-642-21599-5_14 (cit. on pp. 25, 34, 37, 65, 69, 71, 100).

[71] M. Majzoobi, G. Ghiaasi, F. Koushanfar, and S. R. Nassif. **"Ultra-Low Power Current-Based PUF"**. In: *2011 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2011, pp. 2071–2074. DOI: 10.1109/ISCAS.2011.5938005 (cit. on p. 25).

[72] 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 (HOST)*. IEEE, 2011, pp. 134–141. DOI: 10.1109/HST.2011.5955011 (cit. on p. 25).

[73] 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 (HOST)*. IEEE, 2012, pp. 7–12. DOI: 10.1109/HST.2012.6224311 (cit. on p. 25).

[74] T. Okamura (岡村利彦), K. Minematsu (○峯松一彦), Y. Tsunoo (角尾幸保), T. Iida (飯田伴則), T. Kimura (木村隆幸), and K. Nakamura (中村考一). **"DRAM PUF"**. In: *Proceedings of the 29th Symposium on Cryptography and Information Security (SCIS 2012)*. Institute of Electronics, Information and Communication Engineers. 2012. *In Japanese.* (Cit. on pp. 25, 34, 71, 84, 169).

[75] D. Fainstein, S. Rosenblatt, A. Cestero, N. Robson, T. Kirihata, and S. S. Iyer. **"Dynamic Intrinsic Chip ID Using 32nm High-K/Metal Gate SOI Embedded DRAM"**. In: *2012 Symposium on VLSI Circuits (VLSIC)*. IEEE, 2012, pp. 146–147. DOI: 10.1109/VLSIC.2012.6243832 (cit. on pp. 25, 71).

[76] C. Keller, N. Felber, F. Gürkaynak, H. Kaeslin, and P. Junod. **"Physically Unclonable Functions for Secure Hardware"**. Swiss National Science Foundation (SNSF), Nano-Tera.CH, RTD 2010 – QCrypt. 2012. URL: http://www.nanotera.ch/pdf/posters2012/QCrypt53.pdf. *Poster.* (Cit. on pp. 25, 36, 71, 80, 87, 99, 103, 120, 168).

[77] D. E. Holcomb and K. Fu. **"Bitline PUF: Building Native Challenge-Response PUF Capability into Any SRAM"**. In: *Cryptographic Hardware and Embedded Systems – CHES 2014*. Ed. by L. Batina and M. Robshaw. Vol. 8731. Lecture Notes in Computer Science (LNCS). Springer, 2014, pp. 510–526. ISBN: 9783662447093. DOI: 10.1007/978-3-662-44709-3_28 (cit. on p. 25).

[78] L. Bossuet, X. T. Ngo, Z. Cherif, and V. Fischer. **"A PUF Based on a Transient Effect Ring Oscillator and Insensitive to Locking Phenomenon"**. In: *IEEE Transactions on Emerging Topics in Computing* 2.1 (2014), pp. 30–36. DOI: 10.1109/TETC.2013.2287182 (cit. on p. 25).

[79] F. Tehranipoor, N. Karimian, K. Xiao, and J. Chandy. **"DRAM Based Intrinsic Physical Unclonable Functions for System Level Security"**. In: *Proceedings of the 25th Great Lakes Symposium on VLSI*. GLSVLSI '15. Association for Computing Machinery, 2015, pp. 15–20. ISBN: 9781450334747. DOI: 10.1145/2742060.2742069. URL: https://doi.org/10.1145/2742060.2742069 (cit. on pp. 25, 71, 84, 85, 88, 104, 126, 142, 143, 169).

[80] M. S. Hashemian, B. Singh, F. Wolff, D. Weyer, S. Clay, and C. Papachristou. **"A Robust Authentication Methodology Using Physically Unclonable Functions in DRAM Arrays"**. In: *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2015, pp. 647–652. DOI: 10.7873/DATE.2015.0308 (cit. on pp. 25, 71, 84, 88, 104, 126, 142, 143).

[81] O. Willers, C. Huth, J. Guajardo, and H. Seidel. **"MEMS Gyroscopes as Physical Unclonable Functions"**. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. CCS '16. Association for Computing Machinery, 2016, pp. 591–602. ISBN: 9781450341394. DOI: 10.1145/2976749.2978295. URL: https://doi.org/10.1145/2976749.2978295 (cit. on p. 25).

[82] A. Schaller, W. Xiong, N. A. Anagnostopoulos, M. U. Saleem, S. Gabmeyer, S. Katzenbeisser, and J. Szefer. **"Intrinsic Rowhammer PUFs: Leveraging the Rowhammer Effect for Improved Security"**. In: *2017 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 2017, pp. 1–7. DOI: 10.1109/HST.2017.7951729 (cit. on pp. 24, 25, 56, 57, 60, 71, 72, 84, 85, 88, 89, 104, 126, 133, 171).

[83] N. A. Anagnostopoulos, T. Arul, Y. Fan, C. Hatzfeld, J. Lotichius, R. Sharma, F. Fernandes, F. Tehranipoor, and S. Katzenbeisser. **"Securing IoT Devices Using Robust DRAM PUFs"**. In: *2018 Global Information Infrastructure and Networking Symposium (GIIS)*. IEEE, 2018, pp. 1–5. DOI: 10.1109/GIIS.2018.8635789 (cit. on pp. 24, 53, 72, 88, 104, 126, 142, 143).

[84] F. Tehranipoor, N. Karimian, P. A. Wortman, A. Haque, J. Fahrny, and J. A. Chandy. **"Exploring Methods of Authentication for the Internet of Things"**. In: *Internet of Things: Challenges, Advances, and Applications*. Ed. by Q. F. Hassan, A. R. Khan, and S. A. Madani. CRC Press, Chapman & Hall, 2018, pp. 71–90. DOI: 10.1201/9781315155005-4 (cit. on pp. 24, 27).

[85] F. Tehranipoor, N. Karimian, W. Yan, and J. A. Chandy. **"DRAM-Based Intrinsic Physically Unclonable Functions for System-Level Security and Authentication"**. In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 25.3 (2017), pp. 1085–1097. DOI: 10.1109/TVLSI.2016.2606658 (cit. on pp. 24, 71, 80, 84, 85, 88, 104, 126, 142, 143, 169).

[86] W. Xiong, A. Schaller, N. A. Anagnostopoulos, M. U. Saleem, S. Gabmeyer, S. Katzenbeisser, and J. Szefer. **"Run-Time Accessible DRAM PUFs in Commodity Devices"**. In: *Cryptographic Hardware and Embedded Systems – CHES 2016*. Ed. by B. Gierlichs and A. Y. Poschmann. Vol. 9813. Lecture Notes in Computer Science (LNCS). Springer, 2016, pp. 432–453. ISBN: 9783662531402. DOI: 10.1007/978-3-662-53140-2_21 (cit. on pp. 24, 47, 53, 71, 72, 84, 85, 88, 89, 104, 126, 133, 134, 135, 136, 169, 171, 172, 173).

[87] A. Schaller, W. Xiong, N. A. Anagnostopoulos, M. U. Saleem, S. Gabmeyer, B. Škorić, S. Katzenbeisser, and J. Szefer. **"Decay-Based DRAM PUFs in Commodity Devices"**. In: *IEEE Transactions on Dependable and Secure Computing* 16.3 (2019), pp. 462–475. DOI: 10.1109/TDSC.2018.2822298 (cit. on pp. 24, 50, 53, 54, 58, 71, 72, 84, 85, 88, 99, 104, 126, 136, 155, 164, 169, 171, 172, 173, 174).

[88] J. S. Kim, M. Patel, H. Hassan, and O. Mutlu. **"The DRAM Latency PUF: Quickly Evaluating Physical Unclonable Functions by Exploiting the Latency-Reliability Tradeoff in Modern Commodity DRAM Devices"**. In: *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2018, pp. 194–207. DOI: 10.1109/HPCA.2018.00026 (cit. on pp. 24, 61, 64, 71, 84, 85, 88, 104, 126).

[89] Y. Wang, W. Yu, S. Wu, G. Malysa, G. E. Suh, and E. C. Kan. **"Flash Memory for Ubiquitous Hardware Security Functions: True Random Number Generation and Device Fingerprints"**. In: *2012 IEEE Symposium on Security and Privacy*. IEEE, 2012, pp. 33–47. DOI: 10.1109/SP.2012.12 (cit. on pp. 25, 71, 100, 105, 137).

[90] S. T. C. Konigsmark, L. K. Hwang, D. Chen, and M. D. F. Wong. **"CNPUF: A Carbon Nanotube-Based Physically Unclonable Function for Secure Low-Energy Hardware Design"**. In: *2014 19th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2014, pp. 73–78. DOI: 10.1109/ASPDAC.2014.6742869 (cit. on p. 25).

[91] M. Kheir, H. Kreft, and R. Knöchel. **"UWB On-Chip Fingerprinting and Identification Using Carbon Nanotubes"**. In: *2014 IEEE International Conference on Ultra-WideBand (ICUWB)*. IEEE, 2014, pp. 462–466. DOI: 10.1109/ICUWB.2014.6959026 (cit. on p. 25).

[92] G. S. Rose, N. McDonald, L. Yan, B. Wysocki, and K. Xu. **"Foundations of Memristor Based PUF Architectures"**. In: *2013 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*. IEEE, 2013, pp. 52–57. DOI: 10.1109/NanoArch.2013.6623044 (cit. on p. 25).

[93] O. Kavehei, C. Hosung, D. Ranasinghe, and S. Skafidas. **"mrPUF: A Memristive Device Based Physical Unclonable Function"**. 2013. arXiv: 1302.2191 [cond-mat.other] (cit. on p. 25).

[94] G. S. Rose, N. McDonald, L. Yan, and B. Wysocki. **"A Write-Time Based Memristive PUF for Hardware Security Applications"**. In: *2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2013, pp. 830–833. DOI: 10.1109/ICCAD.2013.6691209 (cit. on p. 25).

[95] L. Zhang, X. Fong, C. Chang, Z. H. Kong, and K. Roy. **"Feasibility Study of Emerging Non-Volatile Memory Based Physical Unclonable Functions"**. In: *2014 IEEE 6th International Memory Workshop (IMW)*. IEEE, 2014, pp. 1–4. DOI: 10.1109/IMW.2014.6849384 (cit. on p. 25).

[96] J. Das, K. Scott, D. Burgett, S. Rajaram, and S. Bhanja. **"A Novel Geometry Based MRAM PUF"**. In: *14th IEEE International Conference on Nanotechnology*. IEEE, 2014, pp. 859–863. DOI: 10.1109/NANO.2014.6968027 (cit. on p. 25).

[97] L. Zhang, X. Fong, C. Chang, Z. H. Kong, and K. Roy. **"Highly Reliable Memory-Based Physical Unclonable Function Using Spin-Transfer Torque MRAM"**. In: *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2014, pp. 2169–2172. DOI: 10.1109/ISCAS.2014.6865598 (cit. on p. 25).

[98] T. Marukame, T. Tanamoto, and Y. Mitani. **"Extracting Physically Unclonable Function From Spin Transfer Switching Characteristics in Magnetic Tunnel Junctions"**. In: *IEEE Transactions on Magnetics* 50.11 (2014), pp. 1–4. DOI: 10.1109/TMAG.2014.2325646 (cit. on p. 25).

[99] B. Halak, M. Zwolinski, and M. S. Mispan. **"Overview of PUF-Based Hardware Security Solutions for the Internet of Things"**. In: *2016 IEEE 59th International Midwest Symposium on Circuits and Systems (MWSCAS)*. IEEE, 2016, pp. 1–4. DOI: 10.1109/MWSCAS.2016.7870046 (cit. on pp. 26, 175).

[100] M. A. Feki, F. Kawsar, M. Boussard, and L. Trappeniers. **"The Internet of Things: The Next Technological Revolution"**. In: *Computer* 46.2 (2013), pp. 24–25. ISSN: 0018-9162. DOI: 10.1109/MC.2013.63 (cit. on pp. 26, 175).

[101] M. N. Aman, K. C. Chua, and B. Sikdar. **"Position Paper: Physical Unclonable Functions for IoT Security"**. In: *Proceedings of the 2nd ACM International Workshop on IoT Privacy, Trust, and Security*. IoTPTS '16. Association for Computing Machinery, 2016, pp. 10–13. ISBN: 9781450342834. DOI: 10.1145/2899007.2899013. URL: http://doi.acm.org/10.1145/2899007.2899013 (cit. on p. 26).

[102] T. Idriss, H. Idriss, and M. Bayoumi. **"A PUF-Based Paradigm for IoT Security"**. In: *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*. IEEE, 2016, pp. 700–705. DOI: 10.1109/WF-IoT.2016.7845456 (cit. on p. 26).

[103] J. R. Wallrabenstein. **"Practical and Secure IoT Device Authentication Using Physical Unclonable Functions"**. In: *2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud)*. IEEE, 2016, pp. 99–106. DOI: 10.1109/FiCloud.2016.22 (cit. on p. 27).

[104] J. R. Wallrabenstein. **"Implementing Authentication Systems Based on Physical Unclonable Functions"**. In: *2015 IEEE Trustcom/BigDataSE/ISPA*. Vol. 1. IEEE, 2015, pp. 790–796. DOI: 10.1109/Trustcom.2015.448 (cit. on p. 27).

[105] D. Mukhopadhyay. **"PUFs as Promising Tools for Security in Internet of Things"**. In: *IEEE Design & Test* 33.3 (2016), pp. 103–115. ISSN: 2168-2356. DOI: 10.1109/MDAT.2016.2544845 (cit. on p. 27).

[106] F. Tehranipoor, N. Karimian, P. A. Wortman, and J. A. Chandy. **"Low-Cost Authentication Paradigm for Consumer Electronics Within the Internet of Wearable Fitness Tracking Applications"**. In: *2018 IEEE International Conference on Consumer Electronics (ICCE)*. IEEE, 2018, pp. 1–6. DOI: 10.1109/ICCE.2018.8326233 (cit. on p. 27).

[107] F. Tehranipoor, N. Karimian, P. A. Wortman, and J. A. Chandy. **"Investigation of the Internet of Things in Its Application to Low-Cost Authentication Within Healthcare"**. In: *2017 IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI)*. IEEE, 2017. URL: https://www.researchgate.net/publication/320335572_Investigation_of_the_Internet_of_Things_in_its_Application_to_Low-Cost_Authentication_within_Healthcare (cit. on p. 27).

[108] P. A. Wortman, F. Tehranipoor, N. Karimian, and J. A. Chandy. **"Proposing a Modeling Framework for Minimizing Security Vulnerabilities in IoT Systems in the Healthcare Domain"**. In: *2017 IEEE EMBS International Conference on Biomedical Health Informatics (BHI)*. IEEE, 2017, pp. 185–188. DOI: 10.1109/BHI.2017.7897236 (cit. on p. 27).

[109] N. Karimian, P. A. Wortman, and F. Tehranipoor. **"Evolving Authentication Design Considerations for the Internet of Biometric Things (IoBT)"**. In: *Proceedings of the 11th IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*. CODES+ISSS '16. Association for Computing Machinery, 2016. ISBN: 9781450344838. DOI: 10.1145/2968456.2973748. URL: https://doi.org/10.1145/2968456.2973748 (cit. on p. 27).

[110] N. Karimian, Z. Guo, F. Tehranipoor, D. L. Woodard, M. Tehranipoor, and D. Forte. **"Secure and Reliable Biometric Access Control for Resource-Constrained Systems and IoT"**. 2018. arXiv: 1803.09710 [cs.CR] (cit. on p. 27).

[111] F. Tehranipoor. **"Towards Implementation of Robust and Low-Cost Security Primitives for Resource-Constrained IoT Devices"**. 2018. arXiv: 1806.05332 [cs.CR] (cit. on p. 27).

[112] U. Chatterjee, R. S. Chakraborty, and D. Mukhopadhyay. **"A PUF-Based Secure Communication Protocol for IoT"**. In: *ACM Transactions on Embedded Computing Systems* 16.3 (2017). ISSN: 1539-9087. DOI: 10.1145/3005715. URL: http://doi.acm.org/10.1145/3005715 (cit. on p. 27).

[113] C. Lipps, A. Weinand, D. Krummacker, C. Fischer, and H. D. Schotten. **"Proof of Concept for IoT Device Authentication Based on SRAM PUFs Using ATMEGA 2560-MCU"**. In: *2018 1st International Conference on Data Intelligence and Security (ICDIS)*. IEEE, 2018, pp. 36–42. DOI: 10.1109/ICDIS.2018.00013 (cit. on pp. 28, 84, 85, 104, 105, 142).

[114] M. H. Mahalat, S. Saha, A. Mondal, and B. Sen. **"A PUF Based Light Weight Protocol for Secure WiFi Authentication of IoT devices"**. In: *2018 8th International Symposium on Embedded Computing and System Design (ISED)*. IEEE, 2018, pp. 183–187. DOI: 10.1109/ISED.2018.8703993 (cit. on p. 28).

[115] L. Zhang, Z. H. Kong, C. Chang, A. Cabrini, and G. Torelli. **"Exploiting Process Variations and Programming Sensitivity of Phase Change Memory for Reconfigurable Physical Unclonable Functions"**. In: *IEEE Transactions on Information Forensics and Security* 9.6 (2014), pp. 921–932. ISSN: 1556-6013. DOI: 10.1109/TIFS.2014.2315743 (cit. on pp. 29, 30).

[116] S. S. Zalivaka, L. Zhang, V. P. Klybik, A. A. Ivaniuk, and C.-H. Chang. **"Design and Implementation of High-Quality Physical Unclonable Functions for Hardware-Oriented Cryptography"**. In: *Secure System Design and Trustable Computing*. Ed. by C.-H. Chang and M. Potkonjak. Springer International Publishing, 2016, pp. 39–81. ISBN: 9783319149714. DOI: 10.1007/978-3-319-14971-4_2. URL: https://doi.org/10.1007/978-3-319-14971-4_2 (cit. on p. 29).

[117] S. Meguerdichian and M. Potkonjak. **"Device Aging-Based Physically Unclonable Functions"**. In: *Proceedings of the 48th Design Automation Conference*. DAC '11. Association for Computing Machinery, 2011, pp. 288–289. ISBN: 9781450306362. DOI: 10.1145/2024724.2024793. URL: https://doi.org/10.1145/2024724.2024793 (cit. on p. 29).

[118] I. Eichhorn, P. Koeberl, and V. van der Leest. **"Logically Reconfigurable PUFs: Memory-Based Secure Key Storage"**. In: *Proceedings of the 6th ACM Workshop on Scalable Trusted Computing*. STC '11. Association for Computing Machinery, 2011, pp. 59–64. ISBN: 9781450310017. DOI: 10.1145/2046582.2046594. URL: https://doi.org/10.1145/2046582.2046594 (cit. on pp. 30, 184).

[119] Y. Lao and K. K. Parhi. **"Reconfigurable Architectures for Silicon Physical Unclonable Functions"**. In: *2011 IEEE International Conference on Electro/Information Technology*. IEEE, 2011, pp. 1–7. DOI: 10.1109/EIT.2011.5978614 (cit. on p. 30).

[120] L. Zhang, Z. H. Kong, and C. Chang. **"PCKGen: A Phase Change Memory Based Cryptographic Key Generator"**. In: *2013 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2013, pp. 1444–1447. DOI: 10.1109/ISCAS.2013.6572128 (cit. on p. 30).

[121] Y. Gao, H. Ma, S. F. Al-Sarawi, D. Abbott, and D. C. Ranasinghe. **"PUF-FSM: A Controlled Strong PUF"**. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 37.5 (2018), pp. 1104–1108. DOI: 10.1109/TCAD.2017.2740297 (cit. on p. 30).

[122] J. Zhang, Y. Lin, and G. Qu. **"Reconfigurable Binding against FPGA Replay Attacks"**. In: *ACM Transactions on Design Automation of Electronic Systems* 20.2 (2015). ISSN: 1084–4309. DOI: 10.1145/2699833. URL: https://doi.org/10.1145/2699833 (cit. on p. 30).

[123] W. Liu, L. Zhang, Z. Zhang, C. Gu, C. Wang, M. O'Neill, and F. Lombardi. **"XOR-Based Low-Cost Reconfigurable PUFs for IoT Security"**. In: *ACM Transactions on Embedded Computing Systems* 18.3 (2019). ISSN: 1539-9087. DOI: 10.1145/3274666. URL: https://doi.org/10.1145/3274666 (cit. on p. 30).

[124] Zihan Pang, J. Zhang, Q. Zhou, Shuqian Gong, Xu Qian, and B. Tang. **"Crossover Ring Oscillator PUF"**. In: *2017 18th International Symposium on Quality Electronic Design (ISQED)*. IEEE, 2017, pp. 237–243. DOI: 10.1109/ISQED.2017.7918322 (cit. on p. 30).

[125] V. P. Yanambaka, S. P. Mohanty, E. Kougianos, P. Sundaravadivel, and J. Singh. **"Reconfigurable Robust Hybrid Oscillator Arbiter PUF for IoT Security Based on DL-FET"**. In: *2017 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 2017, pp. 665–670. DOI: 10.1109/ISVLSI.2017.121 (cit. on p. 30).

[126] E. Dubrova. **"A Reconfigurable Arbiter PUF with 4 x 4 Switch Blocks"**. In: *2018 IEEE 48th International Symposium on Multiple-Valued Logic (ISMVL)*. IEEE, 2018, pp. 31–37. DOI: 10.1109/ISMVL.2018.00014 (cit. on p. 30).

[127] Y. Cui, C. Wang, W. Liu, and M. O'Neill. **"A Reconfigurable Memory PUF Based on Tristate Inverter Arrays"**. In: *2016 IEEE International Workshop on Signal Processing Systems (SiPS)*. IEEE, 2016, pp. 171–176. DOI: 10.1109/SiPS.2016.38 (cit. on p. 30).

[128] R. Horstmeyer, S. Assawaworrarit, U. Rührmair, and C. Yang. **"Physically Secure and Fully Reconfigurable Data Storage Using Optical Scattering"**. In: *2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 2015, pp. 157–162. DOI: 10.1109/HST.2015.7140255 (cit. on p. 30).

[129] C. Jin, W. Burleson, M. van Dijk, and U. Rührmair. **"Erasable PUFs: Formal Treatment and Generic Design"**. In: *Proceedings of the 4th ACM Workshop on Attacks and Solutions in Hardware Security*. Association for Computing Machinery, 2020, pp. 21–33. ISBN: 9781450380904. URL: https://doi.org/10.1145/3411504.3421215 (cit. on p. 30).

[130] S. Sutar, A. Raha, and V. Raghunathan. **"Memory-Based Combination PUFs for Device Authentication in Embedded Systems"**. In: *IEEE Transactions on Multi-Scale Computing Systems* 4.4 (2018), pp. 793–810. DOI: 10.1109/TMSCS.2018.2885758 (cit. on pp. 30, 71, 80, 84, 184, 186).

[131] H. Gu and M. Potkonjak. **"Securing Interconnected PUF Network With Reconfigurability"**. In: *2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 2018, pp. 231–234. DOI: 10.1109/HST.2018.8383921 (cit. on p. 30).

[132] S. Schurig. **"Development of a User Interface and Implementation of Specific Software Tools for the Evaluation and Realization of PUFs With Respect to Security Applications"**. Master's Thesis. Technische Universität Darmstadt, 2017 (cit. on p. 36).

[133] S. Schurig. **"Public-Key Kryptographie mit Hilfe von SRAM PUFs"**. Bachelor's Thesis. Technische Universität Darmstadt, 2014 (cit. on pp. 36, 121).

[134] W. Yan, F. Tehranipoor, and J. A. Chandy. **"A Novel Way to Authenticate Untrusted Integrated Circuits"**. In: *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2015, pp. 132–138. DOI: 10.1109/ICCAD.2015.7372560 (cit. on pp. 36, 168).

[135] W. Yan, F. Tehranipoor, and J. A. Chandy. **"PUF-Based Fuzzy Authentication Without Error Correcting Codes"**. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 36.9 (2017), pp. 1445–1457. DOI: 10.1109/TCAD.2016.2638445 (cit. on pp. 36, 168).

[136] H. Handschuh. **"Hardware-Anchored Security Based on SRAM PUFs, Part 1"**. In: *IEEE Security & Privacy* 10.3 (2012), pp. 80–83. ISSN: 1540-7993. DOI: 10.1109/MSP.2012.68 (cit. on pp. 36, 87, 99, 103, 120, 121).

[137] H. Handschuh. **"Hardware-Anchored Security Based on SRAM PUFs, Part 2"**. In: *IEEE Security & Privacy* 10.4 (2012), pp. 80–81. ISSN: 1540-7993. DOI: 10.1109/MSP.2012.98 (cit. on pp. 36, 87, 99, 103, 168).

[138] Y. Dodis, L. Reyzin, and A. Smith. **"Fuzzy Extractors: How to Generate Strong Keys From Biometrics and Other Noisy Data"**. In: *Advances in Cryptology – Eurocrypt 2004*. Ed. by C. Cachin and J. L. Camenisch. Vol. 3027. Lecture Notes in Computer Science (LNCS). Springer, 2004, pp. 523–540. ISBN: 9783540246763. DOI: 10.1007/978-3-540-24676-3_31 (cit. on pp. 36, 87, 99, 103, 120, 168).

[139] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith. **"Fuzzy Extractors: How to Generate Strong Keys From Biometrics and Other Noisy Data"**. In: *SIAM Journal on Computing* 38.1 (2008), pp. 97–139. DOI: 10.1137/060651380 (cit. on pp. 36, 87, 99, 103, 120, 168).

[140] P. Koeberl, J. Li, R. Maes, A. Rajan, C. Vishik, and M. Wójcik. **"Evaluation of a PUF Device Authentication Scheme on a Discrete 0.13um SRAM"**. In: *Trusted Systems – INTRUST 2011*. Ed. by L. Chen, M. Yung, and L. Zhu. Vol. 7222. Lecture Notes in Computer Science (LNCS). Springer, 2012, pp. 271–288. ISBN: 9783642322983. DOI: 10.1007/978-3-642-32298-3_18 (cit. on p. 37).

[141] F. Tehranipoor. **"Design and Architecture of Hardware-Based Random Function Security Primitives"**. Ph.D. dissertation. University of Connecticut (UConn), 2017. URL: http://opencommons.uconn.edu/dissertations/1512 (cit. on pp. 37, 71, 80, 169).

[142] I. Kumari, M. Oh, Y. Kang, and D. Choi. **"Rapid Run-Time DRAM PUF Based on Bit-Flip Position for Secure IoT Devices"**. In: *2018 IEEE SENSORS*. 2018, pp. 1–4. DOI: 10.1109/ICSENS.2018.8589608 (cit. on p. 37).

[143] N. A. Anagnostopoulos, T. Arul, Y. Fan, C. Hatzfeld, A. Schaller, W. Xiong, M. Jain, M. U. Saleem, J. Lotichius, S. Gabmeyer, J. Szefer, and S. Katzenbeisser. **"Intrinsic Run-Time Row Hammer PUFs: Leveraging the Row Hammer Effect for Run-Time Cryptography and Improved Security"**. In: *Cryptography* 2.3 (2018). ISSN: 2410-387X. DOI: 10.3390/cryptography2030013. URL: https://www.mdpi.com/2410-387X/2/3/13 (cit. on pp. 37, 56, 57, 60, 71, 72, 85, 88, 89, 104, 126, 130, 131, 136, 171, 172, 181).

[144] B. M. S. Bahar Talukder, B. Ray, D. Forte, and M. T. Rahman. **"PreLatPUF: Exploiting DRAM Latency Variations for Generating Robust Device Signatures"**. In: *IEEE Access* 7 (2019), pp. 81106–81120. DOI: 10.1109/ACCESS.2019.2923174 (cit. on pp. 37, 61, 64, 85, 88, 104, 126, 142, 143).

[145] N. A. Anagnostopoulos, S. Ahmad, T. Arul, D. Steinmetzer, M. Hollick, and S. Katzenbeisser. **"Low-Cost Security for Next-Generation IoT Networks"**. In: *ACM Transactions on Internet Technology* 20.3 (2020). ISSN: 1533-5399. DOI: 10.1145/3406280. URL: https://doi.org/10.1145/3406280 (cit. on pp. 39, 85, 104, 105, 172, 176).

[146] F. Kohnhäuser, A. Schaller, and S. Katzenbeisser. **"PUF-Based Software Protection for Low-End Embedded Devices"**. In: *Trust and Trustworthy Computing – Trust 2015*. Ed. by M. Conti, M. Schunter, and I. Askoxylakis. Vol. 9229. Lecture Notes in Computer Science (LNCS). Springer, 2015, pp. 3–21. ISBN: 9783319228464. DOI: 10.1007/978-3-319-22846-4_1 (cit. on pp. 40, 85, 110, 168).

[147] N. A. Anagnostopoulos, Y. Fan, T. Arul, R. Sarangdhar, and S. Katzenbeisser. **"Lightweight Security Solutions for IoT Implementations in Space"**. In: *2019 IEEE Topical Workshop on Internet of Space (TWIOS)*. IEEE, 2019, pp. 1–4. DOI: 10.1109/TWIOS.2019.8771257 (cit. on pp. 40, 53, 70, 88, 104, 105, 126, 137, 165, 172).

[148] N. A. Anagnostopoulos, T. Arul, Y. Fan, M. Kumar, and S. Katzenbeisser. **"AR-PUFs: Advanced Security Primitives for the Internet of Things and Cyber-Physical Systems"**. In: *2019 IEEE International Conference on Consumer Electronics (ICCE)*. IEEE, 2019, pp. 1–5. DOI: 10.1109/ICCE.2019.8661840 (cit. on pp. 43, 54, 60, 65, 71, 172).

[149] P. Jaccard. **"Méthode Statistique pour Déterminer la Distribution de la Flore Alpine"**. In: *Actes du 1er Congrès International de Botanique (tenu à Paris à l' occasion de l' Exposition Universelle de 1900)*. Ed. by É. Perrot. Comptes-Rendus du Congrès International de Botanique (1900): Premiére Partie – Communications Scientifiques, Chapitre 1e – Biologie, Morphologie et Physiologie générales. Lucien Declume, Lons-le-Saunier, France, 1900, pp. 31–38. URL: https://gallica.bnf.fr/ark:/12148/bpt6k63514215/f79. *In French.* (Cit. on pp. 43, 52, 56, 59, 74).

[150] P. Jaccard. **"Distribution de la Flore Alpine dans le Bassin des Dranses et dans Quelques Régions Voisines"**. In: *Bulletin de la Societe Vaudoise des Sciences Naturelles*. Ed. by F. Roux. Vol. 37. 140. Corbaz & Comp., Lausanne, Switzerland, 1901, pp. 241–272. DOI: 10.5169/seals-266440. URL: https://www.e-periodica.ch/digbib/view?pid=bsv-002:1901:37::745#251. *In French.* (Cit. on pp. 43, 52, 56, 59, 74).

[151] N. A. Anagnostopoulos, A. Schaller, Y. Fan, W. Xiong, F. Tehranipoor, T. Arul, S. Gabmeyer, J. Szefer, J. A. Chandy, and S. Katzenbeisser. **"Insights into the Potential Usage of the Initial Values of DRAM Arrays of Commercial Off-the-Shelf Devices for Security Applications"**. In: *Proceedings of the 26th Crypto-Day, Nuremberg, Germany* (2017), pp. 9–11. DOI: 10.13140/RG.2.2.27089.63849. URL: https://fg-krypto.gi.de/fileadmin/FG/KRYPTO/Proceedings/LN_CryptoDay26_SUSE.pdf#page=9. *Presentation Slides:* DOI: 10.13140/RG.2.2.21396.50569. (Cit. on p. 46).

[152] B. Keeth, R. J. Baker, B. Johnson, and F. Lin. **"DRAM Circuit Design: Fundamental and High-Speed Topics"**. 2nd Edition. Wiley – IEEE Press, 2007. ISBN: 9780470184752. IEEE Press Series on Microelectronic Systems. URL: https://www.wiley.com/en-us/DRAM+Circuit+Design%3A+Fundamental+and+High+Speed+Topics+%2C+2nd+Edition-p-9780470184752 (cit. on pp. 47, 55).

[153] W. Xiong, A. Schaller, N. A. Anagnostopoulos, M. U. Saleem, S. Gabmeyer, S. Katzenbeisser, and J. Szefer. **"DRAM PUFs in Commodity Devices"**. In: *IEEE Design & Test* 38.3 (2021), pp. 76–83. DOI: 10.1109/MDAT.2021.3063370 (cit. on pp. 53, 88, 104, 126).

[154] Алексей Витенко (Aleksey Vitenko). **“Использование DRAM-PUF для Идентификации Мобильных Устройств под Управлением ОС Android (Using DRAM PUFs to Identify Mobile Devices Managed by the Android OS)”**. In: Апробация *(Approbation)* 1 (40) (2016), pp. 26–29. ISSN: 2305-4484. URL: https://www.elibrary.ru/item.asp?id=25723744. Also available at http://www.chelsma.ru/files/misc/160101aprobacija.pdf (pp. 25–28). *In Russian.* (Cit. on pp. 54, 71).

[155] M. Micheletti and T. LeCroy. **“Tuning DDR4 for Power and Performance”**. MemCon 2013. URL: http://cdn.teledynelecroy.com/files/whitepapers/tuningddr4_for_power_performance.pdf. *Presentation Slides.* (Cit. on p. 55).

[156] Y. Kim, R. Daly, J. Kim, C. Fallin, J. H. Lee, D. Lee, C. Wilkerson, K. Lai, and O. Mutlu. **“Flipping Bits in Memory without Accessing Them: An Experimental Study of DRAM Disturbance Errors”**. In: *SIGARCH Computer Architecture News* 42.3 (2014), pp. 361–372. ISSN: 0163-5964. DOI: 10.1145/2678373.2665726. URL: https://doi.org/10.1145/2678373.2665726 (cit. on p. 55).

[157] O. Mutlu. **“The RowHammer Problem and Other Issues We May Face as Memory Becomes Denser”**. In: *2017 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2017, pp. 1116–1121. DOI: 10.23919/DATE.2017.7927156 (cit. on p. 55).

[158] A. Schaller, W. Xiong, N. A. Anagnostopoulos, M. U. Saleem, S. Gabmeyer, S. Katzenbeisser, and J. Szefer. **“Intrinsic Rowhammer PUFs: Leveraging the Rowhammer Effect for Improved Security”**. 2019. arXiv: 1902.04444 [cs.CR] (cit. on pp. 56, 57, 60, 72, 85, 104, 126).

[159] J. Miskelly and M. O’Neill. **“Fast DRAM PUFs on Commodity Devices”**. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39.11 (2020), pp. 3566–3576. DOI: 10.1109/TCAD.2020.3012218 (cit. on pp. 61, 64, 85, 88).

[160] H. Gordon, J. Edmonds, S. Ghandali, W. Yan, N. Karimian, and F. Tehranipoor. **“Flash-Based Security Primitives: Evolution, Challenges and Future Directions”**. In: *Cryptography* 5.1 (2021). ISSN: 2410-387X. DOI: 10.3390/cryptography5010007. URL: https://www.mdpi.com/2410-387X/5/1/7 (cit. on p. 65).

[161] S. Jia, L. Xia, Z. Wang, J. Lin, G. Zhang, and Y. Ji. **“Extracting Robust Keys from NAND Flash Physical Unclonable Functions”**. In: *Information Security – ISC 2015*. Ed. by J. Lopez and C. J. Mitchell. Vol. 9290. Lecture Notes in Computer Science (LNCS). Springer, 2015, pp. 437–454. ISBN: 9783319233185. DOI: 10.1007/978-3-319-23318-5_24 (cit. on pp. 69, 85, 100, 105, 137).

[162] S. Sakib, A. Milenković, M. T. Rahman, and B. Ray. **“An Aging-Resistant NAND Flash Memory Physical Unclonable Function”**. In: *IEEE Transactions on Electron Devices* 67.3 (2020), pp. 937–943. DOI: 10.1109/TED.2020.2968272 (cit. on pp. 69, 85, 100, 105, 137).

[163] S. Sutar, A. Raha, and V. Raghunathan. **“D-PUF: An Intrinsically Reconfigurable DRAM PUF for Device Authentication in Embedded Systems”**. In: *2016 International Conference on Compilers, Architectures, and Sythesis of Embedded Systems (CASES)*. IEEE, 2016, pp. 1–10. DOI: 10.1145/2968455.2968519 (cit. on pp. 71, 80, 84, 169).

[164] W. Liu, Z. Zhang, M. Li, and Z. Liu. **“A Trustworthy Key Generation Prototype Based on DDR3 PUF for Wireless Sensor Networks”**. In: *Sensors* 14.7 (2014), pp. 11542–11556. ISSN: 1424-8220. DOI: 10.3390/s140711542. URL: https://www.mdpi.com/1424-8220/14/7/11542 (cit. on pp. 71, 80, 169).

[165] W. Liu, Z. Zhang, M. Li, and Z. Liu. **“A Trustworthy Key Generation Prototype Based on DDR3 PUF for Wireless Sensor Networks”**. In: *2014 International Symposium on Computer, Consumer and Control (IS3C)*. IEEE, 2014, pp. 706–709. DOI: 10.1109/IS3C.2014.188 (cit. on pp. 71, 80, 169).

[166] A. Schaller. **"Lightweight Protocols and Applications for Memory-Based Intrinsic Physically Unclonable Functions Found on Commercial Off-The-Shelf Devices"**. Ph.D. dissertation. Technische Universität Darmstadt, 2017. URL: http://tuprints.ulb.tu-darmstadt.de/7014/ (cit. on pp. 71, 169).

[167] Z. Zhang (张振华). **"Design and Implementation of DRAM PUF — DRAM PUF 的设计与实现"**. M.Sc. thesis. Huazhong University of Science and Technology — 华中科技大学, 2015. URL: http://cdmd.cnki.com.cn/Article/CDMD-10487-1015906333.htm. *In Chinese.* (Cit. on pp. 71, 80, 169).

[168] S. Rosenblatt, D. Fainstein, A. Cestero, J. Safran, N. Robson, T. Kirihata, and S. S. Iyer. **"Field Tolerant Dynamic Intrinsic Chip ID Using 32 nm High-K/Metal Gate SOI Embedded DRAM"**. In: *IEEE Journal of Solid-State Circuits* 48.4 (2013), pp. 940–947. DOI: 10.1109/JSSC.2013.2239134 (cit. on p. 71).

[169] S. Rosenblatt, S. Chellappa, A. Cestero, N. Robson, T. Kirihata, and S. S. Iyer. **"A Self-Authenticating Chip Architecture Using an Intrinsic Fingerprint of Embedded DRAM"**. In: *IEEE Journal of Solid-State Circuits* 48.11 (2013), pp. 2934–2943. DOI: 10.1109/JSSC.2013.2282114 (cit. on pp. 71, 169).

[170] R. Kumar, X. Xu, W. Burleson, S. Rosenblatt, and T. Kirihata. **"Physically Unclonable Functions: A Window into CMOS Process Variations"**. In: *Circuits and Systems for Security and Privacy*. Ed. by F. Sheikh, L. Sousa, and K. Iniewski. CRC Press, 2017. DOI: 10.1201/b19499-13 (cit. on pp. 71, 169).

[171] T. Kirihata and S. Rosenblatt. **"Dynamic Intrinsic Chip ID for Hardware Security"**. In: *VLSI: Circuits for Emerging Applications*. Ed. by T. Wojcicki and K. Iniewski. CRC Press, 2017. DOI: 10.1201/b17635 (cit. on pp. 71, 169).

[172] C. Keller, F. Gurkaynak, H. Kaeslin, and N. Felber. **"Dynamic Memory-Based Physically Unclonable Function for the Generation of Unique Identifiers and True Random Numbers"**. In: *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2014, pp. 2740–2743. DOI: 10.1109/ISCAS.2014.6865740 (cit. on pp. 71, 169).

[173] S. Sutar, A. Raha, D. Kulkarni, R. Shorey, J. Tew, and V. Raghunathan. **"D-PUF: An Intrinsically Reconfigurable DRAM PUF for Device Authentication and Random Number Generation"**. In: *ACM Transactions on Embedded Computing Systems* 17.1 (2017). ISSN: 1539-9087. DOI: 10.1145/3105915. URL: https://doi.org/10.1145/3105915 (cit. on pp. 71, 80, 84, 169).

[174] F. Tehranipoor, N. Karimian, W. Yan, and J. A. Chandy. **"Investigation of DRAM PUFs Reliability Under Device Accelerated Aging Effects"**. In: *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2017, pp. 1–4. DOI: 10.1109/ISCAS.2017.8050629 (cit. on pp. 71, 84, 169).

[175] A. Rahmati, M. Hicks, D. E. Holcomb, and K. Fu. **"Probable Cause: The Deanonymizing Effects of Approximate DRAM"**. In: *2015 ACM/IEEE 42nd Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2015, pp. 604–615 (cit. on p. 71).

[176] M. M. Deza and E. Deza. **"Encyclopedia of Distances"**. 4th Edition. Springer, 2016. ISBN: 9783662528433. DOI: 10.1007/978-3-662-52844-0 (cit. on p. 71).

[177] N. Felber (for N. Gisin). **"Secure High-Speed Communication based on Quantum Key Distribution"**. Swiss National Science Foundation (SNSF), Nano-Tera.CH, RTD 2010 – QCrypt, Annual Plenary Meeting. 2012. URL: https://www.scribd.com/presentation/92684363/qcrypt. *Presentation Slides.* (Cit. on p. 71).

[178] R. Canetti and M. Fischlin. **"Universally Composable Commitments"**. In: *Advances in Cryptology – CRYPTO 2001*. Ed. by J. Kilian. Vol. 2139. Lecture Notes in Computer Science (LNCS). Springer, 2001, pp. 19–40. ISBN: 9783540446477. DOI: 10.1007/3-540-44647-8_2 (cit. on pp. 77, 179).

[179] N. A. Anagnostopoulos. **"Optical Fault Injection Attacks in Smart Card Chips and an Evaluation of Countermeasures Against Them"**. M.Sc. thesis. University of Twente, 2014. URL: https://essay.utwente.nl/66014/. Also available at https://essay.utwente.nl/66028/. (Cit. on pp. 77, 78, 166, 179).

[180] W. Rankl and W. Effing. **"Smart Card Security"**. In: *Smart Card Handbook*. John Wiley & Sons, 2010. Chap. 16, pp. 667–734. ISBN: 9780470660911. DOI: 10.1002/9780470660911.ch16. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9780470660911.ch16. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/9780470660911.ch16 (cit. on p. 78).

[181] G. Gianfelici, H. Kampermann, and D. Bruß. **"A Theoretical Framework for PUFs and QR-PUFs"**. 2019. arXiv: 1911.04981 [quant-ph] (cit. on pp. 79, 180).

[182] G. Gianfelici, H. Kampermann, and D. Bruß. **"Theoretical Framework for Physical Unclonable Functions, Including Quantum Readout"**. In: *Physical Review A* 101.4 (2020). DOI: 10.1103/PhysRevA.101.042337. URL: https://link.aps.org/doi/10.1103/PhysRevA.101.042337 (cit. on pp. 79, 180).

[183] F. Hetherton. **"Security Analysis of Identification Protocols Based on Quantum Physical Unclonable Functions"**. M.Sc. thesis. University of Edinburgh, 2020. URL: https://project-archive.inf.ed.ac.uk/msc/20204134/msc_proj.pdf (cit. on pp. 79, 180).

[184] W. Xiong, N. A. Anagnostopoulos, A. Schaller, S. Katzenbeisser, and J. Szefer. **"Spying on Temperature Using DRAM"**. In: *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2019, pp. 13–18. DOI: 10.23919/DATE.2019.8714882 (cit. on p. 84).

[185] S. Katzenbeisser, Ü. Kocabaş, V. Rožić, A.-R. Sadeghi, I. Verbauwhede, and C. Wachsmann. **"PUFs: Myth, Fact or Busted? A Security Evaluation of Physically Unclonable Functions (PUFs) Cast in Silicon"**. In: *Cryptographic Hardware and Embedded Systems – CHES 2012*. Ed. by E. Prouff and P. Schaumont. Vol. 7428. Lecture Notes in Computer Science (LNCS). Springer, 2012, pp. 283–301. ISBN: 9783642330278. DOI: 10.1007/978-3-642-33027-8_17 (cit. on pp. 84, 85, 104, 105, 110, 142).

[186] S. Katzenbeisser, Ü. Kocabaş, V. Rožić, A.-R. Sadeghi, I. Verbauwhede, and C. Wachsmann. **"PUFs: Myth, Fact or Busted? A Security Evaluation of Physically Unclonable Functions (PUFs) Cast in Silicon (Extended Version)"**. Cryptology ePrint Archive, Report 2012/557. 2012. URL: https://ia.cr/2012/557 (cit. on pp. 84, 85, 104, 105, 142).

[187] A. Schaller, B. Škorić, and S. Katzenbeisser. **"On the Systematic Drift of Physically Unclonable Functions Due to Aging"**. In: *Proceedings of the 5th International Workshop on Trustworthy Embedded Devices*. TrustED '15. Association for Computing Machinery, 2015, pp. 15–20. ISBN: 9781450338288. DOI: 10.1145/2808414.2808417. URL: https://doi.org/10.1145/2808414.2808417 (cit. on p. 84).

[188] R. Maes and V. van der Leest. **"Countering the Effects of Silicon Aging on SRAM PUFs"**. In: *2014 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*. IEEE, 2014, pp. 148–153. DOI: 10.1109/HST.2014.6855586 (cit. on p. 84).

[189] T. Arul, N. A. Anagnostopoulos, S. Reißig, and S. Katzenbeisser. **"A Study of the Spatial Auto-Correlation of Memory-Based Physical Unclonable Functions"**. In: *2020 European Conference on Circuit Theory and Design (ECCTD)*. IEEE, 2020, pp. 1–4. DOI: 10.1109/ECCTD49232.2020.9218302 (cit. on p. 84).

[190] F. Wilde, B. M. Gammel, and M. Pehl. **"Spatial Correlation Analysis on Physical Unclonable Functions"**. In: *IEEE Transactions on Information Forensics and Security* 13.6 (2018), pp. 1468–1480. DOI: 10.1109/TIFS.2018.2791341 (cit. on p. 84).

[191] T. Saito, H. Nagase, M. Izuna, T. Shimoi, A. Kanda, T. Ito, and T. Kono. **"High-Temperature Stable Physical Unclonable Functions with Error-Free Readout Scheme Based on 28nm SG-MONOS Flash Memory for Security Applications"**. In: *2017 IEEE International Memory Workshop (IMW)*. IEEE, 2017, pp. 1–4. DOI: 10.1109/IMW.2017.7939086 (cit. on pp. 100, 105, 137).

[192] N. A. Anagnostopoulos, Y. Fan, M. Heinrich, N. Matyunin, D. Püllen, P. Muth, C. Hatzfeld, M. Rosenstihl, T. Arul, and S. Katzenbeisser. **"Low-Temperature Attacks Against Digital Electronics: A Challenge for the Security of Superconducting Modules in High-Speed Magnetic Levitation (MagLev) Trains"**. In: *2021 IEEE 14th Workshop on Low Temperature Electronics (WOLTE)*. IEEE, 2021, pp. 1–4. DOI: 10.1109/WOLTE49037.2021.9555437 (cit. on pp. 104, 105, 109, 126).

[193] M. Mahmoodi, H. Nili, S. Larimian, X. Guo, and D. Strukov. **"ChipSecure: A Reconfigurable Analog eFlash-Based PUF with Machine Learning Attack Resiliency in 55nm CMOS"**. In: *Proceedings of the 56th Annual Design Automation Conference 2019*. DAC '19. Association for Computing Machinery, 2019. ISBN: 9781450367257. DOI: 10.1145/3316781.3324890. URL: https://doi.org/10.1145/3316781.3324890 (cit. on pp. 105, 137, 142, 149).

[194] S. Skorobogatov. **"Low Temperature Data Remanence in Static RAM"**. Tech. rep. UCAM-CL-TR-536. University of Cambridge, Computer Laboratory, 2002. URL: http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-536.pdf (cit. on p. 109).

[195] H. Jiao (焦慧芳), X. Zhang (张小波), X. Jia (贾新章), X. Yang (杨雪莹), and Z. Zhong (钟征宇). **"The Characteristic Study of Data Remanence of SRAM — 静态随机存储器数据残留特性研究"**. In: *Research & Progress of Solid State Electronics (SSE)* — 固体电子学研究与进展 26.4 (2006), pp. 536–539. ISSN: 1000-3819. URL: https://www.cnki.com.cn/Article/CJFDTOTAL-GTDZ200604021.htm. URL (AIRITI): https://www.airitilibrary.com/Publication/alDetailedMesh?docid=10003819-200611-26-4-536-539-a. *In Chinese.* (Cit. on p. 109).

[196] D. Samyde, S. Skorobogatov, R. Anderson, and J.-J. Quisquater. **"On a New Way to Read Data From Memory"**. In: *First International IEEE Security in Storage Workshop 2002*. IEEE, 2002, pp. 65–69. DOI: 10.1109/SISW.2002.1183512 (cit. on p. 109).

[197] T. Tuan, T. Strader, and S. Trimberger. **"Analysis of Data Remanence in a 90nm FPGA"**. In: *IEEE Custom Integrated Circuits Conference (CICC) 2007*. IEEE, 2007, pp. 93–96. DOI: 10.1109/CICC.2007.4405689 (cit. on p. 109).

[198] H.-W. Chen, S. Srinivasan, Y. Xie, and V. Narayanan. **"Impact of Circuit Degradation on FPGA Design Security"**. In: *IEEE Computer Society Annual Symposium on VLSI (ISVLSI) 2011*. IEEE, 2011, pp. 230–235. DOI: 10.1109/ISVLSI.2011.81 (cit. on p. 109).

[199] N. Saxena and J. Voris. **"Data Remanence Effects on Memory-Based Entropy Collection for RFID Systems"**. In: *International Journal of Information Security* 10.4 (2011), pp. 213–222. ISSN: 1615-5270. DOI: 10.1007/s10207-011-0139-0 (cit. on p. 109).

[200] N. Saxena and J. Voris. **"We Can Remember It for You Wholesale: Implications of Data Remanence on the Use of RAM for True Random Number Generation on RFID Tags (RFIDSec 2009)"**. 2009. arXiv: 0907.1256 [cs.CR]. Also presented at the 5th Workshop on RFID Security. (Cit. on p. 109).

[201] C. Cakir, M. Bhargava, and K. Mai. **"6T SRAM and 3T DRAM Data Retention and Remanence Characterization in 65nm Bulk CMOS"**. In: *IEEE Custom Integrated Circuits Conference (CICC) 2012*. IEEE, 2012, pp. 1–4. DOI: 10.1109/CICC.2012.6330672 (cit. on pp. 109, 133).

[202] P. Gutmann. **"Data Remanence in Semiconductor Devices"**. In: *10th USENIX Security Symposium (USENIX Security '01)*. USENIX Association, 2001. URL: https://www.usenix.org/conference/10th-usenix-security-symposium/data-remanence-semiconductor-devices (cit. on pp. 109, 133).

[203] A. Braeken, S. Kubera, F. Trouillez, A. Touhafi, N. Mentens, and J. Vliegen. **"Secure FPGA Technologies and Techniques"**. In: *International Conference on Field Programmable Logic and Applications (FPL) 2009*. IEEE, 2009, pp. 560–563. DOI: 10.1109/FPL.2009.5272414 (cit. on p. 109).

[204] M. Claes, V. van der Leest, and A. Braeken. **"Comparison of SRAM and FF PUF in 65nm Technology"**. In: *Information Security Technology for Applications – NordSec 2011*. Ed. by P. Laud. Vol. 7161. Lecture Notes in Computer Science (LNCS). Springer, 2012, pp. 47–64. ISBN: 9783642296154. DOI: 10.1007/978-3-642-29615-4_5 (cit. on pp. 109, 125).

[205] A. Wild and T. Güneysu. **"Enabling SRAM-PUFs on Xilinx FPGAs"**. In: *24th International Conference on Field Programmable Logic and Applications (FPL) 2014*. IEEE, 2014, pp. 1–4. DOI: 10.1109/FPL.2014.6927384 (cit. on p. 109).

[206] J. A. Halderman, S. D. Schoen, N. Heninger, W. Clarkson, W. Paul, J. A. Calandrino, A. J. Feldman, J. Appelbaum, and E. W. Felten. **"Lest We Remember: Cold-Boot Attacks on Encryption Keys"**. In: *Communications of the ACM* 52.5 (2009), pp. 91–98. ISSN: 0001-0782. DOI: 10.1145/1506409.1506429. URL: https://doi.org/10.1145/1506409.1506429 (cit. on pp. 125, 133).

[207] Y. Kai, Z. Xuecheng, Y. Guoyi, and W. Weixu. **"Security Strategy of Powered-Off SRAM for Resisting Physical Attack to Data Remanence"**. In: *Journal of Semiconductors* 30.9 (2009). URL: http://stacks.iop.org/1674-4926/30/i=9/a=095010 (cit. on p. 125).

[208] K. Wenjing, Y. Kai, Y. Guoyi, and Z. Xuecheng. **"Novel Security Strategies for SRAM in Powered-Off State to Resist Physical Attack"**. In: *Proceedings of the 12th International Symposium on Integrated Circuits (ISIC) 2009*. IEEE, 2009, pp. 298–301 (cit. on p. 125).

[209] L. Zhang, X. Fong, C.-H. Chang, Z. H. Kong, and K. Roy. **"Optimizating Emerging Nonvolatile Memories for Dual-Mode Applications: Data Storage and Key Generator"**. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 34.7 (2015), pp. 1176–1187. ISSN: 0278-0070. DOI: 10.1109/TCAD.2015.2427251 (cit. on p. 125).

[210] D. Vasile, P. Svasta, and M. Pantazică. **"Preventing the Temperature Side Channel Attacks on Security Circuits"**. In: *2019 IEEE 25th International Symposium for Design and Technology in Electronic Packaging (SIITME)*. IEEE, 2019, pp. 244–247. DOI: 10.1109/SIITME47687.2019.8990788 (cit. on p. 125).

[211] P. Colp, J. Zhang, J. Gleeson, S. Suneja, E. de Lara, H. Raj, S. Saroiu, and A. Wolman. **"Protecting Data on Smartphones and Tablets From Memory Attacks"**. In: *Proceedings of the 20th International Conference on Architectural Support for Programming Languages and Operating Systems*. ASPLOS '15. Association for Computing Machinery, 2015, pp. 177–189. ISBN: 9781450328357. DOI: 10.1145/2694344.2694380. URL: https://doi.org/10.1145/2694344.2694380 (cit. on pp. 125, 133).

[212] M. Gruhn and T. Müller. **"On the Practicability of Cold Boot Attacks"**. In: *8th International Conference on Availability, Reliability and Security (ARES) 2013*. IEEE, 2013, pp. 390–397. DOI: 10.1109/ARES.2013.52 (cit. on pp. 125, 133).

[213] O. Kömmerling and M. G. Kuhn. **"Design Principles for Tamper-Resistant Smartcard Processors"**. In: *USENIX Workshop on Smartcard Technology (Smartcard '99)*. USENIX Association, 1999. URL: https://www.usenix.org/conference/usenix-workshop-smartcard-technology/design-principles-tamper-resistant-smartcard (cit. on p. 125).

[214] S.-J. Han, J.-K. Han, G.-J. Yun, M.-W. Lee, J.-M. Yu, and Y.-K. Choi. **"Ultra-Fast Data Sanitization of SRAM by Back-Biasing to Resist a Cold Boot Attack"**. Research Square Platform. 2021. DOI: 10.21203/rs.3.rs-493322/v1. *Preprint.* (Cit. on p. 125).

[215] T Huffmire, S Prasad, T Sherwood, and R Kastner. **"Threats and Challenges in Reconfigurable Hardware Security"**. In: *Proceedings of the 2008 International Conference on Engineering of Reconfigurable Systems & Algorithms (ERSA) 2008*. Ed. by T. P. Plaks. CSREA Press, 2008, pp. 334–345. URL: https://apps.dtic.mil/sti/citations/ADA511928 (cit. on p. 125).

[216] S. Eiroa, J. Castro, M. C. Martinez-Rodriguez, E. Tena, P. Brox, and I. Baturone. **"Reducing Bit Flipping Problems in SRAM Physical Unclonable Functions for Chip Identification"**. In: *19th IEEE International Conference on Electronics, Circuits and Systems (ICECS) 2012*. IEEE, 2012, pp. 392–395. DOI: 10.1109/ICECS.2012.6463720 (cit. on p. 125).

[217] E. M. Chan, J. C. Carlyle, F. M. David, R. Farivar, and R. H. Campbell. **"Bootjacker: Compromising Computers Using Forced Restarts"**. In: *Proceedings of the 15th ACM conference on Computer and Communications Security*. CCS '08. Association for Computing Machinery, 2008, pp. 555–564. ISBN: 9781595938107. DOI: 10.1145/1455770.1455840. URL: https://doi.org/10.1145/1455770.1455840 (cit. on p. 133).

[218] T. Müller and M. Spreitzenbarth. **"FROST: Forensic Recovery of Scrambled Telephones"**. In: *Applied Cryptography and Network Security – ACNS 2013*. Ed. by M. Jacobson, M. Locasto, P. Mohassel, and R. Safavi-Naini. Vol. 7954. Lecture Notes in Computer Science (LNCS). Springer, 2013, pp. 373–388. ISBN: 9783642389801. DOI: 10.1007/978-3-642-38980-1_23 (cit. on p. 133).

[219] J. Bauer, M. Gruhn, and F. C. Freiling. **"Lest We Forget: Cold-Boot Attacks on Scrambled DDR3 Memory"**. In: *Digital Investigation* 16 (2016). Proceedings of the Third Annual DFRWS Europe — DFRWS 2016 Europe, S65–S74. ISSN: 1742-2876. DOI: 10.1016/j.diin.2016.01.009. URL: https://www.sciencedirect.com/science/article/pii/S1742287616300032 (cit. on p. 133).

[220] T. R. Oldham, R. L. Ladbury, M. Friendlich, H. S. Kim, M. D. Berg, T. L. Irwin, C. Seidleck, and K. A. LaBel. **"SEE and TID Characterization of an Advanced Commercial 2Gbit NAND Flash Nonvolatile Memory"**. In: *IEEE Transactions on Nuclear Science* 53.6 (2006), pp. 3217–3222. ISSN: 0018-9499. DOI: 10.1109/TNS.2006.885843 (cit. on p. 165).

[221] T. R. Oldham, M. Friendlich, J. W. Howard, M. D. Berg, H. S. Kim, T. L. Irwin, and K. A. LaBel. **"TID and SEE Response of an Advanced Samsung 4Gb NAND Flash Memory"**. In: *2007 IEEE Radiation Effects Data Workshop*. IEEE, 2007, pp. 221–225. DOI: 10.1109/REDW.2007.4342570 (cit. on pp. 165, 181).

[222] L. Z. Scheick, S. M. Guertin, and G. M. Swift. **"Analysis of Radiation Effects on Individual DRAM Cells"**. In: *IEEE Transactions on Nuclear Science* 47.6 (2000), pp. 2534–2538. ISSN: 0018-9499. DOI: 10.1109/23.903804 (cit. on p. 165).

[223] R. Sarangdhar, Y. Fan, N. A. Anagnostopoulos, U. Gayer, F. Flederer, T. Mikschl, T. Arul, P. John, K. Hierholz, S. Montenegro, and S. Katzenbeisser. **"An Investigation of the Effects of Radiation on Current Key Storage Solutions and on Physical Unclonable Functions (PUFs) Being Used as Key Storage"**. Proceedings of the 27th Crypto-Day, Frankfurt (Oder), Germany. 2017. DOI: 10.13140/RG.2.2.33800.52483 (cit. on p. 165).

[224] M. Dorin, S. Janardhanan, and S. Montenegro. **"Software Streamlining: Reducing Software to Essentials"**. In: *2021 IEEE International Conference on Automation/XXIV Congress of the Chilean Association of Automatic Control (ICA-ACCA)*. IEEE, 2021, pp. 1–4. DOI: 10.1109/ICAACCA51523.2021.9465176 (cit. on p. 165).

[225] M. Faisal, E. Dilger, and S. Montenegro. **"A Unified Approach for Memory Protection in a Bare-Metal and a Real Time Operating System"**. In: *Proceedings of the 23rd Pan-Hellenic Conference on Informatics*. PCI '19. Association for Computing Machinery, 2019, pp. 149–152. ISBN: 9781450372923. DOI: 10.1145/3368640.3368667. URL: https://doi.org/10.1145/3368640.3368667 (cit. on p. 165).

[226] S. Schulz, A. Schaller, F. Kohnhäuser, and S. Katzenbeisser. **"Boot Attestation: Secure Remote Reporting with Off-The-Shelf IoT Sensors"**. In: *Computer Security – ESORICS 2017*. Ed. by S. N. Foley, D. Gollmann, and E. Snekkenes. Vol. 10493. Lecture Notes in Computer Science (LNCS). Springer, 2017, pp. 437–455. ISBN: 9783319663999. DOI: 10.1007/978-3-319-66399-9_24 (cit. on p. 168).

[227] S. Chen, B. Li, and C. Zhou. **"FPGA Implementation of SRAM PUFs Based Cryptographically Secure Pseudo-Random Number Generator"**. In: *Microprocessors and Microsystems* 59 (2018), pp. 57–68. ISSN: 0141-9331. DOI: 10.1016/j.micpro.2018.02.001 (cit. on p. 168).

[228] M. Yue, N. Karimian, W. Yan, N. A. Anagnostopoulos, and F. Tehranipoor. **"DRAM-Based Authentication using Deep Convolutional Neural Networks"**. In: *IEEE Consumer Electronics Magazine* 10.4 (2021), pp. 8–17. DOI: 10.1109/MCE.2020.3002528 (cit. on p. 171).

[229] N. A. Anagnostopoulos, S. Katzenbeisser, J. Chandy, and F. Tehranipoor. **"An Overview of DRAM-Based Security Primitives"**. In: *Cryptography* 2.2 (2018). ISSN: 2410-387X. DOI: 10.3390/cryptography2020007. URL: https://www.mdpi.com/2410-387X/2/2/7 (cit. on pp. 172, 173).

[230] D. Püllen, N. A. Anagnostopoulos, T. Arul, and S. Katzenbeisser. **"Poster: Hierarchical Integrity Checking in Heterogeneous Vehicular Networks"**. In: *2018 IEEE Vehicular Networking Conference (VNC)*. IEEE, 2018, pp. 1–2. DOI: 10.1109/VNC.2018.8628375 (cit. on p. 172).

[231] D. Püllen, N. A. Anagnostopoulos, T. Arul, and S. Katzenbeisser. **"Using Implicit Certification to Efficiently Establish Authenticated Group Keys for In-Vehicle Networks"**. In: *2019 IEEE Vehicular Networking Conference (VNC)*. 2019, pp. 1–8. DOI: 10.1109/VNC48660.2019.9062785 (cit. on p. 172).

[232] L. Negka, G. Gketsios, N. A. Anagnostopoulos, G. Spathoulas, A. Kakarountas, and S. Katzenbeisser. **"Employing Blockchain and Physical Unclonable Functions for Counterfeit IoT Devices Detection"**. In: *Proceedings of the International Conference on Omni-Layer Intelligent Systems*. COINS '19. Association for Computing Machinery, 2019, pp. 172–178. ISBN: 9781450366403. DOI: 10.1145/3312614.3312650. URL: http://doi.acm.org/10.1145/3312614.3312650 (cit. on p. 172).

[233] N. A. Anagnostopoulos, T. Arul, Y. Fan, C. Hatzfeld, J. Lotichius, R. Sharma, F. Fernandes, F. Tehranipoor, and S. Katzenbeisser. **"Securing IoT Devices Using Robust DRAM PUFs"**. In: *2018 Global Information Infrastructure and Networking Symposium (GIIS)*. IEEE, 2018. DOI: 10.1109/GIIS.2018.8635789 (cit. on p. 172).

[234] U. Rührmair, J. Martinez-Hurtado, X. Xu, C. Kraeh, C. Hilgers, D. Kononchuk, J. J. Finley, and W. P. Burleson. **"Virtual Proofs of Reality and their Physical Implementation"**. In: *Proceedings of the 2015 IEEE Symposium on Security and Privacy*. IEEE, 2015, pp. 70–85. DOI: 10.1109/SP.2015.12 (cit. on p. 174).

[235] C. L. Rothwell. **"Exploitation From Malicious PCI Express Peripherals"**. Tech. rep. UCAM-CL-TR-934. University of Cambridge, Computer Laboratory, 2019. URL: https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-934.pdf (cit. on p. 177).

[236] K. Chen. **"Reversing and Exploiting an Apple Firmware Update"**. Black Hat USA 2009, Las Vegas, Nevada, USA. 2009. URL: https://www.blackhat.com/presentations/bh-usa-09/CHEN/BHUSA09-Chen-RevAppleFirm-PAPER.pdf (cit. on p. 177).

[237] A. Seshadri, A. Perrig, L. van Doorn, and P. Khosla. **"SWATT: Software-Based Attestation for Embedded Devices"**. In: *Proceedings of the 2004 IEEE Symposium on Security and Privacy*. IEEE, 2004, pp. 272–282. DOI: 10.1109/SECPRI.2004.1301329 (cit. on p. 177).

[238] A. Seshadri, M. Luk, A. Perrig, L. van Doorn, and P. Khosla. **"SCUBA: Secure Code Update By Attestation in Sensor Networks"**. In: *Proceedings of the 5th ACM Workshop on Wireless Security*. WiSe '06. Association for Computing Machinery, 2006, pp. 85–94. ISBN: 1595935576. DOI: 10.1145/1161289.1161306. URL: http://doi.acm.org/10.1145/1161289.1161306 (cit. on p. 177).

[239] A. Seshadri, M. Luk, and A. Perrig. **"SAKE: Software Attestation for Key Establishment in Sensor Networks"**. In: *Ad Hoc Networks* 9.6 (2011), pp. 1059–1067. ISSN: 1570-8705. DOI: 10.1016/j.adhoc.2010.08.011. URL: http://dx.doi.org/10.1016/j.adhoc.2010.08.011 (cit. on p. 177).

[240] C. Castelluccia, A. Francillon, D. Perito, and C. Soriente. **"On the Difficulty of Software-Based Attestation of Embedded Devices"**. In: *Proceedings of the 16th ACM Conference on Computer and Communications Security*. CCS '09. Association for Computing Machinery, 2009, pp. 400–409. ISBN: 9781605588940. DOI: 10.1145/1653662.1653711. URL: http://doi.acm.org/10.1145/1653662.1653711 (cit. on p. 177).

[241] R. Thibadeau. **"Trusted Computing for Disk Drives and Other Peripherals"**. In: *IEEE Security & Privacy* 4.5 (2006), pp. 26–33. ISSN: 1540-7993. DOI: 10.1109/MSP.2006.136 (cit. on p. 177).

[242] D. Steinmetzer, Y. Yuan, and M. Hollick. **"Beam-Stealing: Intercepting the Sector Sweep to Launch Man-in-the-Middle Attacks on Wireless IEEE 802.11ad Networks"**. In: *Proceedings of the 11th ACM Conference on Security & Privacy in Wireless and Mobile Networks*. WiSec '18. Association for Computing Machinery, 2018, pp. 12–22. ISBN: 9781450357319. DOI: 10.1145/3212480.3212499. URL: https://doi.org/10.1145/3212480.3212499 (cit. on p. 178).

[243] D. Steinmetzer, S. Ahmad, N. Anagnostopoulos, M. Hollick, and S. Katzenbeisser. **"Authenticating the Sector Sweep to Protect Against Beam-Stealing Attacks in IEEE 802.11Ad Networks"**. In: *Proceedings of the 2nd ACM Workshop on Millimeter Wave Networks and Sensing Systems*. mmNets '18. Association for Computing Machinery, 2018, pp. 3–8. ISBN: 9781450359283. DOI: 10.1145/3264492.3264494. URL: http://doi.acm.org/10.1145/3264492.3264494 (cit. on pp. 178, 179).

[244] S. Ahmad. **"Using Physical Unclonable Functions for Data-Link Layer Authenticity Verification to Mitigate Attacks on IEEE 802.11ad Beam Training"**. M.Sc. thesis. Technische Universität Darmstadt, 2018 (cit. on p. 179).

[245] J. Delvaux, D. Gu, D. Schellekens, and I. Verbauwhede. **"Helper Data Algorithms for PUF-Based Key Generation: Overview and Analysis"**. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 34.6 (2015), pp. 889–902. ISSN: 1937-4151. DOI: 10.1109/TCAD.2014.2370531 (cit. on p. 180).

[246] D. Merli, D. Schuster, F. Stumpf, and G. Sigl. **"Side-Channel Analysis of PUFs and Fuzzy Extractors"**. In: *Trust and Trustworthy Computing – Trust 2011*. Ed. by J. M. McCune, B. Balacheff, A. Perrig, A.-R. Sadeghi, A. Sasse, and Y. Beres. Vol. 6740. Lecture Notes in Computer Science (LNCS). Springer, 2011, pp. 33–47. ISBN: 9783642215995. DOI: 10.1007/978-3-642-21599-5_3 (cit. on p. 180).

[247] N. Mexis, N. A. Anagnostopoulos, S. Chen, J. Bambach, T. Arul, and S. Katzenbeisser. **"A Lightweight Architecture for Hardware-Based Security in the Emerging Era of Systems of Systems"**. In: *ACM Journal on Emerging Technologies in Computing Systems* 17.3 (2021). ISSN: 1550-4832. DOI: 10.1145/3458824. URL: https://doi.org/10.1145/3458824 (cit. on p. 196).

[248] N. Mexis, N. A. Anagnostopoulos, S. Chen, J. Bambach, T. Arul, and S. Katzenbeisser. **"A Design for a Secure Network of Networks Using a Hardware and Software Co-Engineering Architecture"**. In: *Proceedings of the SIGCOMM '21 Poster and Demo Sessions*. SIGCOMM '21. Association for Computing Machinery, 2021, pp. 65–67. ISBN: 9781450386296. DOI: 10.1145/3472716.3472849. URL: https://doi.org/10.1145/3472716.3472849 (cit. on p. 196).

List of Abbreviations

5G	Fifth (5th) Generation (of standards for wireless cellular communication)
AR-PUF	Advanced Reconfigurable PUF (Advanced Reconfigurable Physical Unclonable Function / Advanced Reconfigurable Physical Unique Function)
ASIC	Application-Specific Integrated Circuit
B	Byte or Bytes, accordingly. (One byte is equal to eight bits.)
BCH	Bose–Chaudhuri–Hocquenghem (error correction code)
CNT	Carbon Nano-Tube
COTS	Commercial Off-The-Shelf
CPS	Cyber-Physical System
CRC	Collaborative Research Centre
CRP	Challenge-Response Pair
d.h.	das heißt (In German, meaning “that is” / “that means”.)
DFG	Deutsche ForschungsGemeinschaft (In German, meaning “German Research Foundation”.)
DIMM	Dual In-line Memory Module
DLR	Deutsches Zentrum für Luft- und Raumfahrt e.V. (In German, meaning “German Aerospace Center”.)
DNA	DeoxyriboNucleic Acid
DoS	Denial of Service (attack)
Dr.	Doctor
DRAM	Dynamic Random Access Memory
e.g.	exempli gratia (In Latin, meaning “for example’s sake” / “for example”.)
e.V.	eingetragener Verein (In German, meaning “registered association”.)
EC	Electronic Cash (debit card system); originally the acronym stems from the EuroCheque, the unified European cheque system
ECC	Error Correction Code
ECU	Electronic Control Unit
EPF	Expanded Polystyrene Foam
et al.	et alii (In Latin, meaning “and others”.)
etc.	et cetera (In Latin, meaning “and the rest” / “and so forth”.)
FPGA	Field-Programmable Gate Array
GB	GigaByte or GigaBytes, accordingly. (Throughout the text, the term “gigabyte” is used to refer to $1024 \times 1024 \times 1024 = 1073741824$ (2^{30}) bytes, i.e., to what has been proposed to be called a “gibibyte”. Formally, a gigabyte would refer to $1000 \times 1000 \times 1000 = 1000000000$ (10^9) bytes, but *in practice* the term most often is used to refer to $1024 \times 1024 \times 1024$ bytes.)

Gbit	Gigabit or Gigabits, accordingly. (Throughout the text, the term "gigabit" is used to refer to 1024 × 1024 × 1024 = 1073741824 (2^{30}) bits, i.e., to what has been proposed to be called a "gibibit". Formally, a gigabit would refer to 1000 × 1000 × 1000 = 1000000000 (10^9) bits, but *in practice* the term most often is used to refer to 1024 × 1024 × 1024 bits.)
GHz	GigaHertz (One gigahertz is 1000 × 1000 × 1000 = 1000000000 (10^9) hertz.)
i.e.	id est (In Latin, meaning "that is".)
I/O	Input/Output
IBAN	International Bank Account Number
ID	IDentity
IdD	Internet der Dinge (In German, meaning "Internet of Things".)
IEEE	Institute of Electrical and Electronics Engineers (based in the USA)
IIoT	Industrial Internet of Things
IoT	Internet of Things
KB	KiloByte or KiloBytes, accordingly. (Throughout the text, the term "kilobyte" is used to refer to 1024 (2^{10}) bytes, i.e., to what has been proposed to be called a "kibibyte". Formally, a kilobyte would refer to 1000 (10^3) bytes, but *in practice* the term most often is used to refer to 1024 bytes.)
LFSR	Linear Feedback Shift Register
LINAC	LINear ACcelerator
LS	Lightweight Secure
MAC	Media Access Control
MAD	Mutual(ly) Assured Destruction
MB	MegaByte or MegaBytes, accordingly. (Throughout the text, the term "megabyte" is used to refer to 1024 × 1024 = 1048576 (2^{20}) bytes, i.e., to what has been proposed to be called a "mebibyte". Formally, a megabyte would refer to 1000 × 1000 = 1000000 (10^6) bytes, but *in practice* the term most often is used to refer to 1024 × 1024 bytes.)
MCU	MicroContoller Unit
MRAM	Magnetoresistive Random Access Memory
ms	millisecond or milliseconds, accordingly. (One millisecond is $\frac{1}{1000}$ of a second.
NIST	National Institute of Standards and Technology (of the USA)
NVM	Non-Volatile Memory
OS	Operating System
Prof.	Professor
PUF	Physical Unclonable Function ("Physische Unklonbare Funktion", in German.) Or, as suggested in this work, Physical *Unique* Function ("Physische *Einzigartige* Funktion", in German). The term "Physische *Unterscheidbare* Funktion" ("Physical *Distinguishable* Function", in English) could be used in German, so that the acronym remains "PUF" also in the German language.
RAM	Random Access Memory

ReRAM/RRAM	Resistive Random Access Memory
RM	Reconfiguration Module
RNG	Random Number Generator
RO	Ring Oscillator
rPUF	reconfigurable PUF (reconfigurable Physical Unclonable Function / reconfigurable Physical Unique Function)
RSA	Rivest–Shamir–Adleman (public-key cryptography encryption algorithm)
s	second or seconds, accordingly.
SFB	SonderForschungsBereich (In German, meaning “Special Research Area”, indicating a collaborative research program on a wider scientific field.)
SHA	Secure Hash Algorithm
SIM	Subscriber Identity Module or Subscriber Identification Module
SPI	Serial Peripheral Interface
SPP	SchwerPunktProgramme (In German, meaning “Focus Point Program”, indicating a collaborative research program on a particular (prioritised) scientific field.)
SRAM	Static Random Access Memory
TPM	Trusted Platform Module
TRNG	True Random Number Generator
UART	Universal Asynchronous Receiver-Transmitter
US/U.S.	United States (of America)
USA/U.S.A.	United States of America
USB	Universal Serial Bus (Most often, the term refers to the relevant connection protocol, and the relevant port(s) and device(s).)
V	Volt or Volts, accordingly.
V2X	Vehicle-to-Everything (communication)
XOR	eXclusive OR
z.B.	zum Beispiel (In German, meaning “for example”.)
°C	degree Celsius or degrees Celsius, accordingly.

List of Figures

List of Tables

A Curriculum Vitae of the Author

A.1 Personal Information

Name	Nikolaos Athanasios Anagnostopoulos
Place of Birth	Thessaloniki, Greece
Date of Birth	10 September 1986
Email	na@anagnostopoulos.academy

A.2 Education

Years	**Educational Institution**
1992 – 1996	Primary School (91th Primary School of Thessaloniki – 91ο Δημοτικό Σχολείο Θεσσαλονίκης)
1996 – 1998	Primary School (90th Primary School of Thessaloniki – 90ο Δημοτικό Σχολείο Θεσσαλονίκης)
1998 – 2001	Lower Secondary School (1st Model Experimental Lower Secondary School of Thessaloniki – 1ο Πρότυπο Πειραματικό Γυμνάσιο Θεσσαλονίκης)
2001 – 2004	Higher Secondary School (2nd Model Experimental Higher Secondary School of Thessaloniki – 2ο Πρότυπο Πειραματικό Λύκειο Θεσσαλονίκης)
2004 – 2012	University – Bachelor's Degree (Aristotle University of Thessaloniki – Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης)
2012 – 2013	University – Master's Degree (Technical University of Berlin)
2013 – 2014	University – Master's Degree (University of Twente)
2014 – 2022	University – Doctor's Degree (Technical University of Darmstadt)

A.3 Work Experience

Years	**Employed By**	**Position**
2005	Aristotle University of Thessaloniki (Central Library) – Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης (Κεντρική Βιβλιοθήκη)	IT Support Assistant
2014	NXP Semiconductors N.V.	Secure Hardware Analyst (Intern)
2014 – 2019	Technical University of Darmstadt	Research Assistant
2019 to present	University of Passau	Research Assistant

A.4 Professional Memberships

- Institute of Electrical and Electronics Engineers (IEEE) – United States of America
- Association for Computing Machinery (ACM) – United States of America
- Society for Informatics (Gesellschaft für Informatik – GI) – Germany
- International Association for Cryptologic Research (IACR) – United States of America

- American Mathematical Society (AMS) – United States of America
- Royal Institute of Engineers (Koninklijk Instituut Van Ingenieurs – KIVI) – Netherlands
- Association of German Engineers (Verein Deutscher Ingenieure – VDI) – Germany
- Association for Electrical, Electronic and Information Technologies (Verband der Elektrotechnik, Elektronik und Informationstechnik – VDE) – Germany
- Study Association (Studievereniging) Inter-*Actief* of the University of Twente – Netherlands
- EIT Digital Alumni – European Union

B List of Scientific Works Published During Doctoral Studies

B.1 Refereed Publications

B.1.1 Conferences, Workshops & Symposia

- W. Xiong, A. Schaller, N. A. Anagnostopoulos, M. U. Saleem, S. Gabmeyer, S. Katzenbeisser & J. Szefer, "Run-Time Accessible DRAM PUFs in Commodity Devices", Proceedings of the 18th International Conference on Cryptographic Hardware and Embedded Systems (CHES 2016), vol. 9813 of "Lecture Notes in Computer Science (LNCS)", pp. 432-453, Springer, 2016. DOI: 10.1007/978-3-662-53140-2_21

- A. Schaller, W. Xiong, N. A. Anagnostopoulos, M. U. Saleem, S. Gabmeyer, S. Katzenbeisser & J. Szefer, "Intrinsic Rowhammer PUFs: Leveraging the Rowhammer Effect for Improved Security", Proceedings of the 2017 IEEE International Symposium on Hardware Oriented Security and Trust (HOST 2017), IEEE, 2017. DOI: 10.1109/HST.2017.7951729

- N. Matyunin, N. A. Anagnostopoulos, S. Boukoros, M. Heinrich, A. Schaller, M. Kolinichenko & S. Katzenbeisser, "Tracking Private Browsing Sessions Using CPU-Based Covert Channels", Proceedings of the 11th ACM Conference on Security & Privacy in Wireless and Mobile Networks (WiSec 2018), pp. 63-74, ACM, 2018. DOI: 10.1145/3212480.3212489

- N. A. Anagnostopoulos, T. Arul, M. Rosenstihl, A. Schaller, S. Gabmeyer & S. Katzenbeisser, "Low-Temperature Data Remanence Attacks Against Intrinsic SRAM PUFs", Proceedings of the 21st Euromicro Conference on Digital System Design 2018 (DSD 2018), pp. 581-585, IEEE, 2018. DOI: 10.1109/DSD.2018.00102

- T. Arul, N. A. Anagnostopoulos & S. Katzenbeisser, "Behavioral Workload Generation for IPTV", Proceedings of the 2018 IEEE 8th International Conference on Consumer Electronics - Berlin (ICCE-Berlin 2018), IEEE, 2018. DOI: 10.1109/ICCE-Berlin.2018.8576230

- N. A. Anagnostopoulos, T. Arul, Y. Fan, J. Lotichius, C. Hatzfeld, F. Fernandes, R. Sharma, F. Tehranipoor & S. Katzenbeisser, "Securing IoT Devices Using Robust DRAM PUFs", Proceedings of the 2018 Global Information Infrastructure and Networking Symposium (GIIS 2018), IEEE, 2018. DOI: 10.1109/GIIS.2018.8635789

- D. Steinmetzer, S. Ahmad, N. A. Anagnostopoulos, M. Hollick & S. Katzenbeisser, "Authenticating the Sector Sweep to Protect Against Beam-Stealing Attacks in IEEE 802.11ad Networks", Proceedings of the 2nd ACM Workshop on Millimeter Wave Networks and Sensing Systems (mmNets 2018), pp. 3-8, ACM, 2018. DOI: 10.1145/3264492.3264494

- D. Püllen, N. A. Anagnostopoulos, T. Arul & S. Katzenbeisser, "Poster: Hierarchical Integrity Checking in Heterogeneous Vehicular Networks", Proceedings of the 2018 IEEE Vehicular Networking Conference (VNC 2018), IEEE, 2018. DOI: 10.1109/VNC.2018.8628375

- T. Arul, N. A. Anagnostopoulos & S. Katzenbeisser, “Privacy and Usability of IPTV Recommender Systems”, Proceedings of the 2019 IEEE International Conference on Consumer Electronics (ICCE 2019), IEEE, 2019. DOI: 10.1109/ICCE.2019.8662046

- N. A. Anagnostopoulos, T. Arul, Y. Fan, M. Kumar & S. Katzenbeisser, “AR-PUFs: Advanced Security Primitives for the Internet of Things and Cyber-Physical Systems”, Proceedings of the 2019 IEEE International Conference on Consumer Electronics (ICCE 2019), IEEE, 2019. DOI: 10.1109/ICCE.2019.8661840

- N. A. Anagnostopoulos, Y. Fan, T. Arul, R. Sarangdhar & S. Katzenbeisser, “Lightweight Security Solutions for IoT Implementations in Space”, Proceedings of the 2019 IEEE Topical Workshop on Internet of Space (TWIOS 2019), IEEE, 2019. DOI: 10.1109/TWIOS.2019.8771257

- W. Xiong, N. A. Anagnostopoulos, A. Schaller, S. Katzenbeisser & J. Szefer, “Spying on Temperature Using DRAM”, Proceedings of the 22nd Design, Automation & Test in Europe Conference & Exhibition (DATE 2019), pp. 13-18, IEEE, 2019. DOI: 10.23919/DATE.2019.8714882

- D. Püllen, N. A. Anagnostopoulos, T. Arul & S. Katzenbeisser, “Safety and Security Co-Engineering of the FlexRay Bus in Vehicular Networks”, Proceedings of the 1st International Conference on Omni-Layer Intelligent Systems (COINS 2019), pp. 31-37, ACM, 2019. DOI: 10.1145/3312614.3312626

- L. Negka, G. Gketsios, N. A. Anagnostopoulos, G. Spathoulas, A. Kakarountas & S. Katzenbeisser, “Employing Blockchain and Physical Unclonable Functions for Counterfeit IoT Devices Detection”, Proceedings of the 1st International Conference on Omni-Layer Intelligent Systems (COINS 2019), pp. 172-178, ACM, 2019. DOI: 10.1145/3312614.3312650

- D. Püllen, N. A. Anagnostopoulos, T. Arul & S. Katzenbeisser, “Using Implicit Certification to Efficiently Establish Authenticated Group Keys for In-Vehicle Networks”, Proceedings of the 2019 IEEE Vehicular Networking Conference (VNC 2019), IEEE, 2019. DOI: 10.1109/VNC48660.2019.9062785

- D. Püllen, N. Anagnostopoulos, T. Arul & S. Katzenbeisser, “Safety Meets Security: Using IEC 62443 for a Highly Automated Road Vehicle”, Proceedings of the 39th International Conference on Computer Safety, Reliability, and Security (SAFECOMP 2020), vol. 12234 of “Lecture Notes in Computer Science (LNCS)”, pp. 325-340, Springer, 2020. DOI: 10.1007/978-3-030-54549-9_22

- T. Arul, N. A. Anagnostopoulos, S. Reißig & S. Katzenbeisser, “A Study of the Spatial Auto-Correlation of Memory-Based Physical Unclonable Functions”, Proceedings of the 2020 European Conference on Circuit Theory and Design (ECCTD 2020), IEEE, 2020. DOI: 10.1109/ECCTD49232.2020.9218302

- N. Mexis, N. A. Anagnostopoulos, S. Chen, J. Bambach, T. Arul & S. Katzenbeisser, “A Design for a Secure Network of Networks Using a Hardware and Software Co-Engineering Architecture”, Proceedings of the SIGCOMM 2021 Poster and Demo Sessions (SIGCOMM 2021), pp. 65-67, ACM, 2021. DOI: 10.1145/3472716.3472849

- N. A. Anagnostopoulos, Y. Fan, M. Heinrich, N. Matyunin, D. Püllen, P. Muth, C. Hatzfeld, M. Rosenstihl, T. Arul & S. Katzenbeisser, "Low-Temperature Attacks Against Digital Electronics: A Challenge for the Security of Superconducting Modules in High-Speed Magnetic Levitation (MagLev) Trains", Proceedings of the 14th Workshop on Low-Temperature Electronics (WOLTE 2021), IEEE, 2021. DOI (MANUSCRIPT): 10.1109/WOLTE49037.2021.9555437 DOI (POSTER): 10.13140/RG.2.2.17643.67363

- F. Frank, N. A. Anagnostopoulos, S. Böttger, S. Hermann, T. Arul, S. G. Stavrinides & S. Katzenbeisser, "A Dedicated Mixed-Signal Characterisation and Testing Framework for Novel Digital Security Circuits That Use Carbon-Nanotube-Based Physical Unclonable Functions", Proceedings of the 11th International Conference on Modern Circuits and Systems Technologies (MOCAST 2022), IEEE, 2022. DOI: 10.1109/MOCAST54814.2022.9837567

B.1.2 Journals & Magazines

- N. A. Anagnostopoulos, S. Katzenbeisser, J. Chandy & F. Tehranipoor, "An Overview of DRAM-Based Security Primitives", Cryptography, vol. 2, iss. 2, MDPI, 2018. DOI: 10.3390/cryptography2020007

- A. Schaller, W. Xiong, N. A. Anagnostopoulos, M. U. Saleem, S. Gabmeyer, B. Škorić, S. Katzenbeisser & J. Szefer, "Decay-Based DRAM PUFs in Commodity Devices", IEEE Transactions on Dependable and Secure Computing (TDSC), vol. 16, iss. 3, pp. 462-475, IEEE, 2018. DOI: 10.1109/TDSC.2018.2822298

- N. A. Anagnostopoulos, T. Arul, Y. Fan, C. Hatzfeld, A. Schaller, W. Xiong, M. Jain, M. U. Saleem, J. Lotichius, S. Gabmeyer, J. Szefer & S. Katzenbeisser, "Intrinsic Run-Time Row Hammer PUFs: Leveraging the Row Hammer Effect for Run-Time Cryptography and Improved Security", Cryptography, vol.2, iss. 3, MDPI, 2018. DOI: 10.3390/cryptography2030013

- N. A. Anagnostopoulos, T. Arul, M. Rosenstihl, A. Schaller, S. Gabmeyer & S. Katzenbeisser, "Attacking SRAM PUFs Using Very-Low-Temperature Data Remanence", Microprocessors and Microsystems, vol. 71, art. 102864, Elsevier, 2019. DOI: 10.1016/j.micpro.2019.102864

- D. Püllen, N. A. Anagnostopoulos, T. Arul & S. Katzenbeisser, "Securing FlexRay-Based In-Vehicle Networks", Microprocessors and Microsystems, vol. 77, art. 103144, Elsevier, 2020. DOI: 10.1016/j.micpro.2020.103144

- M. Yue, N. Karimian, W. Yan, N. A. Anagnostopoulos & F. Tehranipoor, "DRAM-based Authentication using Deep Convolutional Neural Networks", IEEE Consumer Electronics Magazine, vol. 10, iss.4, pp.8-17, IEEE, 2020. DOI: 10.1109/MCE.2020.3002528

- N. A. Anagnostopoulos, S. Ahmad, T. Arul, D. Steinmetzer, M. Hollick & S. Katzenbeisser, "Low-Cost Security for Next-Generation IoT Networks", ACM Transactions on Internet Technology, vol. 20, iss. 3, art. 30, ACM, 2020. DOI: 10.1145/3406280

- W. Xiong, A. Schaller, N. A. Anagnostopoulos, M. U. Saleem, S. Gabmeyer, S. Katzenbeisser & J. Szefer, "DRAM PUFs in Commodity Devices", IEEE Design & Test, vol. 38, iss. 3, pp. 76-83, IEEE, 2021. DOI: 10.1109/MDAT.2021.3063370

- N. Mexis, N. A. Anagnostopoulos, S. Chen, J. Bambach, T. Arul & S. Katzenbeisser, “A Lightweight Architecture for Hardware-Based Security in the Emerging Era of Systems of Systems”, ACM Journal on Emerging Technologies in Computing Systems, vol. 17, iss. 3, art. 43, ACM, 2021. DOI: 10.1145/3458824

B.2 Non-Refereed Publications

B.2.1 Pre-prints

- W. Xiong, A. Schaller, N. A. Anagnostopoulos, M. U. Saleem, S. Gabmeyer, S. Katzenbeisser & J. Szefer, “Run-Time Accessible DRAM PUFs in Commodity Devices”, Cryptology ePrint Archive: Report 2016/253, 2016. URL: https://eprint.iacr.org/2016/253
- N. A. Anagnostopoulos, S. Katzenbeisser, M. Rosenstihl, A. Schaller, S. Gabmeyer & T. Arul, “Low-Temperature Data Remanence Attacks Against Intrinsic SRAM PUFs”, Cryptology ePrint Archive: Report 2016/769, 2016. URL: https://eprint.iacr.org/2016/769
- N. A. Anagnostopoulos, T. Arul, Y. Fan, C. Hatzfeld, A. Schaller, W. Xiong, M. Jain, M. U. Saleem, J. Lotichius, S. Gabmeyer, J. Szefer & S. Katzenbeisser, “Intrinsic Run-Time Row Hammer PUFs: Leveraging the Row Hammer Effect for Run-Time Cryptography and Improved Security”, MDPI Preprints, 2018. DOI: 10.20944/preprints201804.0369.v1
- A. Schaller, W. Xiong, N. A. Anagnostopoulos, M. U. Saleem, S. Gabmeyer, S. Katzenbeisser & J. Szefer, “Intrinsic Rowhammer PUFs: Leveraging the Rowhammer Effect for Improved Security”, arXiv e-prints: 1902.04444 [cs.CR], 2019. URL: https://arxiv.org/abs/1902.04444
- N. Karimian, F. Tehranipoor, N. Anagnostopoulos & W. Yan, “DRAMNet: Authentication based on Physical Unique Features of DRAM Using Deep Convolutional Neural Networks”, arXiv e-prints: 1902.09094 [cs.CR], 2019. URL: https://arxiv.org/abs/1902.09094
- N. A. Anagnostopoulos, “The Role of Cost in the Integration of Security Features in Integrated Circuits for Smart Cards”, arXiv e-prints: 2101.10293 [q-fin.RM], 2021. URL: https://arxiv.org/abs/2101.10293
- N. A. Anagnostopoulos, “Exploring the Complicated Relationship Between Patents and Standards, With a Particular Focus on the Telecommunications Sector”, arXiv e-prints: 2101.10548 [econ.GN], 2021. URL: https://arxiv.org/abs/2101.10548
- N. A. Anagnostopoulos, “Ear Recognition”, arXiv e-prints: 2101.10540 [cs.CV], 2021. URL: https://arxiv.org/abs/2101.10540
- N. A. Anagnostopoulos, Y. Fan, M. U. Saleem, N. Mexis, F. Frank, T. Arul & S. Katzenbeisser, “On the Sustainability of Lightweight Cryptography Based on PUFs Implemented on NAND Flash Memories Using Programming Disturbances”, arXiv e-prints: 2204.02498 [cs.CR], 2022. URL: https://arxiv.org/abs/2204.02498

N. A. Anagnostopoulos, Y. Fan, M. U. Saleem, N. Mexis, F. Frank, T. Arul & S. Katzenbeisser, "On the Sustainability of Lightweight Cryptography Based on PUFs Implemented on NAND Flash Memories Using Programming Disturbances", TechRxiv preprint server, 2022. URL: https://www.techrxiv.org/articles/preprint/On_the_Sustainability_of_Lightweight_Cryptography_Based_on_PUFs_Implemented_on_NAND_Flash_Memories_Using_Programming_Disturbances/19529263 DOI: 10.36227/techrxiv.19529263

- F. Frank, W. Xiong, N. A. Anagnostopoulos, A. Schaller, T. Arul, F. Koushanfar, S. Katzenbeisser, U. Rührmair & S. Szefer, "Abusing Commodity DRAMs in IoT Devices to Remotely Spy on Temperature", arXiv e-prints: 2208.02125 [cs.CR], 2022. URL: https://arxiv.org/abs/2208.02125

B.2.2 Conferences, Workshops, Conventions & Symposia

- N. A. Anagnostopoulos, A. Schaller, Y. Fan, W. Xiong, F. Tehranipoor, T. Arul, S. Gabmeyer, J. Szefer, J. A. Chandy & S. Katzenbeisser, "Insights into the Potential Usage of the Initial Values of DRAM Arrays of Commercial Off-the-Shelf Devices for Security Applications", 26th Crypto-Day, 1-2 June 2017, Nürnberg, Germany. URL: https://fg-krypto.gi.de/fileadmin/FG/KRYPTO/Proceedings/LN_CryptoDay26_SUSE.pdf. DOI (PRESENTATION SLIDES): 10.13140/RG.2.2.21396.50569. DOI (MANUSCRIPT): 10.13140/RG.2.2.27089.63849

- R. Sarangdhar, Y. Fan, N. A. Anagnostopoulos, U. Gayer, F. Flederer, T. Mikschl, T. Arul, P. R. John, K. Hierholz, S. Montenegro & S. Katzenbeisser, "An Investigation of the Effects of Radiation on Current Key Storage Solutions and on Physical Unclonable Functions (PUFs) Being Used as Key Storage", 27th Crypto-Day, 7-8 December 2017, Frankfurt (Oder), Germany. DOI: 10.13140/RG.2.2.33800.52483

- N. A. Anagnostopoulos, S. Gabmeyer, T. Arul & S. Katzenbeisser, "An Extensive Classification and Analysis of Attacks Against Physical Unclonable Functions (PUFs)", 27th Crypto-Day, 7-8 December 2017, Frankfurt (Oder), Germany. DOI: 10.13140/RG.2.2.25411.91689

- M. Kumar, N. A. Anagnostopoulos, Y. Fan & S. Katzenbeisser, "Advanced Reconfigurable Physical Unclonable Functions (AR-PUFs) and Their Security Applications", 28th Crypto-Day, 7-8 June 2018, Kirchheim bei München, Germany. [In: Loebenberger, D. & Nüsken, M. (eds.), crypto day matters 28. Bonn: Gesellschaft für Informatik e.V. / FG KRYPTO.] DOI: 10.18420/cdm-2018-28-22

- N. A. Anagnostopoulos, T. Arul, Y. Fan, C. Hatzfeld, F. Tehranipoor & S. Katzenbeisser, "Addressing the Effects of Temperature Variations on Intrinsic Memory-Based Physical Unclonable Functions", 28th Crypto-Day, 7-8 June 2018, Kirchheim bei München, Germany. [In: Loebenberger, D. & Nüsken, M. (eds.), crypto day matters 28. Bonn: Gesellschaft für Informatik e.V. / FG KRYPTO.] DOI: 10.18420/cdm-2018-28-23

- N. A. Anagnostopoulos, T. Arul, Y. Fan, R. Sarangdhar, R. Sharma, M. Rosenstihl, C. Hatzfeld, F. Tehranipoor & S. Katzenbeisser, "On the Effects of Environmental Factors on the Functionality of Modern Dynamic Random Access Memory Modules", 7th International Conference "Micro&Nano" 2018, 5-7 November 2018, Thessaloniki, Greece. DOI: 10.13140/RG.2.2.16373.83681

- W. Xiong, A. Schaller, N. A. Anagnostopoulos, M. U. Saleem, S. Gabmeyer, S. Katzenbeisser & J. Szefer, Top Pick Candidate Paper: “Run-Time Accessible DRAM PUFs in Commodity Devices” [presented in the 18th International Conference on Cryptographic Hardware and Embedded Systems (CHES 2016)], 2019 Workshop on Top Picks in Hardware and Embedded Security, 7 November 2019, Westminster, Colorado, United States of America.

- D. Püllen, N. A. Anagnostopoulos, T. Arul & S. Katzenbeisser, “ISA-62443 in the Automotive Context: Threat Identification and Mitigation for a Novel Vehicular Architecture”, 9. Tagung Automatisiertes Fahren, 21-22 November 2019, München, Germany.

- N. Mexis, N. A. Anagnostopoulos, T. Arul, F. Frank & S. Katzenbeisser, “Fuzzy Extractors using Low-Density Parity-Check Codes”, 33rd Crypto-Day, 17 September 2021, online. [In: Gazdag, S.-L., Tiepelt, M., Loebenberger, D. & Nüsken, M. (Editors), crypto day matters 33. Bonn: Gesellschaft für Informatik e.V. / FG KRYPTO.] DOI: 10.18420/cdm-2021-33-41

- S. Böttger, F. Frank, N. A. Anagnostopoulos, T. Arul, S. Katzenbeisser & S. Hermann, “Development of Nanomaterial-Based Physically Unclonable Functions and Dedicated Measurement and Testing Devices”, Fall School on Nano-Electronics for Secure Systems (NESSY), 10-11 November 2021, Lübeck, Germany. URL: https://www.anagnostopoulos.academy/assets/529c351492/211104_NessyPoster_SB-1.pdf

- N. A. Anagnostopoulos, Y. Fan, M. U. Saleem, N. Mexis, F. Frank, T. Arul & S. Katzenbeisser, “On the Sustainability of Lightweight Cryptography Based on Flash PUFs”, Workshop on “Sustainability in Security & Security for Sustainability” (co-located with the 25th Design, Automation and Test in Europe Conference & Exhibition (DATE 2022), as Workshop 09), 18 March 2022, online.
  - Proceedings not published. Published preprint versions of this work:
    URL (TechRxiv): https://www.techrxiv.org/articles/preprint/On_the_Sustainability_of_Lightweight_Cryptography_Based_on_PUFs_Implemented_on_NAND_Flash_Memories_Using_Programming_Disturbances/19529263 DOI (TechRxiv): 10.36227/techrxiv.19529263 arXiv: https://arxiv.org/abs/2204.02498

- E. Gelóczi, N. Mexis, N. A. Anagnostopoulos, F. Frank, T. Arul, S. G. Stavrinides & S. Katzenbeisser, “Secure Communication via Chaotic Cryptography”, 34th Crypto-Day, 9-10 June 2022, Weiden in der Oberpfalz, Germany [In: Loebenberger, D. & Nüsken, M. (Editors), crypto day matters 34. Bonn: Gesellschaft für Informatik e.V. / FG KRYPTO.] DOI: 10.18420/cdm-2022-34-01

B.2.3 Data Sets

- W. Xiong, A. Schaller, N. A. Anagnostopoulos, M. U. Saleem, S. Gabmeyer, S. Katzenbeisser & J. Szefer, “DRAM DECAY PUF & DRAM ROWHAMMER PUF”, Trust-Hub. URL: https://www.trust-hub.org/data

B.2.4 Encyclopedia Entries

- N. A. Anagnostopoulos, “Physical Unclonable Function”, MDPI Encyclopedia. DOI: 10.32545/encyclopedia201806.0001.v4

C List of Theses Supervised During Doctoral Studies

C.1 Supervised for the Technical University of Darmstadt

- Kashif Jawed: "An Investigation and Implementation for the Effect of Row Hammering in Static Random Access Memory (SRAM) and Dynamic Random Access Memory (DRAM)", 2016. (M. Sc. Thesis)
- Sebastian Schurig: "Development of a User Interface and Implementation of Specific Software Tools for the Evaluation and Realization of PUFs With Respect to Security Applications", 2017. (M. Sc. Thesis)
- Prankur Chauhan: "Improvement and Integration of Software Tools for the Evaluation and Realization of Physical Unclonable Functions (PUFs) into an Open-Source Library of Cryptographic Components (CogniCrypt)", 2017. (M. Sc. Thesis)
- Ravi Sarangdhar: "An Investigation of the Effects of Radiation on Current Key Storage Solutions and on Physical Unclonable Functions (PUFs) Being Used as Key Storage", 2017. (M. Sc. Thesis)
- Saad Ahmad: "Using Physical Unclonable Functions for Data-Link Layer Authenticity Verification to Mitigate Attacks on IEEE 802.11ad Beam Training", 2018. (M. Sc. Thesis)
- Umair Muhammad Saleem: "Flash-Based Physical Unclonable Functions (PUFs) Using Commercial Off-the-Shelf (COTS) NAND Flash Memory", 2018. (M. Sc. Thesis)
- Ratika Sharma: "Effects of Voltage Variation on Memory-Based Physical Unclonable Functions", 2018. (M. Sc. Thesis)
- Stephanie Senjuty Bartsch: "An Investigation of the Effects of Temperature-Based Aging on DRAM Retention-Based PUFs", Technical University of Darmstadt, 2019. (M. Sc. Thesis)
- Bassel Hanna: "PUF-Based Authentication Web-Server", Technical University of Darmstadt, 2019. (M. Sc. Thesis)
- Iratxe González Garrido: "Improving the Security of IOTA Car Wallets", Technical University of Darmstadt, 2020. (M. Sc. Thesis)
- Sergej Reißig: "Testing the Spatial Correlation of Weak Memory-Based PUFs", Technical University of Darmstadt, 2020. (M. Sc. Thesis)

C.2 Supervised for the University of Passau

- Michal Slezak: "Implementation and Security Evaluation of Advanced Reconfigurable PUFs", University of Passau, 2020. (M. Sc. Thesis)
- Daniel Schaubschläger: "Lightweight Power-Supply-Noise-Based True Random Number Generator", University of Passau, 2020. (B.Sc. Thesis)

- Sampada Diwanji: “A Single Sign-On System Based on Physical Unclonable Functions”, University of Passau, 2020. (M.Sc. Thesis)
- Rine Rajendran: “Testing the Capabilities of Machine Learning for Classification in Physical Unclonable Functions (PUFs)”, University of Passau, 2021. (M.Sc. Thesis)
- Vinoth Kumar Adusumilli Govindarajulu: “Low-Temperature Data Remanence Attacks Against External SRAM PUFs”, University of Passau, 2021. (M.Sc. Thesis)
- Nishanth Kandaswamy Subramanian: “Development of Accurate 2D & 3D Curve-Fitting Models to Study the Characteristics of Power Magnetics and Capacitors”, University of Passau, 2021. (M.Sc. Thesis)
- Nico Mexis: “Fuzzy Extractors unter Einsatz von Low-Density Parity-Check Codes”, University of Passau, 2021. (B.Sc. Thesis)
- Emiliia Nazarenko: “Implementation and Investigation of a Real-World Stream Encryption Mechanism Based on Synchronised Chua Chaotic Circuits”, University of Passau, 2021. (M.Sc. Thesis)

D List of Courses Taught During Doctoral Studies

D.1 Taught at the Technical University of Darmstadt

- "Privacy by Design" seminar (Winter Semester 2015–2016)
- "Privacy-Enhancing Technologies" seminar (Winter Semester 2016–2017)
- "Project Lab Security Engineering" practical / internship course (Winter Semester 2016–2017)
- "Privacy by Design" seminar (Summer Semester 2017)
- "Project Lab Security Engineering" practical / internship course (Summer Semester 2017)
- "Privacy-Enhancing Technologies" seminar (Winter Semester 2017–2018)
- "Project Lab Security Engineering" practical / internship course (Winter Semester 2017–2018)
- "Privacy by Design" seminar (Summer Semester 2018)
- "Project Lab Security Engineering" practical / internship course (Summer Semester 2018)
- "Privacy-Enhancing Technologies" seminar (Winter Semester 2018–2019)
- "Project Lab Security Engineering" practical / internship course (Winter Semester 2018–2019)
- "Security Engineering Lab" practical / internship course (Winter Semester 2018–2019)
- "Bachelor's Internship" practical / internship course (Winter Semester 2018/19)

D.2 Taught at the University of Passau

- "Computer Architecture" exercises (Summer Semester 2019)
- "Technical Computer Science" exercises (Winter Semester 2019–2020)
- "Hardware-Based Security" exercises (Winter Semester 2019–2020)
- "Advanced Security Engineering Lab" practical / internship exercise course (Winter Semester 2019–2020)
- "Master Seminar Security Engineering" seminar (Summer Semester 2020)
- "Computer Architecture" exercises (Summer Semester 2020)
- "Master Seminar Security Engineering" seminar (Winter Semester 2020–2021)
- "Advanced Security Engineering Lab" practical / internship exercise course (Winter Semester 2020–2021)
- "Master Seminar Security Engineering" seminar (Summer Semester 2021)

- “Advanced Security Engineering Lab” practical / internship exercise course (Summer Semester 2021)
- “Master Seminar Security Engineering” seminar (Winter Semester 2021–2022)
- “Advanced Security Engineering Lab” practical / internship exercise course (Winter Semester 2021–2022)
- “Master Seminar Security Engineering” seminar (Summer Semester 2022)
- “Advanced Security Engineering Lab” practical / internship exercise course (Summer Semester 2022)

E Other Professional Activities Performed During Doctoral Studies

E.1 Reviews For Scientific Venues and Publications

E.1.1 Journals

- Microelectronics Journal, Elsevier (Outstanding Reviewer)
- Transactions on Very Large Scale Integration (TVLSI) Systems, IEEE
- Transactions on Dependable and Secure Computing (TDSC), IEEE
- Consumer Electronics Magazine, IEEE
- Security and Communication Networks, Hindawi - Wiley
- Microelectronic Engineering, Elsevier
- Cryptography, MDPI
- Journal of Industrial Information Integration, Elsevier
- Transactions on Circuits and Systems II: Express Briefs, IEEE
- Information, MDPI
- Transactions on Circuits and Systems I: Regular Papers, IEEE
- Discover Internet of Things, Springer
- Computation, MDPI
- Microprocessors and Microsystems, Elsevier
- Transactions on Information Forensics and Security, IEEE
- Computer Standards & Interfaces, Elsevier
- Electronics, MDPI
- Artificial Intelligence Review, Elsevier
- Energy Reports, Elsevier
- Micromachines, MDPI
- Entropy, MDPI
- Journal of Low-Power Electronics and Applications, MDPI
- Sensors, MDPI
- Frontiers in Signal Processing, Frontiers Media

- Symmetry, MDPI
- Transactions on Cognitive Communications and Networking, IEEE
- Applied Sciences, MDPI

E.1.2 Conferences, Workshops & Symposia

- International Symposium on Circuits and Systems (ISCAS) 2019, IEEE
- 2nd International Conference on Computer Applications & Information Security (ICCAIS 2019), IEEE Region 8
- 10th International Workshop on Constructive Side-Channel Analysis and Secure Design (COSADE 2019), IACR
- International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS) 2019, ACM & IEEE
- 4th IEEE International Circuit and System Symposium (ICSyS 2019), IEEE
- 15th IEEE Asia Pacific Conference on Circuits and Systems (APCCAS 2019), IEEE
- 7th International Symposium on Security in Computing and Communications (SSCC 2019), Indian Institute of Information Technology and Management – Kerala (IIITM–K)
- International Symposium on Circuits and Systems (ISCAS) 2020, IEEE
- 3rd International Conference on Computer Applications & Information Security (ICCAIS 2020), IEEE Region 8
- IEEE Cloud Summit 2020, IEEE
- International Symposium on Circuits and Systems (ISCAS) 2021, IEEE
- 17th IEEE Asia Pacific Conference on Circuits and Systems (APCCAS 2021), IEEE
- 11th IEEE International Conference on Consumer [Electronics] Technology in Berlin (ICCE-Berlin 2021), IEEE
- IEEE Cloud Summit 2021, IEEE
- Design, Automation and Test in Europe (DATE) Conference 2022, European Design and Automation Association (EDAA) & IEEE
- International Symposium on Circuits and Systems (ISCAS) 2022, IEEE
- 12th IEEE International Conference on Consumer [Electronics] Technology in Berlin (ICCE-Berlin 2022), IEEE

E.2 Presentations Given at Non-Formal Scientific Venues

- CROSSING Research Seminar: P3: “Hardware-Entangled Cryptography”, 13 August 2015, Darmstadt, Germany.
- Azure Meetup Berlin: Advanced Analytics for Highly Secured Environments: “Hardware-Entangled Security: Extracting a Secure Key From IoT Hardware”, 9 November 2016, Berlin, Germany.

E.3 Positions Served in Scientific Activities and Venues

- Valet de Cyber: 23rd ACM Conference on Computer and Communications Security (CCS 2016), 24-28 October 2016, Vienna, Austria.
- Session Chair: Automotive 1 session, 2018 IEEE 8th International Conference on Consumer Electronics - Berlin (ICCE-Berlin), 2-5 September 2018, Berlin, Germany.
- Moderator: “Security of Cyber-Physical Systems” session, IEEE Computer Society DVP-SYP (Distinguished Visitors Program - Student & Young Professional) Virtual Conference on Hot Topics in Cybersecurity, 16-17 October 2020, online.
- Research Assistant: Phase I (2014-2018) of the Project “P3: Hardware-Entangled Security” (project number 236615297) of the Collaborative Research Centre (CRC): “CROSSING – Cryptography-Based Security Solutions: Enabling Trust in New and Next Generation Computing Environments” (SonderForschungsBereich – SonderForschungsBereich (SFB) 1119) of the German Research Foundation (Deutsche ForschungsGemeinschaft – DFG).
- Research Assistant: Phase I (2020-2023) of the Project “NANOSEC: Tamper-Evident PUFs based on Nanostructures for Secure and Robust Hardware Security Primitives” (project number 439892735) of the Priority Program (SchwerPunktProgramme – SPP) 2253: “Nano Security: From Nano-Electronics to Secure Systems” of the German Research Foundation (Deutsche ForschungsGemeinschaft – DFG).