cryptography

Article

An Overview of DRAM-Based Security Primitives

Nikolaos Athanasios Anagnostopoulos [1], Stefan Katzenbeisser [1], John Chandy [2] and Fatemeh Tehranipoor [3,*]

[1] Computer Science Department, Technical University of Darmstadt, Mornewegstraße 32, S4|14, 64293 Darmstadt, Germany; anagnostopoulos@seceng.informatik.tu-darmstadt.de (N.A.A.); katzenbeisser@seceng.informatik.tu-darmstadt.de (S.K.)

[2] Department of Electrical and Computer Engineering, University of Connecticut, 371 Fairfield Way, U-4157, Storrs, CT 06269-4157, USA; chandy@engr.uconn.edu

[3] School of Engineering, San Francisco State University, 1600 Holloway Avenue, San Francisco, CA 94132, USA

* Correspondence: tehranipoor@sfsu.edu; Tel.: +1-415-338-7821

Received: 25 February 2018; Accepted: 26 March 2018; Published: 28 March 2018

Abstract: Recent developments have increased the demand for adequate security solutions, based on primitives that cannot be easily manipulated or altered, such as hardware-based primitives. Security primitives based on Dynamic Random Access Memory (DRAM) can provide cost-efficient and practical security solutions, especially for resource-constrained devices, such as hardware used in the Internet of Things (IoT), as DRAMs are an intrinsic part of most contemporary computer systems. In this work, we present a comprehensive overview of the literature regarding DRAM-based security primitives and an extended classification of it, based on a number of different criteria. In particular, first, we demonstrate the way in which DRAMs work and present the characteristics being exploited for the implementation of security primitives. Then, we introduce the primitives that can be implemented using DRAM, namely Physical Unclonable Functions (PUFs) and True Random Number Generators (TRNGs), and present the applications of each of the two types of DRAM-based security primitives. We additionally proceed to assess the security such primitives can provide, by discussing potential attacks and defences, as well as the proposed security metrics. Subsequently, we also compare these primitives to other hardware-based security primitives, noting their advantages and shortcomings, and proceed to demonstrate their potential for commercial adoption. Finally, we analyse our classification methodology, by reviewing the criteria employed in our classification and examining their significance.

Keywords: dynamic random access memory (DRAM); physical unclonable function (PUF); true random number generator (TRNG); security primitive; overview

1. Introduction

Recent events have served to amplify the need for more adequate security and privacy solutions for modern computer systems. Such events include the disclosure of a state-organised system of online surveillance covering the whole world [1], as well as newly reported software and hardware vulnerabilities that affect systems used every day by normal users. As these events affect the vast majority of the public, there has been significant pressure to address them in a thorough and transparent manner.

It is for this reason that research regarding IT security has been growing rapidly the last few years. The quick development of this research field has, in turn, brought forward a rise in relevant publications, regarding both software and hardware security. As an increasing number of vulnerabilities is being found in current security mechanisms and implementations, there is a growing demand for better security primitives that will prove more resistant to existing attacks.

This has led to an increased interest for hardware-based security primitives, such as Physical Unclonable Functions (PUFs) and True Random Number Generators (TRNGs), because hardware implementations are less exposed to attackers than software ones. Intrinsic implementations, which do not require the addition of other hardware components, have been proposed as a lightweight and cost-efficient basis for security solutions. Memory components, such as Static Random Access Memories (SRAMs), Dynamic Random Access Memories (DRAMs) and Flash memories [2–6], have proven to be particularly well-suited for the implementation of intrinsic hardware-based security primitives.

DRAMs seem to have a number of additional advantages regarding their usage for the implementation of security primitives, such as their presence in most contemporary computer systems, their large storage size, and the easy way in which they can be accessed, even at run-time. In particular, their large storage size can guarantee both the existence of an adequate amount of entropy and that a part of them can be used, for a limited amount of time, exclusively as a dedicated security primitive, while the system is running. These advantages have led to an increased focus being placed on novel DRAM-based security primitives in recent publications regarding hardware-based security primitives.

However, the fast pace of research and the large number of recent publications regarding DRAM-based security primitives make it difficult to keep track of developments in this field and thus retain a degree of obscurity over it. Additionally, the public tends to trust hardware much more than software, due to the volatile nature of software and the physical nature of hardware. It is, therefore, crucial to perform an overview study of the literature regarding DRAM-based security primitives, not only in order to expose the current state of the art in this field, but also to provide helpful insights into its future development.

We aim to address this urgent need with the current paper, by making the following contributions:

1. We provide a comprehensive overview of literature relevant to DRAM-based security primitives.
2. We then classify this literature using a number of criteria, in order to allow for a clear and thorough view into the current state of the art regarding DRAM-based security primitives.
3. We also consider, in our taxonomy, their potential applications of such primitives, as well as their security, in order to provide a brief evaluation of them.
4. We additionally compare them to other hardware-based security primitives, noting their advantages and disadvantages, and we also examine their potential for commercial adoption, in order to present an assessment of how practical they are as security mechanisms.
5. Finally, we discuss the criteria employed in our classification and their significance in assessing the relevant literature regarding DRAM-based implementations as security mechanisms.

We, therefore, perform, to the best of our knowledge, the first systematic classification, analysis and assessment of works regarding DRAM-based security primitives. We aim in this way to present a thorough and transparent overview of this field and provide clear insights into the current and future trends regarding DRAM-based security primitives.

The rest of this paper is organised in the following way. Section 2 provides background information concerning preliminary concepts, with a focus on the topics of DRAMs, PUFs and RNGs. In Section 3, we present a comprehensive overview and a detailed taxonomy of the literature regarding DRAM-based security primitives, based on a number of classification criteria. In this way, we allow for a thorough view into the current state of the art in the relevant scientific field. We compare DRAM-based security primitives to other hardware-based ones and examine their potential to be commercially adopted, in Section 4. Additionally, we also discuss, in this section, the significance of the classification criteria being employed in assessing the relevant literature. Finally, Section 5 contains a very brief summary of the previous sections and a few final remarks regarding future research, concluding, in this way, our overview paper.

2. Preliminary Concepts

We can distinguish three main concepts that form the background of our paper. The first one is the hardware component used, Dynamic Random Access Memories (DRAMs). The inherent properties of DRAMs are exploited in order to implement the security primitives that form the other two background concepts that this section discusses, Physical Unclonable Functions (PUFs) and Random Number Generators (RNGs). Both PUFs and RNGs have proven very useful for the implementation of cryptographic applications, especially in resource-constrained devices, such as the hardware used in the implementation of the Internet of Things (IoT). PUFs can be used for the implementation of key agreement, identification and authentication protocols, while RNGs can produce ephemeral keys and nonces, which are extremely vital for the security of cryptographic protocols.

Ephemeral keys can be produced by TRNGs, making it almost impossible for an attacker to gain access to them, as a good TRNG is extremely unlikely to produce the same output, even if an attacker gets hold of it. Additionally, such keys can be employed in a one-time pad (OTP) scheme, where they will only be used once [7]. The single usage of such ephemeral keys guarantees that the OTP scheme is information-theoretically secure and provides perfect secrecy, therefore making TRNGs really important for cryptography. Furthermore, nonces produced by TRNGs have also become a common feature of cryptographic protocols, in order to prevent replay attacks. Therefore, TRNGs are essential primitives for the secure implementation of a large number of modern cryptographic protocols.

Moreover, as PUFs ideally act as physical functions that always produce the same output for a specific input, not only their outputs can be used as keys in key agreement schemes, or identifiers for identification and authentication purposes, but they also provide the additional advantage that they do not have to be stored after they are used. This is a significant advantage, because, in this way, cryptographic protocols can be implemented without the need for additional secure storage hardware. Finally, as DRAM-based PUFs and TRNGs are based on an inherent memory module of most contemporary computer systems, they allow for the implementation of cryptographic protocols even on IoT hardware and other resource-constrained devices that do not support additional security mechanisms, such as Trusted Platform Module (TPM) implementations and other security primitives that require hardware additions.

In this section, we examine how DRAMs work and which of their characteristics have been employed for the implementation of security primitives, namely PUFs and TRNGs. We then proceed to additionally discuss basic concepts regarding these security primitives. In this way, we aim to provide insights into the way DRAM-based security primitives work, as well as a preliminary introduction to their cryptographic applications, which will be discussed in more detail in Section 3.3.

2.1. Dynamic Random Access Memories

Dynamic Random Access Memories (DRAMs) are a type of Random Access Memory (RAM) that has been incorporated into the vast majority of modern computer systems. RAM is a type of volatile memory, a memory that can only store values while it is being powered. However, Dynamic RAMs (DRAMs), unlike Static RAMs (SRAMs), not only lose their stored content when they are not being powered, but also need to have their content constantly refreshed, due to the significant leakage that is inherent to their design. On the contrary, the design of SRAMs allows them to keep their content without being refreshed, as the design of SRAM cells is such that the stored value is constantly being reinforced on its own.

However, DRAMs are one of the most widely used types of RAM, as the design of DRAM cells is very simple, lightweight and cost-efficient. Most often, DRAM cells consist of a single capacitor that stores charge above or below a certain threshold, indicating one logical value or the other, and a single gatekeeper transistor that controls access to the storage capacitor. In comparison, the most common design for SRAM cells consists of six transistors, therefore providing a clear explanation for the usual difference in the size of the DRAM and the SRAM incorporated on a system. As DRAM cells require less space and are cheaper to implement than SRAM cells, most modern commercial systems tend to have DRAM storage sizes in the range of gigabytes (GB) and SRAM storage sizes in the order of kilobytes (KB), a difference that translates in billions of DRAM cells and only thousands of SRAM cells.

DRAM cells are grouped into memory arrays, where each row is connected to a wordline, which enables access to that row. All the cells in a single column are connected to a bitline, which is used to extract the value stored on a particular DRAM cell, to which access has been enabled through the wordline. The bitlines are connected to sense amplifiers that amplify the voltage of each bitline to a level that can be interpreted as logical zero or logical one. Obviously, this setup, which is shown in Figure 1, only allows access to one row at a time, while the values of all the cells of this row can be read at the same time using the bitlines. However, the sense amplifiers work in a differential way, as they compare the voltage of two different bitlines in order to determine whether the charge of one of them has increased or decreased from the reference level stored in the other one [8].

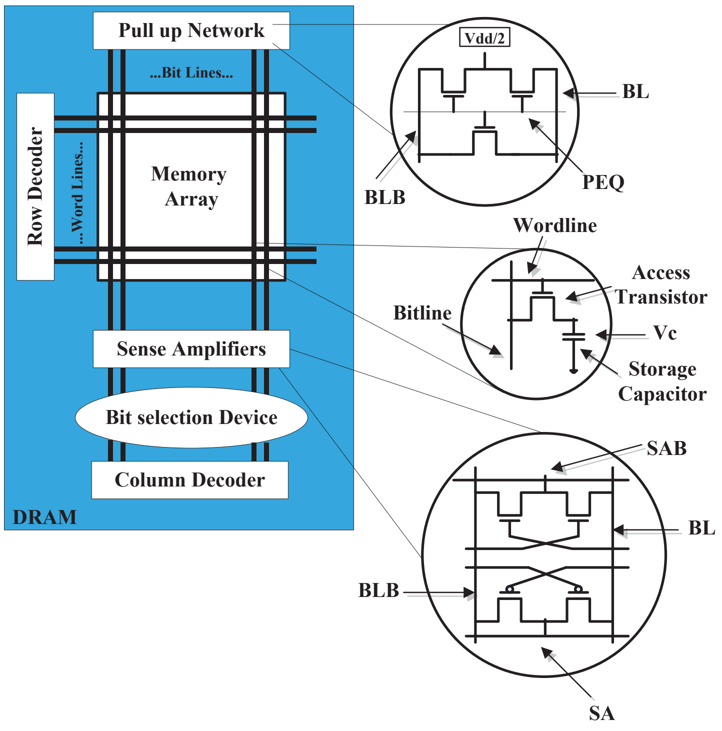


Figure 1. Memory structure of a one-transistor one-capacitor (1T1C) DRAM array.

In order for a row to be accessed, first, the bitlines are charged to $\frac{Vdd}{2}$ and, then, the relevant wordline is also charged, in order to force the row's transistors to contact and, therefore, allow access to the relevant capacitors. If a cell's capacitor is charged, charge flows from the capacitor to the bitline, causing the bitline to get slightly more charged, whereas if the capacitor is not charged, charge flows from the bitline to the capacitor, causing the bitline to become slightly less charged. Then, a sense amplifier compares the charge of this bitline to the reference $\frac{Vdd}{2}$ charge stored in another bitline, and amplifies their difference. This leads into the cell's value being recognised as logical one or logical zero, when the cells of a row are being read. In order to facilitate the differential amplification process, DRAM cells are usually organised into two different categories; *true cells*, whose charged capacitor indicates a value of logical one, and *anti-cells*, which store a value of logical one when their capacitor is discharged [8]. For both categories of cells, the opposite state of their capacitors indicates a value of logical zero.

This procedure also helps refresh the values stored on the DRAM cells, as the bitlines being compared are forced to exchange charges [9]. This happens in such a way that either will discharge the cell's capacitor, if the relevant bitline is charged below the $\frac{Vdd}{2}$ charge stored in the other bitline, or will charge the capacitor, if the relevant bitline is charged above the $\frac{Vdd}{2}$ charge stored in the other one. As DRAM cells store their charge on a capacitor, their charge will inevitably leak unless it is regularly replenished. Therefore, an automated process exists in order to refresh the values stored in all the DRAM cells, within such a time interval that even the most leaking cell will always be able to provide the correct value. The refresh process is almost identical to reading, but the logical values of the DRAM cells are not used further.

Writing usually happens in a similar way to reading, with the difference being that while a row is being accessed, particular bitlines are forced to potentially different charges, in order for the correct values to be written in the cells, through the previously described operation of the sense amplifiers [9]. After a row has been accessed, either for writing, reading or to be refreshed, the relevant wordline is discharged, therefore trapping the charge of the capacitor in the cell, until it either leaks or is refreshed.

However, apart from DRAMs with cells consisting of one transistor and one capacitor (1T1C), as the one shown in Figure 1, there are also other DRAM implementations, such as purpose-built embedded DRAMs, with cells consisting of only two transistors (2T) [10,11]. In these, the storage capacitor is replaced by a storage transistor connected to a transistor that allows access for writing. In these 2T DRAM cells, writing and reading are decoupled, therefore requiring two different sets of a wordline and a bitline each, one set of which is used for writing and the other for reading [10]. This design is quite similar to the one used in early DRAM cells, which consisted of three transistors (3T), one used for write access, one for read access and one for storage. This design again required two wordlines and two bitlines for each cell, as one set of a wordline and a bitline was used for writing and the other for reading [9]. Nevertheless, all of these designs again require frequent refreshing of the values stored in their DRAM cells, due to leakages.

2.2. DRAM-Based Security Primitives

Hardware-based security primitives exploit inherent characteristics of hardware in order to extract entropy, in the form of random and unique outputs, which can be used for the implementation of cryptographic solutions. These outputs can be robust, in which case they can be used for identification and authentication, or unstable, in which case they can be used in cryptography as random one-time keys and nonces. Manufacturing variations cause minor imperfections in commodity hardware, such as DRAMs, which can serve to create inherent characteristics in a random and unique way in each individual hardware instance. Therefore, while such imperfections do not affect the correct operation of hardware, they can be exploited for the implementation of security primitives. Researchers have identified a number of inherent characteristics in DRAMs that can be leveraged for the implementation of hardware-based security primitives.

In particular, it has been noted that the cells of DRAMs may assume different values at startup [12], in a similar way to SRAM cells. In this case, the startup charge of the capacitor of each individual cell (in a 1T1C DRAM) may be slightly above the $\frac{Vdd}{2}$ threshold value (Figure 2a), or slightly below it (Figure 2b), a situation that leads to the cell's value being interpreted as logical one or logical zero, respectively, if the cell is a true cell, and vice versa, if the cell is an anti-cell. Figure 2 provides an overview of how voltage moves, in each of the two cases, through the DRAM circuitry presented in Figure 1, when the cells are read. The cells are previously untouched, not having been written to, by the system or the user.

Another characteristic of DRAM cells that can be exploited for the extraction of entropy is their retention times. As already mentioned before, DRAM cells need to be frequently refreshed or the charge of their capacitors leaks away, causing the values stored in them to flip. However, the decay characteristic is not the same for all the cells of a particular DRAM, with cells exhibiting different periods of charge remanence. This phenomenon can be exploited in a number of different ways, resulting in different degrees of instability. Researchers have either stopped the refresh function of the DRAM [13–16], exploiting the retention of different cells, or cut the power of the whole chip [17], taking advantage of the data remanence of the cells, or even forced different low voltages in the wordlines, in order to intensify the decay effect [18].

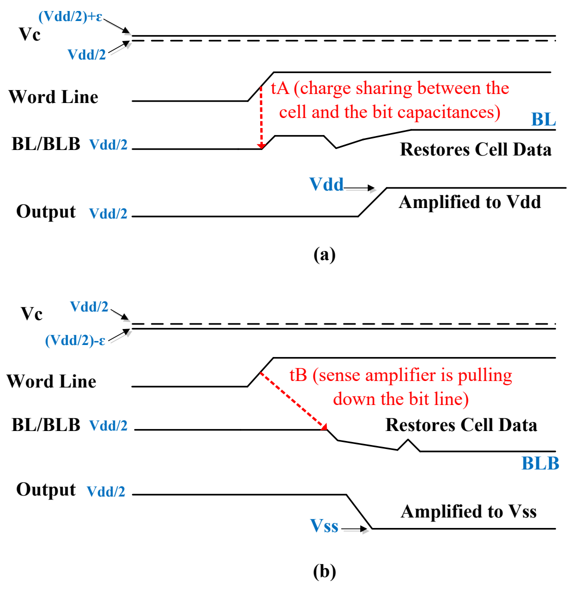


Figure 2. Timing diagram of a read operation of a previously untouched DRAM cell, for cells biased to Vdd (**a**) or Vss (**b**) at startup due to process variations. In this case, Vdd signifies the supply voltage and Vss the ground voltage.

Furthermore, in order to increase the number of flips exhibited over time, a row hammer process can be used [5]. In this case, rows are constantly written alternately with zeros and ones, causing the values of cells in other rows to flip, due to excessive leakage. In order to maximise the effects of row hammering, the rows constantly changing value (hammer rows) alternate with the rows that will be used for the implementation of the security primitive (primitive rows), as shown in Figure 3.

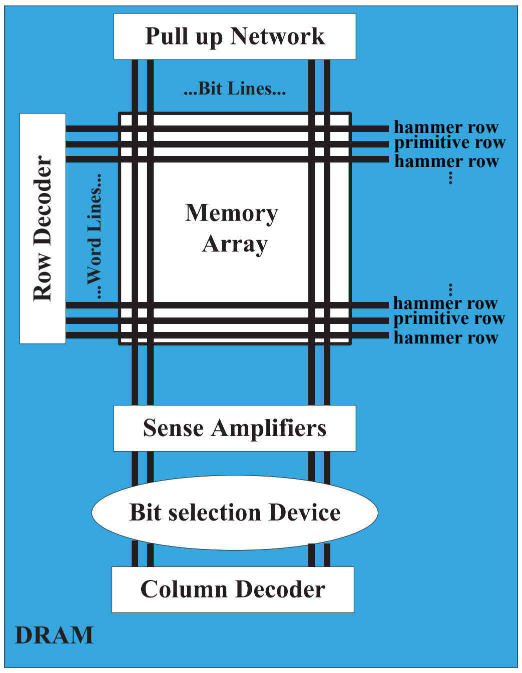


Figure 3. The alternating order between rows used for row hammering (hammer rows) and rows used for the implementation of a security primitive (primitive rows).

It is, therefore, evident that the retention characteristic of DRAM cells is highly dependent on the leakage of their capacitors, which can be affected by nearby components, such as other cells in the same or in other rows. Figure 4 presents some of the different leakage paths, showing with red colour paths in the same row, and with blue, paths affecting cells in other rows. Note that the leakage of a cell can also affect cells in non-adjacent rows, and can affect nearby bitlines and wordlines, as well as the transistors and capacitors of other cells, as shown in Figure 4.

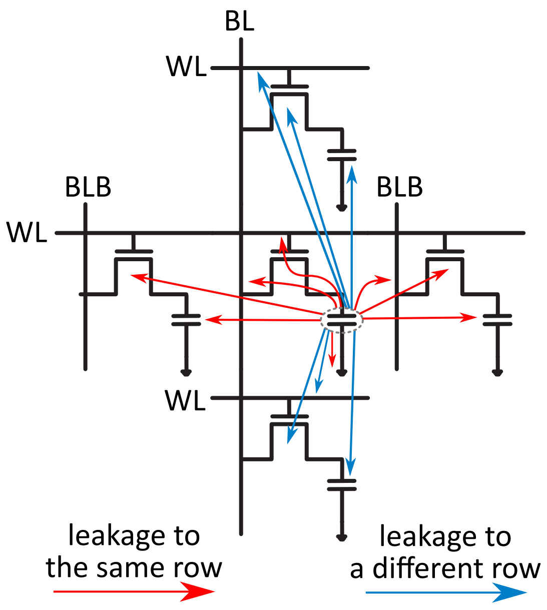


Figure 4. Some of the potential leakage paths for a single cell's capacitor. Charge can leak to components in the same row (red lines), or even in other rows (blue lines), especially in the case of row hammering.

We need to mention that DRAM cells also exhibit a Variable Retention Time (VRT) phenomenon, which means that the same cell may randomly switch between a high retention state (corresponding to a high retention time) and a low retention state (corresponding to a low retention time) at different points in time [8,19]. This behaviour depends on the amount of unoccupied traps that exist in the gate region of the transistor at a given point in time, which can absorb some of the draining charge, and is related to the manufacturing process of the cell's transistor. This phenomenon can add to the instability of the retention characteristic of DRAM cells.

Moreover, another characteristic of DRAM cells that has been exploited for the implementation of security primitives is the time required for the read and write operations to be successful [20,21]. If this time is decreased, some cells will fail to be read or written, while others will still be read or written successfully, depending on slight variations in their transistors and capacitors. Obviously, however, for the output of the DRAM to be truly random, the time allowed for the read and the write operations must not be set too high or too low, as this would cause the vast majority of cells to be read and written successfully or to fail to be read and written, respectively. Changing the time parameters, such as the access latency, used for the read and write operations of a DRAM can be done either with the introduction of a delay component [20], or by manipulating their values using software commands, when this is possible [21].

We should, finally, note that, based on the stability of the DRAM characteristics exploited, different security primitives can be implemented. If the characteristic exhibits a high degree of stability, then a Physical Unclonable Function (PUF) can be implemented out of it, which can be used for identification, authentication or even as a secure key storage. Even if the characteristic is not fully stable, error correction can be applied so that the DRAM-based PUF can truly act as a function, which always provides the same output for a specific input. Otherwise, if the characteristic is highly unstable, it can be leveraged for the production of a random number generator. In this case, we want the output to be as unstable as possible, so that we can ensure that the numbers (actually, bitstrings) that are produced will be highly random. In both cases, however, we require that the characteristic holds enough randomness to be unpredictable and is quite unique per DRAM instance, in order to prevent trivial attacks against the security primitives being implemented.

2.2.1. Physical Unclonable Functions

As already mentioned, Physical Unclonable Functions (PUFs) act as functions encoded in hardware, which produce a unique output, being referred to as a response, for a specific input, being called a challenge. PUFs provide a varying level of security, and can, therefore, be used in different applications, depending on the number of their available input–output pairs, which are referred to as Challenge-Response Pairs (CRPs). For example, a PUF with only a single challenge-response pair can be used for identification, while a PUF with multiple CRPs can be used to provide multiple different session keys for authentication. In the first case, the response needs to be secret, while, in the second one, responses can be also used without any secrecy, as long as the related CRPs are not used again.

Although a number of publications considered the usage of physical characteristics for identification, authentication and anti-counterfeiting [22–24], 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 [25,26]. Nevertheless, 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 PUF came in 2002 [27,28], being reported in the same publications that also proposed the term “physical unclonable function” in order to describe a hardware-entangled function that produces outputs that are unique for each hardware instance. However, again, the primitives proposed (arbiter PUFs) were dedicated hardware components that, although they were 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 [29,30], which was the first PUF that not only was implemented in silicon, but was also based on a hardware component, SRAM, that was part of most contemporary computer systems, and which could also be evaluated without the need of additional hardware. Nevertheless, the SRAM PUF is based on the startup 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 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” [31,32].

However, at the same time, modelling and machine-learning attacks against the so-called “strong” PUFs [33–36], forced the development of newer more resistant implementations, such as the XOR (eXclusive OR) 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 will also be 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. It is for this reason that the majority of the most recent PUF implementations are based on memory elements, such as SRAMs, DRAMs and Flash memories.

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”. Nevertheless, the ring oscillator PUF, a delay-based PUF, has been described in some publications as a “weak” PUF [32] and in others as a “strong” one [35]. Additionally, as we discuss in other sections, DRAM PUFs can quite often provide multiple CRPs, and even a very large number of them [11]. However, in this work, we refrain from judging whether the different DRAM-based PUFs are “weak” or “strong”, and only examine the amount of CRPs that they can potentially allow for, which we sometimes proceed to compare to the single CRP that most SRAM PUF implementations provide.

Regarding the prevalence of memory-based PUF implementations in recent publications, this is clearly demonstrated in Table 1, which presents a brief history regarding the 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, in order to present this information to the reader in a way that is very simple to understand without additional effort.

Table 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	Publications
2001–2002	Optical PUF [25,26]
2002–2003	Arbiter PUF [27,28]
2004	Feed-Forward Arbiter PUF [33]
2006	Coating PUF [37]
2007	SRAM PUF [29,30], Ring Oscillator PUF [38], Latch PUF [39], XOR Arbiter PUF [38]
2008	Lightweight Arbiter PUF [40], Butterfly PUF [41], Flip-Flop PUF [42]
2010	Glitch PUF [43]
2011	Flash PUF [6], Current-based PUF [44], Bistable Ring PUF [45]
2012	Buskeeper PUF [46], DRAM Decay PUF [13,14,18]
2014	Bitline PUF [47], Transient Effect Ring Oscillator PUF [48]
2015	DRAM Startup PUF [12], DRAM Latency PUF [20]
2016	MEMS PUF [49]
2017	Row Hammer PUF [5]
	Memory-based PUFs Delay-based PUFs Other PUFs

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 [2,7]. Additionally, DRAM PUFs based on the startup 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 [3]. 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 [4,5]. Furthermore, the most recent implementation of the DRAM latency-based PUF can offer a significantly lower evaluation time than the DRAM decay-based PUF, therefore making evaluation practical in all cases [21]. Finally, the row hammer PUF is an implementation that can significantly increase the entropy extracted from the decay characteristic of DRAMs [5].

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 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 startup values of the DRAM can be considered to be able to provide a number of CRPs, if 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. It is therefore essential to investigate DRAM PUFs in detail, in order to shed light on the current state of the art, as well as provide insights into their full potential.

2.2.2. Random Number Generators

Since antiquity, humans have been interested in the generation of random numbers for a plenitude of reasons, ranging from gambling to cryptography and statistics. Random number generation can be either truly random, when a chaotic system is being used, or pseudorandom, when a seed is being used to generate a sequence that appears random, but is repeatable using the same seed as a basis for the generation of a future sequence. A chaotic system, on the contrary, is based on so many different factors that the sequences generated by it are not fully predictable. Such a system can often be modelled or simulated, up to some degree, but its exact output is not fully predictable. For example, throwing dice or flipping a coin can be modelled and simulated, but the results of these actions are not fully predictable because the exact circumstances under which these actions may be performed are extremely hard to predict.

However, if we know the exact force with which a coin is being flipped, on which exact spot on the coin this force has been placed, what was the initial state of the coin, what are characteristics of the coin, such as its mass, the material of which it is made, etc., and, also, that no other forces affect the coin, then perhaps the outcome of the coin flipping can be predicted. In exactly the same way, if we know the seed of a number generator and its exact algorithm, we can then usually predict the exact output of it. Therefore, the generation of random numbers is highly dependent on the system being used for it and, more specifically, on how deterministic this system is.

A non-deterministic system will produce highly unpredictable outputs and can therefore be used for the implementation of *True Random Number Generators* (TRNGs), while a simple system that is also fully deterministic will produce outputs that are highly predictable and therefore not suitable for random number generation. Finally, a complex deterministic system will produce output that is, up to some degree, predictable, but does appear random, and can therefore be used for the implementation of a *PseudoRandom Number Generator* (PRNG). Therefore, the degree of complexity of a system and its determinism have to be assessed, in order to determine the ability of a system to be used for random number generation. Nevertheless, this task can prove to be much more complex than assessing the output of such a system, for its randomness, unpredictability, etc. For this reason, researchers have developed tests that can indeed examine the suitability of a system to act as a Random Number Generator (RNG) based on its outputs. One of the most popular suites of such tests has been produced by the National Institute of Standards and Technology (NIST) in the United States of America (USA) [50].

Both TRNGs and PRNGs may pass such statistical tests successfully, but once the initial seed of a PRNG, including its initial state, is known, all its outputs can be predicted successfully. TRNGs are therefore much more secure, being completely unpredictable, and are, therefore, more suitable for applications related to security, such as data encryption. However, PRNGs are also very popular due to their flexible and low-cost implementations and their low generation times [51]. Recent research has, therefore, focused on the implementation of cost-efficient and flexible TRNGs, which could be employed as security primitives, mainly in cryptographic protocols. Such TRNGs could be implemented using commercial hardware, such as DRAMs, which is not only widely available, but would also not require additional dedicated hardware, which would raise the cost of production.

Such implementations would enable the use of intrinsic TRNGs that would be inherently available even in resource-constrained devices, such as IoT hardware. As already noted, TRNGs are particularly useful for the production of ephemeral, one-time keys and nonces, which can allow the implementation of complex cryptographic protocols. For this reason, intrinsic TRNGs could enable the usage of complex cryptography even on low-end devices. Therefore, DRAM-based TRNGs could prove to be a major breakthrough in the history of hardware-based TRNGs.

As software can only very rarely prove to be unpredictable, most TRNG implementation in modern computer systems use hardware in order to generate random numbers. A large number of hardware components, such as power supplies [52,53], Zener diodes and delay and clocking elements, which inherently carry an increased noise load, have been utilised for the implementation of TRNGs.

However, the implementation of such TRNGs usually requires additional circuitry or other hardware, either for the construction of the TRNG components or for the evaluation of the TRNG hardware. As a result, not only such TRNGs may increase the production costs, but also have an increased generation time, as multiple components may be required for their construction and evaluation. On the contrary, an intrinsic silicon TRNG, such as a DRAM-based TRNG, can provide increased security in comparison to a PRNG, while providing low generation times and requiring only dedicated software, a need that is also common to most PRNGs.

Recent research has proven that the characteristics of DRAMs may, in some cases, provide outputs that are random and unique per DRAM instance, but also highly unstable, and can therefore be used for the implementation of intrinsic hardware TRNGs in commodity devices. In particular, the potential of the retention time of DRAM cells to serve as a source of entropy for the implementation of a TRNG had been noted even in the very first publications regarding the decay characteristic of the DRAM cells as a basis for the implementation of security primitives [15,16]. Additionally, the data remanence of DRAM cells has also been investigated as a characteristic that can be exploited for the creation of a robust DRAM-based TRNG [17,54]. Finally, a more recent publication studies the potential of the startup values of DRAM cells to be employed for the production of a TRNG, and the difficulties in the realisation of such a scheme [55]. Therefore, as recent research [19] keeps providing promising results regarding the potential of DRAMs to serve as a basis for the formation of cost-efficient, fast and flexible TRNGs, we believe that it is crucial to investigate in detail the current state of the art regarding DRAM-based TRNGs and their future potential for cryptographic applications.

3. Overview of the Current State of the Art Regarding DRAM-Based Security Primitives

As already mentioned in the previous section, the recent development of DRAM-based security primitives can potentially lead to significant advances in the field of IT security, because such primitives not only can be implemented in a cost-efficient manner, as DRAMs constitute intrinsic components of most contemporary computer systems, but also, quite often, can be used at run-time. For this reason, in this section, we investigate the relevant literature, provide a thorough overview of it and classify it, in such a way as to provide a clear picture of the current state of the art and useful insights regarding potential future developments.

Our overview and classification scheme have been inspired by relevant publications containing classifications and taxonomies of other hardware-based security primitives and mechanisms. We briefly cite a very small number of such overview papers on the security of smart grids and smart homes [56,57], on the security of Radio-Frequency IDentification (RFID) [58,59], on IoT infrastructure [60–63], on PUFs [32,64–71] and on hardware Trojans [72–74], which clearly demonstrate the diversity of hardware security mechanisms that have so far been examined.

3.1. Brief Literature Taxonomy

As Table 2 reveals, there has been an increased interest regarding DRAM-based security primitives in previous years. This is also evident from the publication of two doctoral dissertations in 2017 that consider DRAM-based security primitives to a significant extent [54,75]. Additionally, other doctoral dissertations also discuss DRAM-based security primitives [76,77], exhibiting that these primitives are becoming well-known. Finally, the existence of other publications from Eastern Europe [78,79] and Asia [13,77] are further indicators of the global outreach of the topic of DRAM-based security primitives.

It is, therefore, important to examine in some detail the most influential publications regarding DRAM-based security primitives, in order to identify their most important contributions and, in this way, expose the current state of the art of this scientific field. We choose to do so in a chronological order, as Table 2 shows, in order to allow the reader to view how this field has evolved and get some very first insights into its potential future development.

As it has already been mentioned, there are four major characteristics and effects related to DRAMs that have so far been exploited for the implementation of a DRAM-based security primitive.

These are DRAM decay and data remanence effects, the startup values of the DRAM cells, DRAM access latencies and the effect of applying a row hammer algorithm on the rows of a DRAM array. We, therefore, choose to also classify the publications presented in Table 2 according to the exploited characteristic(s), which they refer to, in order to provide a more detailed view of the way in which DRAM-based security primitives were developed. We choose a colour classification, in order to make the results of our classification easier for the reader to understand. As some publications consider two characteristics, we indicate those by using both relevant colour indicators in Table 2.

Table 2. History of the development of DRAM-based security primitives. The publications examined are colour-coded according to the DRAM characteristic being exploited for the implementation of the security primitives discussed in them.

Year	Total Amount of Publications	Publications
2012	4	Okamura et al. [13], Fainstein et al. [18], Keller et al. [14] (poster), Felber [15] (presentation slides)
2013	2	Rosenblatt et al. [80], Rosenblatt et al. [81]
2014	3	Liu et al. [82], Liu et al. [83], Keller et al. [16]
2015	4	Tehranipoor et al. [12], Hashemian et al. [20], Zhang [84], Rahmati et al. [85]
2016	4	Tehranipoor et al. [17], Sutar et al. [86], Vitenko [78], Xiong et al. [4]
2017	10	Sutar et al. [87], Tehranipoor et al. [88], Tang et al. [11], Eckert et al. [55], Tehranipoor et al. [3], Schaller [75], Tehranipoor [54], Kumar et al. [89], Kirihata et al. [90], Schaller et al. [5]
2018 (Jan)	2	Kim et al. [21], Sutar et al. [19]
DRAM natural decay effect DRAM intensified decay effect due to VWL DRAM startup values DRAM data remanence effect DRAM row hammer effect DRAM access latency		

3.2. Overview of the Literature Regarding DRAM-Based Security Primitives

As Table 2 shows, the first publications regarding DRAM-based security primitives have occurred in 2012 and all were considering DRAM PUF implementations based on the retention characteristic of DRAMs. Since then, however, the amount of publications per year seems to be stable, with a sudden increase in 2017, which may also be continued in 2018. Additionally, the number of DRAM characteristics employed for the implementation of security primitives has also increased, and may further increase in the future. An overview of the current literature regarding DRAM-based security primitives follows.

In 2012, Okamura et al. [13] published a paper regarding the usage of the retention characteristic of a DRAM in order to produce a PUF. They also examined the effect of temperature on this characteristic, with a focus on the effects of different temperatures on the number of bits flipping. They compared PUF responses measured at 25 °C to responses measured and −5 °C, 45 °C and 65 °C, showing that the number of flipped bits increases, for higher temperatures and higher decay times, during which the DRAM is not being refreshed. Finally, they showed that unique 192-bit keys can be generated within 30 s and 672-bit keys within 60 s.

Fainstein et al. [18] published a paper describing a Retention-based Intrinsic Chip ID (RICID) using embedded DRAM (eDRAM). They demonstrated intrinsic chip identification using a 32 nm Silicon-On-Insulator (SOI) high-κ/metal gate embedded DRAM, which provided a high probability of ID uniqueness and recognition. Their technique constitutes a decay-based DRAM PUF, where the decay is intensified by using higher wordline low voltages (VWLs). They used a bit fail map of a

DRAM array, in which retention pass or fail bits (represented as logical zeros or ones, respectively) are used to generate a pair of binary strings A and B using different VWLs. Using a higher VWL for B results in a greater number of retention fails such that the list of failing cell positions includes all of the positions that failed in A and new ones because more cells leak enough for their logical value to flip.

Keller et al. [14] presented a poster about the usage of the retention characteristic for the implementation of a DRAM PUF, and Felber [15] made also presentation slides for the same implementation and its usage in a Quantum Key Distribution (QKD) system.

In 2013, Rosenblatt et al. [80] extended the RICID idea published by Fainstein et al. [18] using the same DRAM PUF implementation in order to include field-tolerant authentication by detecting a number of bits that can ensure successful recognition and also prevent ID spoofing during the read operation. This allowed for a 100% recognition rate of the unique chip IDs generated under different temperature and voltage conditions, using more than 400,000 ID pair comparisons in more than 250 chips. Successful recognition rates were predicted to be very close to 100% using an analytical model for one million different pairs.

In a different publication in the same year, Rosenblatt et al. [81] presented an architecture that enables self-authentication in chips, using electrically programmable fuses (eFUSEs) that store PUF responses produced by an eDRAM. Again, the DRAM PUF is based on the retention times of the DRAM cells, which can be lowered using higher VWLs. The responses produced are proven to be unique using Monte Carlo simulations and the chips can be authenticated even when the responses are noisy. The simulation results are confirmed by testing more than 50 chips and an analytical model is used to predict that attacking the DRAM PUF using brute force would require more than 10 years to be successful and that successful authentication rates are very close to 100% using an analytical model for one million different chips.

In 2014, Liu et al. [82] published a paper regarding a DRAM decay PUF based on a Double-Data-Rate type three (DDR3) Synchronous DRAM (SDRAM). They also described an algorithm for key generation, with an integrated fuzzy extractor, which can produce secure 128-bit keys. Due to the wide adoption of DDR3 memory in wireless sensor networks (WSNs), and as the proposed DRAM PUF technology provides high security and does not require hardware changes, the proposed PUF is suitable for usage in WSN applications. The feasibility of such a PUF is validated using an Field-Programmable Gate Array (FPGA) board and a number of removable Small Outline Dual In-line Memory Module (SODIMM) DRAMs. An extended version of this paper was also published the same year by Liu et al. [83], in which a more detailed overview of the algorithm, including a thorough description of the fuzzy extractor parameters, is presented. Both papers present positive results regarding the decay rate and the intra-device and inter-device Hamming distances.

Keller et al. [16] also presented a DRAM decay PUF based on a DDR3 SODIMM DRAM connected to an FPGA board, which can, however, also be used for the generation of random numbers. A memory controller that permanently disables the refresh operation has been implemented. The authors note that the implemented DRAM PUF allows for repeated queries at run-time, in contrast to an SRAM PUF. Furthermore, utilising a 512 MB DDR3 module, 400 random bits can be produced per second.

In 2015, Tehranipoor et al. [12] introduced a DRAM PUF based on the startup values of the DRAM cells, which works in a similar way to the SRAM PUF. Nevertheless, due to the size of the DRAM, this PUF allows for a much larger response size. It relies on the fact that DRAM cells initialise to random values at startup because of variations in their manufacturing process, which are however too small to affect their regular operation. Different operating conditions, such as different voltage and temperature levels are investigated in order to test the PUF's reliability. The most stable cells are selected during enrollment for further use.

Hashemian et al. [20] presented in the same year a DRAM PUF implementation based on the write access failures. An external delay circuit can be implemented in order to change the time parameter of the write operation, in such a way that a number of cells fail to be written. Monte Carlo simulations were performed using an implementation of a Simulation Program with Integrated

Circuit Emphasis (SPICE), a program used for the simulation of hardware. These simulations showed that, for 1000 chips with 10% inter-die and 8% intra-die variation, the proposed PUF exhibits high uniqueness with a 50.01% average inter-die Hamming distance and good robustness under temporal fluctuations in the supply voltage and the temperature and under aging effects for its estimated 10-year lifetime. The PUF also provides good reproducibility with a 7% average intra-die Hamming distance for temperatures ranging from 20 °C to 50 °C and 7.4% average intra–die Hamming distance for supply voltages ranging from 0.9 to 1.2 volts. Furthermore, aging analysis shows that under Negative-Bias Temperature Instability (NBTI) effects, unstable bits constitute on average only 0.92% of the PUF responses over the estimated 10-year lifetime.

Zhang [84] published his master's thesis on the same year, which is highly related to the papers by Liu et al. [82,83]. The same PUF implementation is presented, along with an extended related literature section, additional sections on the applications of PUFs and more information regarding the fuzzy extractor scheme and the metrics used to evaluate this DRAM decay PUF. The results prove that this PUF implementation can serve for secure key generation.

Rahmati et al. [85] presented what can be characterised as an "attack" PUF implementation, as their paper explored how the decay characteristics of DRAM cells in approximate DRAM, a DRAM that does not provide 100% stability of its stored values can create an error pattern that could serve as a system identifying fingerprint. Approximate DRAM allows for some values to decay, as its refreshing time is not adequate enough for all the cells to be refreshed before they decay. Therefore, the values stored in the cells with the lowest retention times will fade away, creating a unique pattern of error, which may not affect significantly the content stored over all the DRAM array, but can lead to successful recognition and de-anonymisation. This publication was followed by others that focus more on the optimisation of the quality of an approximate DRAM [91,92].

In 2016, Tehranipoor et al. [17] published a paper presenting a hardware TRNG based on the data remanence effect of a DRAM, whereby information remains in a DRAM even after powering it down. This phenomenon is practically another version of the cell decay characteristic. Nevertheless, as in this case, the whole DRAM, or even the whole chip, is powered down, a slightly different behaviour may appear, which is also dependent on *burn-in* and *grounding* effects, i.e., the time that a value has been stored on a cell and whether the overall circuit is grounded or not. Additionally, the startup behaviour of the cells can play a role because, if the cells decay completely during the period of time that the DRAM is powered off, they will probably assume their startup values when the DRAM is again powered on. Therefore, this implementation is inclined to exhibit high instability, which makes it fit to serve as an TRNG. An FPGA and its on-board DRAM are used for testing, together with a power-cycle circuit used to enable and disable the DRAM.

Sutar et al. [86] published a paper on a DRAM decay PUF that is intrinsically reconfigurable, supporting a large number of challenge-response pairs (CRPs) through the variation of different parameters. Their design is implemented and validated using an FPGA and removable SODIMM DDR3 DRAM modules. Their design also allows for reduced authentication times in comparison to previous works. Additionally, the authors tested the robustness of their authentication mechanism under temperature variations and aging effects. Their results demonstrate a 100% true-positive (successful authentication) rate for a 10 °C temperature variation with a 0% false-positive rate. For a nine-month old DRAM module, their authentication mechanism also provided a 100% true-positive rate and a 0% false-positive rate.

Vitenko [78] presented a rather interesting implementation of a DRAM PUF based on the retention characteristic of DRAM cells, as it was the first implementation on a truly commercial device. The DRAM of a tablet computer produced by LG, a well-known electronics company that has its headquarters in Seoul, South Korea, was used in order to implement a PUF and the relevant DRAM module was controlled using the Android OS. The evaluation of the responses of the PUF proves that it can be used for identification purposes, as they are proven to be random, unique and stable enough. This paper proves in a definitive way that DRAM PUFs can be used commercially.

Xiong et al. [4] implemented a DRAM decay PUF that can be queried at run-time on two commercially available evaluation boards using a Linux kernel module, therefore exhibiting that DRAM PUFs can be implemented and used in IoT hardware. Additionally, they also presented a firmware implementation of the same PUF on both devices using the firmware of these devices. However, this implementation cannot be accessed at run-time. Finally, they also presented lightweight cryptographic protocols based on their PUF implementations, which can be used for device authentication and secure channel establishment.

In 2017, Sutar et al. [87] presented a memory-based combination PUF implementation that combined an SRAM PUF with a DRAM decay PUF. This implementation can potentially combine the advantages of both types of memory-based PUFs and address the shortcomings of each individual PUF type. This implementation addresses potential problems of the two different PUF types, such as the DRAM module being removable, in which case, it can easily be transferred to another system for the implementation of a number of different attacks against its security. The proposed PUF exhibits high entropy, supports a large number of challenge-response pairs, and is intrinsically reconfigurable. The SRAM PUF utilizes a power-cycling approach to generate startup values as responses, while the DRAM PUF responses consist of unique bit-flip patterns generated through a refresh-pausing approach. The combination PUF has been implemented using an FPGA and several off-the-shelf SRAMs and DRAMs. Their experimental results demonstrate substantial improvements over current memory-based PUFs including the ability to resist various attacks. Extensive authentication tests across a wide temperature range (20–60 °C) and accelerated aging (12 months) demonstrate the robustness of the proposed design, which achieves a 100% true-positive rate and a 0% false-positive rate for authentication across these parameter ranges.

Tehranipoor et al. [88] published an investigation of the reliability of the responses of a DRAM PUF based on the startup values of its DRAM cells under device-accelerated aging effects. Their paper presents accelerated aging experimental results over 18 months on three DRAM PUFs based on startup values. Experiments were conducted using three Commercial Off-The-Shelf (COTS) DRAM modules. Additionally, in order to accelerate aging, a thermal inducting system was employed. Based on their results, the effect of NBTI stress appears to be negligible on DRAMs and can be ignored. In other words, this type of DRAM PUFs is resistant to aging, and can, therefore, improve the performance and reliability of the entire system during its operational life.

Tang et al. [11] presented a DRAM PUF based on the retention characteristic of DRAM cells, utilising the location of cells with a weak retention characteristic in a 65 nm 2T Complementary Metal–Oxide–Semiconductor (CMOS) eDRAM. In this case, multiple locations of the DRAM are utilised to produce the PUF's final response, as their error patterns are combined in the final response. This scheme allows for the creation of more than 10^{32} unique CRPs from a 1 Kbit eDRAM array. The responses become more stable through the application of a zero-overhead repetitive write-back technique along with bitmasking, while instabilities induced by voltage and temperature variation are mitigated by adjusting the read reference voltage and refresh time before each authentication operation. Finally, the area of each cell of the proposed DRAM PUF is calculated to be less than 1 μm^2.

Eckert et al. [55] published a paper dealing with a hardware TRNG implementation based on the startup values of Double-Data-Rate type two (DDR2) DRAM. In this paper, they show that the startup values of newer DRAMs may not suitable to serve as a PUF, a conclusion that was also reached by Anagnostopoulos et al. [93]. However, by combining multiple measurements through an XOR operation and by using a von Neumann corrector, the authors manage to implement a TRNG. The authors assess the randomness of their scheme using the NIST suite of tests and prove that it can be used as a TRNG. We note here that the eXclusive OR (XOR) logical operation provides a logical output of zero when all its inputs have the same logical value and of one when at least one of them has a different logical values from the others.

Tehranipoor et al. [3] extended the previous publication by Tehranipoor et al. [12] regarding a DRAM PUF based on the startup values of the DRAM cells. Their approach has a relatively low generation time and no additional hardware is required. The effect of different operating conditions, based on voltage and temperature variations and aging is examined. Their implementation consists of external Dual In-line Package (DIP) DRAM modules connected to an FPGA board, and provides highly distinct PUF responses with an average inter-die Hamming distance of 0.4937, which can be utilised to construct 128-bit secure keys.

Schaller [75] published his doctoral dissertation in 2017, in which he examines PUF implementations on COTS hardware and their applications. He discusses in detail DRAM PUFs based on both the retention characteristic of DRAM cells and the row hammer effect, as well as their applications, and potential protocols based on them. His dissertation proves that DRAM and other memory-based PUFs can be implemented on COTS hardware and serve as security primitives for the implementation of lightweight cryptographic protocols.

Tehranipoor [54] also presented her doctoral dissertation in 2017, in which DRAM-based PUFs and TRNGs are presented and discussed in detail. The overall dissertation is considering the design and architecture of various hardware-based TRNGs and PUFs that meet the requirements and needs for low-cost solutions in today's electronic systems. For this reason, particular focus is placed upon DRAM-based security primitives. Additionally, this dissertation also deals with the architectural requirements for designing resource-efficient embedded system platforms. Works presented in this dissertation include a DRAM PUF based on startup values and two DRAM-based TRNGs, one that is based on the startup values of DRAM cells and another that is based on their data remanence.

Kumar et al. [89] published a book chapter, a part of which deals with hardware security based on intrinsic IDs produced by eDRAM. These IDs are produced by eDRAM based on the retention of its cells, which is lowered by using higher VWLs. The technique being employed is very similar to the ones used by Rosenblatt et al. [80] and Rosenblatt et al. [81]. The authors prove that this implementation can be successfully employed for the authentication of a chip.

Kirihata et al. [90] also published a book chapter, in which they also discuss a DRAM PUF implementation based on the retention of its cells, which is lowered by using higher VWLs. The scheme being employed is again very similar to the ones used by Rosenblatt et al. [80] and Rosenblatt et al. [81], and is used for anti-counterfeiting purposes. This scheme has the advantage of being based on an intrinsic element, the DRAM, thus not requiring additional hardware for its implementation, and therefore being highly cost-efficient.

In general, we must note that the publications by Fainstein et al. [18], Rosenblatt et al. [80], Rosenblatt et al. [81], Kumar et al. [89] and Kirihata et al. [90] form a family of publications investigating the potential of eDRAM for the implementation of a DRAM PUF based on the retention characteristic of its cells, which is artificially lowered by using higher VWLs. To this end, these publications examine in detail the applications and security of such a primitive, in order to determine its suitability to serve as the basis of lightweight cryptographic protocols.

Schaller et al. [5] presented a paper regarding the potential of the row hammer effect to be used in order to enhance the uniqueness and randomness of a DRAM PUF, if it is used in conjunction to the decay characteristic of the DRAM cells that is exhibited when the refresh operation of the DRAM is paused. Their implementation on COTS evaluation boards proved that generation times can be significantly decreased in this way, leading to a PUF that is accessible at run-time and can be used in a quick manner. Additionally, this was the first positive application of the DRAM row hammer effect.

In 2018, Kim et al. [21]presented an implementation of a DRAM PUF based on the latency of the read and write operations of the DRAM, and proved that it has a significantly lower generation time in comparison to the DRAM decay PUF. For this purpose, they implemented both DRAM decay-based and DRAM latency-based PUFs on a large number of Double-Data-Rate type four (DDR4) DRAM modules connected to an evaluation board. Their implementation of the DRAM latency PUF requires no additional hardware for measurements, in contrast to the implementation of the same type of PUF by Hashemian et al. [20], as they use memory controllers that allow them to change the time parameters of the read and write operations of the DRAM through software means. Their implementation is accessible at run-time and the effects of temperature on it have been investigated, proving a significant decrease of generation times for the DRAM latency PUF in comparison to the DRAM decay PUF at all temperatures.

Finally, also in 2018, Sutar et al. [19] extended the previous paper by Sutar et al. [86] regarding a reconfigurable DRAM decay PUF, by also presenting a TRNG based on the DRAM decay PUF and the VRT phenomenon. Their implementation was based on several COTS DDR3 DRAM modules being connected to an FPGA board and was proven to be around five times faster than other prior DRAM PUFs. Their PUF implementation was also proven to be resilient and robust against temperature variations and aging effects, while the potential of the proposed TRNG design was verified using the NIST statistical test suite.

3.3. Applications of DRAM-Based Security Primitives

As the previous section shows, DRAM-based security primitives have been employed in a number of different applications, while both PUF and TRNG implementations have been constructed with equally good results. In this section, therefore, we will investigate the different primitives proposed by each work and their applications. In order, however, to do this, we first need to discuss what are the potential applications of PUFs and TRNGs and the way in which these applications work.

In particular, as already mentioned, the main difference between a PUF and a TRNG is the stability of their outputs. While a good PUF will provide pretty much the same output for the same input, a good TRNG will provide different outputs each time the same input is fed into it. This property of a good TRNG is also shown on Figure 5, where, for the same input A, the TRNG provides a different output, each time it is queried. Therefore, while TRNGs can be used for the construction of ephemeral keys and nonces, which are very useful in cryptography, they are not ideal for identification or authentication purposes. Nevertheless, ephemeral keys can provide information-theoretic security and perfect secrecy when used in a one-time pad scheme and nonces are widely used in order to protect against replay attacks.

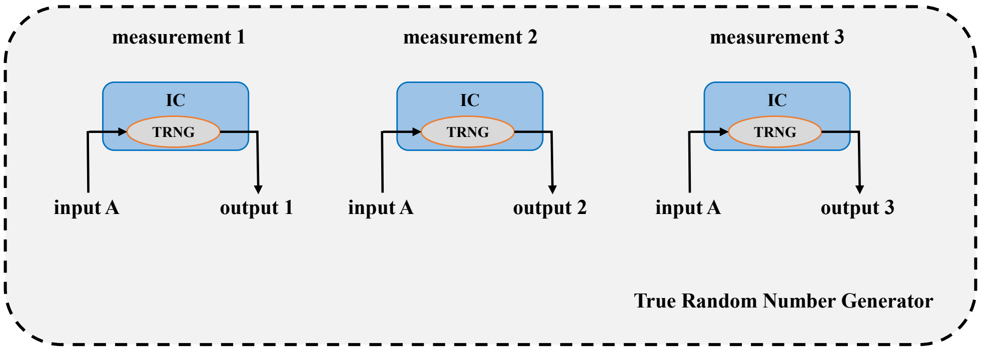


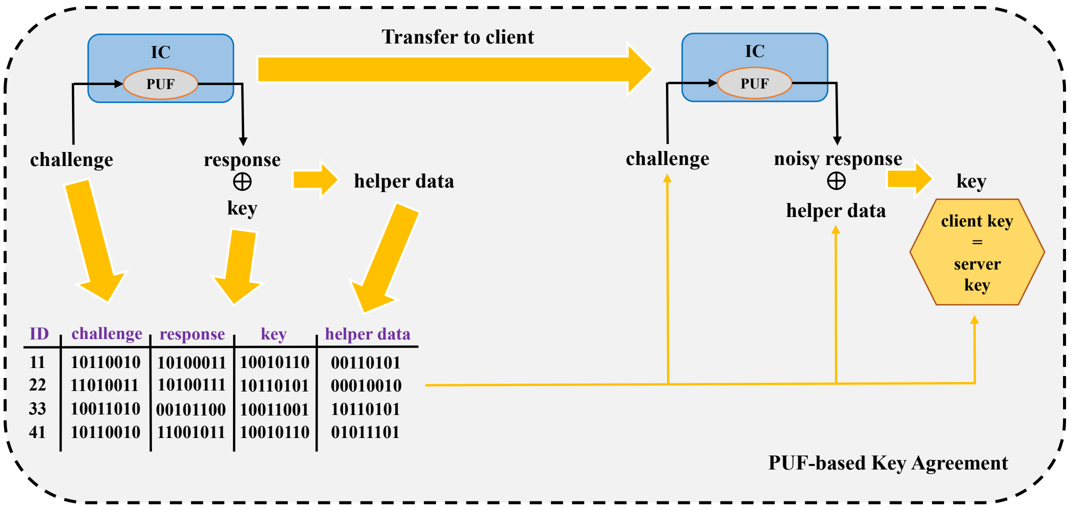
Figure 5. Principle of operation of a True Random Number Generator.

Additionally, a key produced by a TRNG can also be stored and used for identification and authentication. However, a key generated from a TRNG cannot be directly traced back to it and, therefore, has a weak connection to the device itself. On the contrary, 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 applications, namely key agreement, identification and authentication.

Regarding secure key agreement, usually a fuzzy extractor scheme needs to be applied, as a PUF often provides slightly noisy responses, which could otherwise affect its reliability [94,95]. This means that, for a particular challenge, the PUF may not always generate exactly the same response, but may often generate similar responses, which, however, contain some amount of noise. For this reason, we employ a fuzzy extractor scheme, which incorporates some Error Correction Code (ECC) [14], in order to stabilise the PUF response [96]. PUF-based key agreement consists of two phases, which are shown in Figure 6.

In the first phase, the enrollment phase, which is executed entirely on the server's side, the PUF response 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 ID regarding the CRP and the key used. As Figure 6 shows, the same CRP can be used with a different key, which will result in different helper data. If the fuzzy extraction 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.



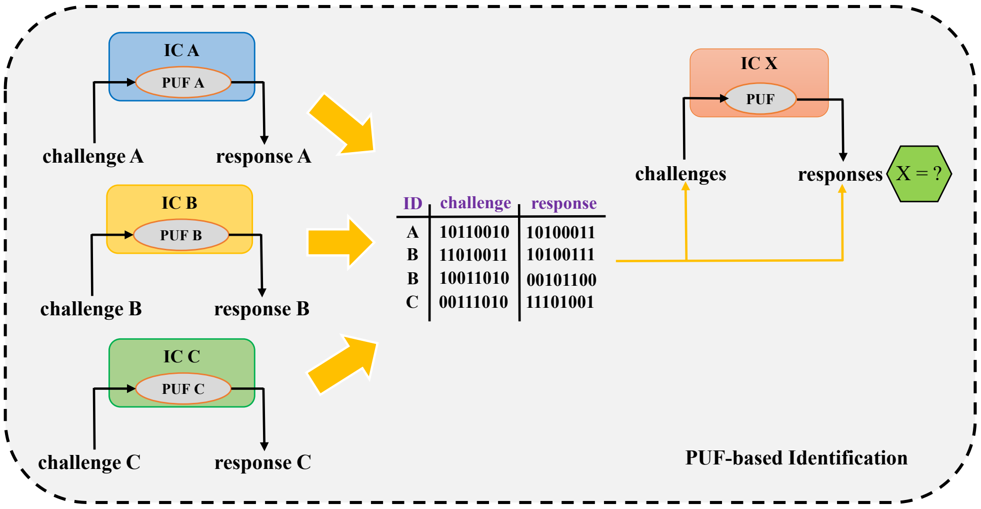
ID	challenge	response	key	helper data
11	10110010	10100011	10010110	00110101
22	11010011	10100111	10110101	00010010
33	10011010	00101100	10011001	10110101
41	10110010	11001011	10010110	01011101

Figure 6. PUF-based key agreement.

Regarding the fuzzy extraction scheme and its related ECC, a number of schemes have been proposed for DRAM-based PUFs, including fuzzy extractor schemes using helper data [4,19,75,82–84,86], fuzzy match schemes with or without hashing [81,89,90] and fuzzy vault schemes that also use helper data [13]. Additionally, using a universal hash implementation based on the Toeplitz matrix [83,84], which can be represented by a Linear-Feedback Shift Register (LFSR), is another way of stabilising the DRAM PUF responses. All of these schemes need to be combined with an ECC, which can potentially be based on either Hamming [16,19,86], BCH [82–84] or Reed–Solomon [13] error correction code.

However, instead of error correction, one can also employ the use of only the most stable cells [3,11,12,54,88], in order to create stable responses. 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.

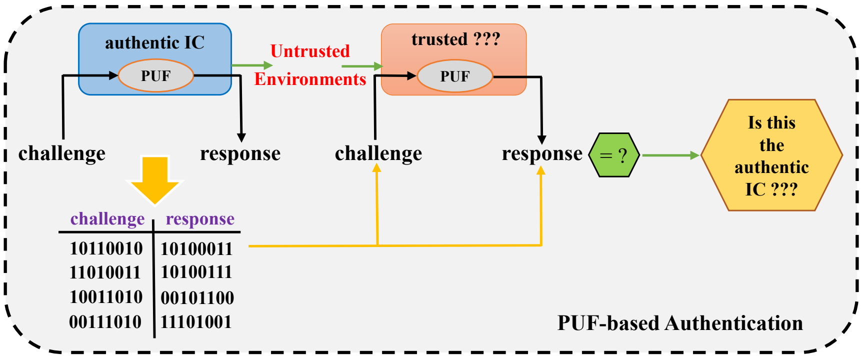
PUF-based identification works in a similar way, in the sense that a server queries multiple PUFs in order to create a database of (some of) their CRPs, which are stored according to the PUF from which they came from. Then, at any point in time, this server can identify any of these PUFs, without explicitly revealing the stored CRPs, by sending challenges to it 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 7 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.



ID	challenge	response
A	10110010	10100011
B	11010011	10100111
B	10011010	00101100
C	00111010	11101001

Figure 7. PUF-based identification.

Finally, PUF-based authentication is very similar to PUF-based identification, as once again a server creates a database of (some of) the CRPs of a PUF. However, this time, the PUF, and its related device, 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 8. 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.



challenge	response
10110010	10100011
11010011	10100111
10011010	00101100
00111010	11101001

Figure 8. PUF-based authentication.

In order to provide a clear and thorough overview of the applications of DRAM-based security primitives, we classify the relevant literature, first, in Table 3, based on the security primitive it concerns and, then, in Table 4, based on the applications these primitives have been suggested for. We should mention here that, in Table 3, we also include the category "Attack PUF", in order to describe a case where the PUF characteristic is used in order to actually facilitate an attack, rather than provide security. Additionally, in Table 4, we group together the "Authentication" and "Anti-Counterfeiting" categories, as they are highly related, one serving to validate the authenticity of the primitive and the other to invalidate counterfeit instances. The same principle is applied to "Identification" and "De-Anonymisation", where one category implies the identification of a primitive for security purposes and the other also its identification, but in order to facilitate an attack. Finally, of course, some additional factors that need to be taken into account, regarding the application for which a DRAM-based security primitive is more suitable for, are the ability of such a primitive to be used at run-time or not and the amount of CRPs it can generate. We discuss these factors in more detail in the other sections of this paper.

Table 3. Classification of publications according to the security primitive they concern. The publications are colour-coded according to the DRAM characteristic being exploited for the implementation of the security primitives examined in them.

Publication	PUF	TRNG	"Attack" PUF
Okamura et al., 2012 [13]	✓		
Fainstein et al., 2012 [18]	✓		
Keller et al., 2012 [14]	✓		
Felber, 2012 [15]	✓	✓	
Rosenblatt et al., 2013 [80]	✓		
Rosenblatt et al., 2013 [81]	✓		
Liu et al., 2014 [82]	✓		
Liu et al., 2014 [83]	✓		
Keller et al., 2014 [16]	✓	✓	
Tehranipoor et al., 2015 [12]	✓		
Hashemian et al., 2015 [20]	✓		
Zhang, 2015 [84]	✓		
Rahmati et al., 2015 [85]			✓
Tehranipoor et al., 2016 [17]		✓	
Sutar et al., 2016 [86]	✓		
Vitenko, 2016 [78]	✓		
Xiong et al., 2016 [4]	✓		
Sutar et al., 2017 [87]	✓		
Tehranipoor et al., 2017 [88]	✓		
Tang et al., 2017 [11]	✓		
Eckert et al., 2017 [55]		✓	
Tehranipoor et al., 2017 [3]	✓		
Schaller, 2017 [75]	✓ (both)		
Tehranipoor, 2017 [54]	✓ (only)	✓ (both)	
Kumar et al., 2017 [89]	✓		
Kirihata et al., 2017 [90]	✓		
Schaller et al., 2017 [5]	✓ (both)		
Kim et al., 2018 [21]	✓ (both)		
Sutar et al., 2018 [19]	✓	✓	

DRAM natural decay effect DRAM intensified decay effect due to VWL DRAM startup values
DRAM data remanence effect DRAM row hammer effect DRAM access latency

Table 4. Classification of publications according to their applications. The publications presented are colour-coded according to the DRAM characteristic being exploited for the implementation of the security primitives examined in them.

Publication	**Authentication and Anti-Counterfeiting**	**Identification and De-Anonymisation**	**Random Number Generation**	**Key Agreement**
Okamura et al., 2012 [13]	✓			✓
Fainstein et al., 2012 [18]	✓	✓		
Keller et al., 2012 [14]	✓			✓
Felber, 2012 [15]	✓	✓	✓	✓
Rosenblatt et al., 2013 [80]	✓	✓		✓
Rosenblatt et al., 2013 [81]	✓	✓		✓
Liu et al., 2014 [82]				✓
Liu et al., 2014 [83]				✓
Keller et al., 2014 [16]		✓	✓	
Tehranipoor et al., 2015 [12]	✓	✓		✓
Hashemian et al., 2015 [20]	✓	✓		
Zhang, 2015 [84]	✓			✓
Rahmati et al., 2015 [85]		✓		
Tehranipoor et al., 2016 [17]			✓	
Sutar et al., 2016 [86]	✓	✓		✓
Vitenko, 2016 [78]		✓		
Xiong et al., 2016 [4]	✓	✓		✓
Sutar et al., 2017 [87]	✓			
Tehranipoor et al., 2017 [88]	✓			✓
Tang et al., 2017 [11]	✓			✓
Eckert et al., 2017 [55]			✓	
Tehranipoor et al., 2017 [3]	✓	✓		✓
Schaller, 2017 [75]	✓ (both)	✓ (both)	✓ (both)	✓ (both)
Tehranipoor, 2017 [54]	✓ (both)	✓ (only)	✓ (both)	✓ (both)
Kumar et al., 2017 [89]	✓	✓		✓
Kirihata et al., 2017 [90]	✓	✓		✓
Schaller et al., 2017 [5]	✓ (both)	✓ (both)		✓ (both)
Kim et al., 2018 [21]	✓ (both)			
Sutar et al., 2018 [19]	✓	✓	✓	✓

DRAM natural decay effect DRAM intensified decay effect due to VWL DRAM startup values
DRAM data remanence effect DRAM row hammer effect DRAM access latency

3.4. Security Evaluation of DRAM-Based Security Primitives

Another important aspect that we need to examine is the actual level of security provided by DRAM-based primitives, as well as the different ways in which this can be assessed. In order to do this, we need to first examine the factors that may affect the security of DRAM-based primitives. Apart from conventional attack vectors, there are also other factors that affect the security of such primitives. These factors include the time needed for the generation of primitives, the number of unique outputs the primitive can generate, whether the DRAM module being used is removable or not, whether the primitive can generate outputs at run-time, whether it is affected by changes in the environmental conditions, or in the voltage being supplied and, finally, whether it is affected by aging.

In particular, if the generation time is high, then, obviously, there is a higher probability for an attack to be successful, without being detected. Additionally, if the DRAM module is removable, then an attacker can easily replace it, or steal it and connect it to a different device. Furthermore, a security primitive that can generate a lot of unique outputs provides a higher level of security than one that can only generate a single output, and is much more resistant, for example, to a de-anonymisation attack. Moreover, a security primitive that is accessible at run-time has a much higher potential to be used in practical security applications than one that is only accessible at boot-time. On the other hand, a primitive that is only accessible at boot-time can probably also only be attacked at boot-time and, therefore, has a smaller attack surface. Finally, obviously, a primitive that is significantly affected by factors that can easily be controlled by third-party entities, such as temperature, voltage, or whose outputs are significantly altered by aging, can offer a lower level of security than another primitive that remains unaffected.

3.4.1. Attacks and Defences

For this reason, we need to note here that security is a *relative* term, which is highly dependent on the manufacturing cost, the cost of performing an attack and the potential gains/damages of such an attack [97]. For example, and as invasive attacks have been proposed against DRAM-based security primitives [19,82–84,86,87], one could point out that a complex mesh shield of detectors around the DRAM could potentially prevent such attacks almost completely. However, if the manufacturing cost of such a mesh shield is much higher than the inclusion of another primitive that is invulnerable to such attacks, then DRAM-based security primitives may fail as security primitives. Additionally, however, if the costs of a successful invasive attack outweigh the potential gains for the attacker, then such attacks will inevitably not be used in such a scale as to render the security primitive insecure. Finally, we also need to distinguish between different implementations, as, for example, attacks based on the replacement of the DRAM module [3,14,54,87], have a different cost when removable DRAMs are employed than when on-board DRAM modules are used.

Keeping, therefore, in mind that there is no perfect security, and that security is highly relevant [97], we observe that a number of potential attacks have been proposed against DRAM-based security primitives, and we note possible defences and countermeasures. Invasive attacks [19,82–84,86,87] can be prevented if the DRAM module is tamper-proof or tamper-evident, meaning that it stops working after being physically tampered with or its behaviour is noticeably altered. Non-invasive probing, such as memory readouts [3,4,54,75,87], can be prevented if the DRAM can only be accessed by privileged code or is a dedicated security module to which access is limited. Attacks taking advantage of the collection of a large number of CRPs, such as machine learning attacks [19,86], as well as row hammer attacks [4], can also be prevented in the same way as memory readouts. Fault injections [3] can be detected through software or hardware means and appropriate action be taken.

Brute force attacks [3,81], as well as dictionary attacks, can be prevented if the entropy of the output of the DRAM security primitive is adequate. The entropy of the output is dependent on the probability that each bit of it, which corresponds to a memory cell, has a logical value of zero or one, on whether there are correlations among the values of the different memory cells and on the size of the output. Guessing attacks [4,19,86] are also highly unlikely to be successful if the DRAM-based security primitive produces outputs with high enough entropy, even when such attacks are based on expert manufacturing knowledge [19,54,55,86], which can also be prevented by keeping such knowledge only available to trustworthy parties.

In order for the entropy of the output of the DRAM-based security primitive to be high enough to prevent brute force, dictionary and guessing attacks, its possible values must be so many that an attacker is highly unlikely to find the correct one, either by luck (in the case of guessing attacks) or by an exhaustive (in the case of brute force attacks) or selective search (in the case of dictionary attacks). Therefore, if the bits of the output are highly unpredictable, having an almost equal probability to have a logical value of zero or one, are not correlated and the output has a large size, the attacker will be highly unlikely to be successful as the possible values of the output will be 2^N, where N is the size of the output.

Even if the output of the DRAM-based security primitive is biased towards one of the two logical values, if the size of the output is large enough and the bias is not extremely strong, the entropy of the output may still be adequate enough to prevent brute force, dictionary and guessing attacks, as the number of the possible values of the output is equal to the number of possible combinations $\binom{N}{U} = \frac{N!}{U!(N-U)!}$, where N is the size of the output and U the number of bits having the opposite value from the one towards which the output is biased. However, again, this only holds if the bits of the output are not correlated.

Finally, the attacker must be prevented from acquiring knowledge regarding the DRAM-based security primitive that can be used to render the relevant conditional entropy of its output low. For example, if an attacker can use expert manufacturing knowledge to correctly predict a large part of the output, the entropy of the output conditioned on this knowledge may become low enough to allow an otherwise infeasible attack to be successful.

The replacement or stealing of the DRAM module [3,14,54,87] can be prevented by using only on-board DRAM modules and detectors of their constant presence on the board. Environmental attacks, such as temperature-based attacks [3–5,12,13,16,17,19–21,54,75,80,85–87,89,90], as well as voltage-based attacks [3–5,11,12,20,54,80,81,89,90], can be prevented by using temperature and voltage sensors and taking appropriate actions.

Protocol attacks [4,19,54,75,81,86], such as replay, Man-in-the-Middle (MitM) and Denial of Service (DoS) attacks, can be prevented through the use of nonces, authenticated channels and a high degree of redundancy, respectively. Out-Of-Band (OOB) authentication can also help prevent protocol attacks. Collision and spoofing attacks [81,89,90] can also be prevent in the same way as protocol attacks in conjunction to using on-board DRAMs that have outputs with high entropy. Attacks based on aging [3–5,11,12,19–21,54,75,80,86–88,90] can be mitigated by using DRAM modules that are resilient to aging and anti-aging techniques.

Additionally, when inherent error correction hinders the PUF or the TRNG behaviour [5,21], it can be simply disabled or DRAMs without this feature can be used. Additionally, the TRNG behaviour of a DRAM-based implementation can be enhanced by combining multiple outputs and using de-biasing schemes, such as a von Neumann corrector [54,55].

Finally, we also need to examine the "attack" PUF scheme proposed by Rahmati et al. [85]. Attacks against this implementation constitute countermeasures against the de-anonymisation attack it is being used for. Its attack model is based on supply-chain and eavesdropping attacks [85], and countermeasures against it, which constitute attacks against the PUF's de-anonymisation application, are data segregation, noise and data scrambling. Noise can be countered through error correction schemes, while data segregation and scrambling can be potentially disabled.

3.4.2. Evaluation Metrics

We also need to examine the different metrics used in the relevant literature in order to assess the security of DRAM-based primitives. A significant reason for this is that the variety of metrics being employed in order to evaluate the security of different DRAM-based security primitives significantly hinders our ability to compare and assess them. This variety stems from the novelty of these implementations and their unique characteristics. In particular, not only conventional metrics, such as the Hamming weight [3–5,12,13,16–19,54,75,80–84,86,87,89,90], the intra-device and inter-device Hamming distances [3,4,11,12,16,19,20,54,82–84,86–88], the Shannon entropy [4,5,13,16,75,85] and the min-entropy [3,13,20,54,83,84], have been extensively employed, but also a number of less well-known metrics, such as the intra-device and inter-device Jaccard indices [4,5,21,75,85], the Sokal–Michener (simple matching) coefficient [78], the overlap distance [13,18,80,81,89,90] and other quantitative and statistical performance metrics [78].

Additionally, these metrics may have a different meaning, depending on the DRAM-based security primitive they are used for. For example, the Hamming weight of a DRAM decay PUF refers to the number of bit flips that have occured, while the Hamming weight of a DRAM PUF based on the startup values of the DRAM refers to the number of cells whose startup value is equal to logical one. Moreover, a number of other metrics, such as the number of stable cells [3,11,12,19,54,81,88–90], the probability of uniqueness [18,80,81,89,90], the chance of mismatch [85], the bit error rate [21,75] and the number of CRPs or keys being produced [3–5,11,14,15,19,21,54,75,80,81,85–87,89,90], have also been employed in order to assess the security that a DRAM-based primitive can provide.

Finally, we also need to mention the NIST statistical suite of tests, which serves as a sui generis suite of metrics for DRAM-based TRNG implementations [12,16,17,19,54,55], as well as the generation time as a metric to assess security [4,19,21,86]. It is therefore clear that it is currently not easy to compare different DRAM-based security primitives and their implementations, as a common single set of metrics is not currently being used in the relevant literature.

4. Discussion

In this section, we briefly compare DRAM-based security primitives to other hardware-based security primitives, discuss the potential of DRAM-based security primitives to be commercially adopted and, finally, present and discuss the different classification criteria employed in this publication.

4.1. A Short Comparison of DRAM-Based Security Primitives to Other Hardware-Based Security Primitives

As already mentioned, DRAMs have a number of advantages over other similar hardware-based security primitives. In particular, while a large variety of hardware-based security implementations have been proposed, memory-based primitives are attractive security mechanisms due to the ubiquitous presence of memory in virtually every computer system. Moreover, these memory-based security primitives do not require any additional circuitry for their operation, giving them a distinct advantage over other implementations. Additionally, unlike SRAM, standalone DRAM has a large size in most contemporary computer systems and most DRAM-based security primitives can also be accessed as a security primitive at run-time and not only at boot-time. Finally, most DRAM-based PUFs can provide multiple CRPs, in contrast to SRAM-based PUFs, which usually only allow for a single CRP.

However, DRAM-based security primitives may sometimes require a long generation time, especially when they are based on the retention times of DRAM cells. Nevertheless, recent research has significantly improved the output generation times, even for DRAM decay-based PUFs, as the relevant literature shows. Furthermore, if the DRAM module being used is removable, it provides more flexibility, but, at the same, it introduces the risk of being stolen or replaced. This issue can, however, be resolved if an on-board DRAM module is used. Finally, the characteristics of DRAM modules being used for the implementation of security primitives are quite often particularly susceptible to temperature changes. This is an issue that can, however, be resolved by the introduction of a temperature sensor.

We can therefore conclude that the intrinsic nature of DRAM-based primitives, their large size and their inclusion in most contemporary computer systems make them appealing for security applications, as they are highly cost-efficient, lightweight and can act as sources of significant entropy. However, a number of open issues still remains to be addressed by future research.

4.2. The Potential of DRAM-Based Security Primitives for Commercial Adoption

As we have already discussed in the previous sections, DRAMs are already widely available and in mass production, forming an intrinsic part of most contemporary computer systems. Additionally, they can also provide run-time access to characteristics that can be used for the implementation of security primitives, and have, most often, a significantly large size, which can ensure that DRAM-based security primitives can provide enough entropy. Therefore, and as DRAM-based security primitives have a number of advantages over other hardware-based security primitives, we need to assess their potential to be adopted as widely used commercially available security implementations.

However, in order to do this, we first need to examine the implementations proposed in the relevant literature and their characteristics. For this purpose, Table 5 presents the various implementations described in the different works concerning DRAM-based security primitives. Based on this table, it is obvious that DRAM-based security primitives have been tested using both simulations and practical implementations. Regarding practical implementations of such security primitives, these range from external DIP, removable DIMM and on-board DRAM modules connected to FPGA and evaluation COTS boards to widely used commercial devices. It should, therefore, be evident that DRAM-based security primitives can easily become commercially available and be used in a large scale.

Table 5. Classification of publications according to their implementation setup. The publications are colour-coded according to the DRAM characteristic being exploited for the implementation of the security primitives examined in them.

Implementation Setup	**Publications**
Monte Carlo simulation	Fainstein et al., 2012 [18], Rosenblatt et al., 2013 [80], Rosenblatt et al., 2013 [81], Kumar et al., 2017 [89], Kirihata et al., 2017 [90]
SPICE-based simulation	Hashemian et al., 2015 [20]
novel eDRAM ASIC	Fainstein et al., 2012 [18], Rosenblatt et al., 2013 [80], Rosenblatt et al., 2013 [81], Tang et al., 2017 [11], Kumar et al., 2017 [89], Kirihata et al., 2017 [90]
FPGA and external DIP DRAM	Tehranipoor et al., 2015 [12], Tehranipoor et al., 2017 [88], Tehranipoor et al., 2017 [3], Tehranipoor, 2017 [54] (only)
FPGA and removable DIMM DRAM	Liu et al., 2014 [82], Liu et al., 2014 [83], Keller et al., 2014 [16], Zhang, 2015 [84], Rahmati et al., 2015 [85], Sutar et al., 2016 [86], Sutar et al., 2017 [87], Sutar et al., 2018 [19]
FPGA and on-board DRAM	Okamura et al., 2012 [13], Tehranipoor et al., 2016 [17], Eckert et al., 2017 [55], Tehranipoor, 2017 [54]
evaluation board and external DIP DRAM	Rahmati et al., 2015 [85]
evaluation board and removable DIMM DRAM	Kim et al., 2018 [21]
evaluation board and on-board DRAM	Xiong et al., 2016 [4], Schaller, 2017 [75],Schaller et al., 2017 [5]
conventional commercial hardware	Vitenko, 2016 [78] (on-board DRAM)
DRAM natural decay effect DRAM intensified decay effect due to VWL DRAM startup values DRAM data remanence effect DRAM row hammer effect DRAM access latency	

As already noted in the previous sections, existing barriers against the commercial adoption of DRAM-based security primitives can be overcome in various different ways, towards which existing research is already moving. This is evident in recent publications, some of which consider the generation times of these primitives and how they can be improved [4,19,21,86] or the implementation of such security primitives using the latest commercially available DRAM modules, such as Low-Power Double-Data-Rate type 4 (LPDDR4) DRAM modules [21], or widely used commercial devices [78]. To this end, research has considered DRAM-based security primitives based on the startup values of cells, which can most often only provide a single input–output pair, but has also focused on DRAM-based security primitives that provide multiple input–output pairs, such as DRAM decay-based and DRAM latency-based primitives, as it can clearly been seen on Table 2.

Finally, as DRAM-based security primitives are already present in most contemporary computer systems, including IoT hardware, which is usually resource-constrained, they provide the ideal basis for the implementation of intrinsic highly cost-efficient security mechanisms in IoT devices. As IoT devices quite often cannot inherently support additional security mechanisms, such as Trusted Platform Modules (TPMs) and other security primitives that require hardware additions, DRAM modules can allow for the implementation of security primitives in such devices, which can then be utilised in a number of different security applications, such as cryptographic protocols. This is further supported by the ability of DRAM modules to serve as a basis for the creation of both TRNGs and PUFs, which are also accessible at run-time. We can, therefore, conclude that DRAM-based security primitives exhibit significant potential for commercial adoption, especially in IoT devices.

This is of particular importance, as a constant rise has been noted in the use of IoT devices for a multitude of different applications, which include user and customer authentication, weather prediction, traffic monitoring and health applications that range from monitoring general health indicators to tracking specific heart signals [98]. This constant expansion of IoT devices into everyday life significantly increases the need for adequate security solutions that are compatible with the constrained resources of IoT hardware, such as DRAM-based security primitives [7]. Therefore, there are increased incentives for the commercial adoption of DRAM-based security primitives.

4.3. Classification Criteria

In this section, we provide a quick overview of the classification criteria used in this paper to classify DRAM-based security primitives and the relevant literature and discuss their significance.

In particular, we note that we have provided a comprehensive overview of works concerning such primitives, and have used the following criteria in order to produce a thorough classification of them:

- Year of publication, in order to demonstrate the significance of the topic of DRAM-based security primitives and provide insights into the future development of the relevant scientific field.
- DRAM characteristic being exploited for the implementation of a security primitive, in order to demonstrate the diversity of characteristics being used and highlight the number of works about them.
- Security primitive being implemented, in order to demonstrate that works concerning both PUF and TRNG implementations exist and that both primitive types can be generated, with equally good results, using DRAMs.
- Applications used, in order to demonstrate that all DRAM-based primitives can be employed in a wide range of security applications, serving as the basis for the implementation of relevant cryptographic protocols.
- Security considerations, regarding both attacks against the implemented security primitives and countermeasures against such attacks, as well as an overview of the employed security metrics, in order to provide detailed insights into the security of DRAM-based primitives.
- Implementation setup, in order to discuss whether they can be commercially adopted and how practical are the implementations discussed in the relevant literature.

Finally, we also note that we have additionally provided a comparison of DRAM-based security primitives to other hardware-based security primitives, in order to provide a clear and thorough view of their advantages and disadvantages, in comparison to other hardware-based security primitives.

5. Conclusions

In this work, we have presented a comprehensive overview and an extended classification of DRAM-based security primitives. We have examined the characteristics that are being exploited for the implementation of such primitives, and also provided extended insights into the way a DRAM works. We have also discussed in detail PUFs and TRNGs as the security primitives being implemented. Additionally, we have provided insights into the development of the relevant scientific field, the applications of DRAM-based security primitives and their evaluation as security mechanisms.

Moreover, we have also compared them to other hardware-based security primitives, noting their advantages and disadvantages, discussed their potential for commercial adoption and, finally, presented our classification methodology and the significance of the criteria employed in our classification.

Based on our extensive survey of the relevant literature, we can conclude that DRAM-based security primitives can be easily adopted for commercial use and can provide significant advantages when employed as a basis for security in resource-constrained devices, such as IoT hardware. Although current implementations exhibit a number of potential shortcomings, current research is already trying to address them. We can, therefore, conclude that DRAM-based security primitives could easily be employed in commercial applications in the future, and that the relevant scientific field may potentially thrive in the future.

To this end, some related issues that still remain to be addressed in future works include the establishment of commonly accepted security metrics, in order to enable an easier evaluation of the implemented DRAM-based security primitives, a thorough assessment of ways to mitigate their dependence on temperature and voltage variations and the effects of aging, as well as a review and analysis of ways to secure removable DRAM modules, which could allow for more flexibility. Additionally, another topic that is still under investigation concerns further ways in which the time required for the generation of the outputs of DRAM-based security primitives could be reduced even further. Finally, a future study could also examine whether the Integrated Circuit (IC) design parameters relevant to the design of DRAM modules can affect these outputs. The quick growth and the recent advances in this scientific field, as they have been demonstrated in this paper, lead us to believe that all the open research topics discussed in this section may be thoroughly investigated in the near future.

Acknowledgments: This work has been partly funded by the DFG (Deutsche Forschungsgemeinschaft; German Research Foundation) as part of project P3 within the CRC (Collaborative Research Center) 1119 CROSSING. This work has also been supported by Comcast Corporation, a company headquartered in Philadelphia, PA, USA, and Honeywell International Inc., a company headquarted in Morris Plains, NJ, USA.

Author Contributions: Nikolaos Athanasios Anagnostopoulos has contributed in this paper by gathering and classifying the literature used in it, by assessing the results of this classification, by writing and proofreading most of this paper and by creating some of the figures used in it. Stefan Katzenbeisser has guided the writing of this paper, by providing useful background information, especially regarding the way in which entropy can prevent some of the attacks discussed, and by giving valuable feedback that significantly improved the quality of the paper. John Chandy has also encouraged and guided the writing of this paper, by helping select and assess the relevant literature and by providing crucial feedback that truly helped to improve the quality of this paper. Fatemeh Tehranipoor has conceived the idea for this paper and was essential to its materialisation into a written manuscript. She has contributed by gathering, analysing and classifying the literature used in it and helping to assess it and by writing some sections of this paper and creating some of the figures used in it.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

1T1C	one-Transistor one-Capacitor
2T	two-Transistor
3T	three-Transistor
ASIC	Application-Specific Integrated Circuit
CMOS	Complementary Metal–Oxide–Semiconductor
COTS	Commercial Off-The-Shelf
DDR	Double-Data-Rate (type one)
DDR2	Double-Data-Rate type two
DDR3	Double-Data-Rate type three
DDR4	Double-Data-Rate type four
DIMM	Dual In-line Memory Module
DIP	Dual In-line Package
DoS	Denial of Service
DRAM	Dynamic Random Access Memory
ECC	Error Correction Code

eDRAM	embedded Dynamic Random Access Memory
eFUSE	electrically programmable fuse
FPGA	Field-Programmable Gate Array
GB	GigaByte
IC	Integrated Circuit
IT	Information Technology
IoT	Internet of Things
KB	KiloByte
Kbit	Kilobit
LPDDR4	Low Power Double-Data-Rate type 4
LFSR	Linear-Feedback Shift Register
MB	MegaByte
MDPI	Multidisciplinary Digital Publishing Institute
MitM	Man-in-the-Middle
NBTI	Negative-Bias Temperature Instability
NIST	National Institute of Standards and Technology (USA)
OOB	Out-Of-Band
OS	Operating System
OTP	One-Time Pad
PRNG	Pseudo-Random Number Generator
PUF	Physical Unclonable Function
QKD	Quantum Key Distribution
RAM	Random Access Memory
RFID	Radio-Frequency IDentification
RICID	Retention-based Intrinsic Chip ID
SDRAM	Synchronous Dynamic Random Access Memory
SODIMM	Small Outline Dual In-line Memory Module
SOI	Silicon-On-Insulator
SPICE	Simulation Program with Integrated Circuit Emphasis
SRAM	Static Random Access Memory
TPM	Trusted Platform Module
TRNG	True Random Number Generator
USA	United States of America
VRT	Variable Retention Time
VWL	Wordline Low Voltage
WSN	Wireless Sensor Network
XOR	eXclusive OR

References

1. Landau, S. Making Sense from Snowden: What's Significant in the NSA Surveillance Revelations. *IEEE Secur. Priv.* **2013**, *11*, 54–63.
2. Katzenbeisser, S.; Kocabaş, Ü.; Rožić, V.; Sadeghi, A.R.; Verbauwhede, I.; Wachsmann, C. PUFs: Myth, Fact or Busted? A Security Evaluation of Physically Unclonable Functions (PUFs) Cast in Silicon. In *International Workshop on Cryptographic Hardware and Embedded Systems*; Springer: Berlin, Germany, 2012; pp. 283–301.
3. Tehranipoor, F.; Karimian, N.; Yan, W.; Chandy, J.A. DRAM-Based Intrinsic Physically Unclonable Functions for System-Level Security and Authentication. *IEEE Trans. Very Large Scale Integr. Syst.* **2017**, *25*, 1085–1097.
4. Xiong, W.; Schaller, A.; Anagnostopoulos, N.A.; Saleem, M.U.; Gabmeyer, S.; Katzenbeisser, S.; Szefer, J. Run-Time Accessible DRAM PUFs in Commodity Devices. In *International Conference on Cryptographic Hardware and Embedded Systems*; Springer: Berlin, Germany, 2016; pp. 432–453.
5. Schaller, A.; Xiong, W.; Anagnostopoulos, N.A.; Saleem, M.U.; Gabmeyer, S.; Katzenbeisser, S.; Szefer, J. Intrinsic Rowhammer PUFs: Leveraging the Rowhammer Effect for Improved Security. In Proceedings of the 2017 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), McLean, VA, USA, 1–5 May 2017; pp. 1–7.
6. Prabhu, P.; Akel, A.; Grupp, L.M.; Wing-Kei, S.Y.; Suh, G.E.; Kan, E.; Swanson, S. Extracting Device Fingerprints from Flash Memory by Exploiting Physical Variations. In *International Conference on Trust and Trustworthy Computing*; Springer: Berlin, Germany, 2011; pp. 188–201.
7. Tehranipoor, F.; Karimian, N.; Wortman, P. A.; Haque, A.; Fahrny, J.; Chandy, J. A.; Exploring Methods of Authentication for the Internet of Things. In *Internet of Things: Challenges, Advances, and Applications*; Hassan, Q.F., Khan, A.R., Madani, S.A., Eds.; CRC Press: Boca Raton, FL, USA, 2017.

8. Liu, J.; Jaiyen, B.; Kim, Y.; Wilkerson, C.; Mutlu, O. An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms. In *ACM SIGARCH Computer Architecture News*; ACM: New York, NY, USA, 2013; Volume 41, pp. 60–71.
9. Keeth, B.; Baker, R.J.; Johnson, B.; Lin, F. *DRAM Circuit Design: Fundamental and High-Speed Topics*, 2nd ed.; Wiley-IEEE Press: Piscataway, NJ, USA, 2007.
10. Chun, K.C.; Jain, P.; Kim, T.H.; Kim, C.H. A 667 MHz Logic-Compatible Embedded DRAM Featuring an Asymmetric 2T Gain Cell for High Speed On-Die Caches. *IEEE J. Solid State Circuits* **2012**, *47*, 547–559.
11. Tang, Q.; Zhou, C.; Choi, W.; Kang, G.; Park, J.; Parhi, K.K.; Kim, C.H. A DRAM Based Physical Unclonable Function Capable of Generating $> 10^{32}$ Challenge Response Pairs per 1Kbit Array for Secure Chip Authentication. In Proceedings of the 2017 IEEE Custom Integrated Circuits Conference (CICC), Austin, TX, USA, 30 April–3 May 2017.
12. Tehranipoor, F.; Karimian, N.; Xiao, K.; Chandy, J. DRAM Based Intrinsic Physical Unclonable Functions for System Level Security. In Proceedings of the Great Lakes Symposium on VLSI, Pittsburgh, PA, USA, 20–22 May 2015; pp. 15–20.
13. Okamura, T.; Minematsu, K.; Tsunoo, Y.; Iida, T.; Kimura, T.; Nakamura, K. DRAM PUF (in Japanese). In *Proceedings of the 29th Symposium on Cryptography and Information Security (SCIS 2012)*; Institute of Electronics, Information and Communication Engineers: Tokyo, Japan, 2012.
14. Keller, C.; Felber, N.; Gürkaynak, F.; Kaeslin, H.; Junod, P. *Physically Unclonable Functions for Secure Hardware (poster)*; RTD 2010—QCrypt; Swiss National Science Foundation (SNSF): Bern, Switzerland; Nano-Tera.CH: Lausanne, Switzerland, 2012.
15. Felber, N. *for Gisin, N. Secure High-Speed Communication Based on Quantum Key Distribution (presentation slides)*; RTD 2010—QCrypt, Annual Plenary Meeting; Swiss National Science Foundation (SNSF): Bern, Switzerland; Nano-Tera.CH: Lausanne, Switzerland, 2012.
16. Keller, C.; Gürkaynak, F.; Kaeslin, H.; Felber, N. Dynamic Memory-Based Physically Unclonable Function for the Generation of Unique Identifiers and True Random Numbers. In Proceedings of the 2014 IEEE International Symposium on Circuits and Systems (ISCAS), Melbourne, VIC, Australia, 1–5 June 2014; pp. 2740–2743.
17. Tehranipoor, F.; Yan, W.; Chandy, J.A. Robust Hardware True Random Number Generators using DRAM Remanence Effects. In Proceedings of the 2016 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), McLean, VA, USA, 3–5 May 2016; pp. 79–84.
18. Fainstein, D.; Rosenblatt, S.; Cestero, A.; Robson, N.; Kirihata, T.; Iyer, S.S. Dynamic Intrinsic Chip ID Using 32nm High-K/Metal Gate SOI Embedded DRAM. In Proceedings of the 2012 Symposium on VLSI Circuits (VLSIC), Honolulu, HI, USA, 13–15 June 2012; pp. 146–147.
19. Sutar, S.; Raha, A.; Kulkarni, D.; Shorey, R.; Tew, J.; Raghunathan, V. D-PUF: An Intrinsically Reconfigurable DRAM PUF for Device Authentication and Random Number Generation. *ACM Trans. Embed. Comput. Syst.* **2018**, *17*.
20. Hashemian, M.S.; Singh, B.; Wolff, F.; Weyer, D.; Clay, S.; Papachristou, C. A Robust Authentication Methodology Using Physically Unclonable Functions in DRAM Arrays. In Proceedings of the Design, Automation & Test in Europe Conference, Grenoble, France, 9–13 March 2015; pp. 647–652.
21. Kim, J.S.; Patel, M.; Hassan, H.; Mutlu, O. The DRAM Latency PUF: Quickly Evaluating Physical Unclonable Functions by Exploiting the Latency-Reliability Tradeoff in Modern DRAM Devices. In Proceedings of the 24th International Symposium on High-Performance Computer Architecture (HPCA), Vienna, Austria, 24–28 February 2018.
22. Bauder, D.W. *An Anti-Counterfeiting Concept for Currency Systems*; Technical Report PTK-11990; Sandia National Labs: Albuquerque, NM, USA, 1983.
23. Simmons, G.J. A System for Verifying User Identity and Authorization at the Point-of Sale or Access. *Cryptologia* **1984**, *8*, 1–21.
24. Simmons, G.J. Identification of Data, Devices, Documents and Individuals. In Proceedings of the 25th Annual IEEE International Carnahan Conference on Security Technology, Taipei, Taiwan, 1–3 October 1991; pp. 197–218.
25. Pappu, R.S. Physical One-Way Functions. Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2001.
26. Pappu, R.; Recht, B.; Taylor, J.; Gershenfeld, N. Physical One-Way Functions. *Science* **2002**, *297*, 2026–2030.

27. Gassend, B.; Clarke, D.; van Dijk, M.; Devadas, S. Silicon Physical Random Functions. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*; ACM: New York, NY, USA, 2002; pp. 148–160.
28. Gassend, B.L.P. Physical Random Functions. Master's Thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2003.
29. Guajardo, J.; Kumar, S.S.; Schrijen, G.J.; Tuyls, P. FPGA Intrinsic PUFs and their Use for IP Protection. In *International Workshop on Cryptographic Hardware and Embedded Systems*; Springer: Berlin, Germany, 2007; pp. 63–80.
30. Holcomb, D.E.; Burleson, W.P.; Fu, K. Initial SRAM State as a Fingerprint and Source of True Random Numbers for RFID Tags. In Proceedings of the Conference on RFID Security, Malaga, Spain, 11–13 July 2007.
31. Yan, W.; Jin, C.; Tehranipoor, F.; Chandy, J.A. Phase Calibrated Ring Oscillator PUF Design and Implementation on FPGAs. In Proceedings of the 27th International Conference on Field Programmable Logic and Applications (FPL 2017), Ghent, East Flanders, Belgium, 4–8 September 2017.
32. Herder, C.; Yu, M.D.; Koushanfar, F.; Devadas, S. Physical Unclonable Functions and Applications: A Tutorial. *Proc. IEEE* **2014**, *102*, 1126–1141.
33. Lee, J.W.; Lim, D.; Gassend, B.; Suh, G.E.; van Dijk, M.; Devadas, S. A Technique to Build a Secret Key in Integrated Circuits for Identification and Authentication Applications. In Proceedings of the 2004 Symposium on VLSI Circuits, Digest of Technical Papers, Honolulu, HI, USA, 17–19 June 2004; pp. 176–179.
34. Majzoobi, M.; Koushanfar, F.; Potkonjak, M. Testing Techniques for Hardware Security. In Proceedings of the 2008 IEEE International Test Conference, Santa Clara, CA, USA, 28–30 October 2008.
35. Rührmair, U.; Sehnke, F.; Sölter, J.; Dror, G.; Devadas, S.; Schmidhuber, J. Modeling Attacks on Physical Unclonable Functions. In Proceedings of the 17th ACM Conference on Computer and Communications Security, Chicago, IL, USA, 4–8 October 2010; pp. 237–249.
36. Rührmair, U.; Sölter, J.; Sehnke, F.; Xu, X.; Mahmoud, A.; Stoyanova, V.; Dror, G.; Schmidhuber, J.; Burleson, W.; Devadas, S. PUF Modeling Attacks on Simulated and Silicon Data. *IEEE Trans. Inf. Forensics Secur.* **2013**, *8*, 1876–1891.
37. Tuyls, P.; Schrijen, G.J.; Škorić, B.; Van Geloven, J.; Verhaegh, N.; Wolters, R. Read-Proof Hardware from Protective Coatings. In *International Workshop on Cryptographic Hardware and Embedded Systems*; Springer: Berlin, Germany, 2006; pp. 369–383.
38. Suh, G.E.; Devadas, S. Physical Unclonable Functions for Device Authentication and Secret Key Generation. In Proceedings of the 44th Annual Design Automation Conference, San Diego, CA, USA, 4–8 June 2007; pp. 9–14.
39. Su, Y.; Holleman, J.; Otis, B. A 1.6 pJ/bit 96% Stable Chip-ID Generating Circuit Using Process Variations. In Proceedings of the Digest of Technical Papers of the 2007 IEEE International Solid-State Circuits Conference (ISSCC 2007), San Francisco, CA, USA, 11–15 February 2007; pp. 406–611.
40. Majzoobi, M.; Koushanfar, F.; Potkonjak, M. Lightweight Secure PUFs. In Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design, San Jose, CA, USA, 10–13 November 2008; pp. 670–673.
41. Kumar, S.S.; Guajardo, J.; Maes, R.; Schrijen, G.J.; Tuyls, P. The Butterfly PUF Protecting IP on Every FPGA. In Proceedings of the 2008 IEEE International Workshop on Hardware-Oriented Security and Trust (HOST 2008), Anaheim, CA, USA, 9 June 2008; pp. 67–70.
42. Maes, R.; Tuyls, P.; Verbauwhede, I. Intrinsic PUFs from Flip-Flops on Reconfigurable Devices. In Proceedings of the 3rd Benelux Workshop on Information and System Security (WISSec 2008), Eindhoven, The Netherlands, 13–14 November 2008; Volume 17.
43. Suzuki, D.; Shimizu, K. The Glitch PUF: A New Delay-PUF Architecture Exploiting Glitch Shapes. In *International Workshop on Cryptographic Hardware and Embedded Systems*; Springer: Berlin, Germany, 2010; pp. 366–382.
44. Majzoobi, M.; Ghiaasi, G.; Koushanfar, F.; Nassif, S.R. Ultra-Low Power Current-Based PUF. In Proceedings of the 2011 IEEE International Symposium on Circuits and Systems (ISCAS), Rio de Janeiro, Brazil, 15–18 May 2011; pp. 2071–2074.
45. Chen, Q.; Csaba, G.; Lugli, P.; Schlichtmann, U.; Rührmair, U. The Bistable Ring Puf: A New Architecture for Strong Physical Unclonable Functions. In Proceedings of the 2011 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), San Diego, CA, USA, 5–6 June 2011; pp. 134–141.

46. Simons, P.; van der Sluis, E.; van der Leest, V. Buskeeper PUFs, a Promising Alternative to D Flip-Flop PUFs. In Proceedings of the 2012 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), San Francisco, CA, USA, 3–4 June 2012; pp. 7–12.
47. Holcomb, D.E.; Fu, K. Bitline PUF: Building Native Challenge-Response PUF Capability into Any SRAM. In *International Workshop on Cryptographic Hardware and Embedded Systems*; Springer: Berlin, Germany, 2014; pp. 510–526.
48. Bossuet, L.; Ngo, X.T.; Cherif, Z.; Fischer, V. A PUF Based on a Transient Effect Ring Oscillator and Insensitive to Locking Phenomenon. *IEEE Trans. Emerg. Top. Comput.* **2014**, *2*, 30–36.
49. Willers, O.; Huth, C.; Guajardo, J.; Seidel, H. MEMS Gyroscopes as Physical Unclonable Functions. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016; pp. 591–602.
50. Rukhin, A.; Soto, J.; Nechvatal, J.; Smid, M.; Barker, E.; Leigh, S.; Levenson, M.; Vangel, M.; Banks, D.; Heckert, A.; et al. *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*; Special Publication 800:22; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2010.
51. Chan, J.J.M.; Sharma, B.; Lv, J.; Thomas, G.; Thulasiram, R.; Thulasiraman, P. True Random Number Generator Using GPUs and Histogram Equalization Techniques. In Proceedings of the 13th IEEE International Conference on High Performance Computing and Communications (HPCC 2011), Banff, AB, Canada, 2–4 September 2011; pp. 161–170.
52. Tehranipoor, F.; Karimian, N.; Yan, W.; Chandy, J.A. A Study of Power Supply Variation as a Source of Random Noise. In Proceedings of the 30th International Conference on VLSI Design and 16th International Conference on Embedded Systems (VLSID 2017), Hyderabad, India, 7–11 January 2017; pp. 155–160.
53. Tehranipoor, F.; Wortman, P.; Karimian, N.; Yan, W.; Chandy, J. DVFT: A Lightweight Solution for Power Supply Noise Based TRNG Using a Dynamic Voltage Feedback Tuning System. *IEEE Trans. Very Large Scale Integr. Syst.* **2018**, doi:10.1109/TVLSI.2018.2804258.
54. Tehranipoor, F. Design and Architecture of Hardware-Based Random Function Security Primitives. Ph.D. Thesis, University of Connecticut (UConn), Storrs, CT, USA, 2017.
55. Eckert, C.; Tehranipoor, F.; Chandy, J.A. DRNG: DRAM-Based Random Number Generation Using its Startup Value Behavior. In Proceedings of the 60th IEEE International Midwest Symposium on Circuits and Systems (MWSCAS), Boston, MA, USA, 6–9 August 2017.
56. Komninos, N.; Philippou, E.; Pitsillides, A. Survey in Smart Grid and Smart Home Security: Issues, Challenges and Countermeasures. *IEEE Commun. Surv. Tutor.* **2014**, *16*, 1933–1954.
57. Mahmud, R.; Vallakati, R.; Mukherjee, A.; Ranganathan, P.; Nejadpak, A. A Survey on Smart Grid Metering Infrastructures: Threats and Solutions. In Proceedings of the 2015 IEEE International Conference on Electro/Information Technology (EIT), Dekalb, IL, USA, 21–23 May 2015; pp. 386–391.
58. Karygiannis, A.; Phillips, T.; Tsibertzopoulos, A. RFID Security: A Taxonomy of Risk. In Proceedings of the First International Conference on Communications and Networking in China (ChinaCom 2006), Beijing, China, 25–27 October 2006.
59. Mitrokotsa, A.; Rieback, M.; Tanenbaum, A. Classification of RFID Attacks. In Proceedings of the 2nd International Workshop on RFID Technology – Concepts, Applications, Challenges (IWRT 2008), Barcelona, Spain, June 2008; pp. 73–86.
60. Atzori, L.; Iera, A.; Morabito, G. The Internet of Things: A Survey. *Comput. Netw.* **2010**, *54*, 2787–2805.
61. Al-Fuqaha, A.; Guizani, M.; Mohammadi, M.; Aledhari, M.; Ayyash, M. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 2347–2376.
62. Da Xu, L.; He, W.; Li, S. Internet of Things in Industries: A Survey. *IEEE Trans. Ind. Inf.* **2014**, *10*, 2233–2243.
63. Perera, C.; Zaslavsky, A.; Christen, P.; Georgakopoulos, D. Context Aware Computing for the Internet of Things: A Survey. *IEEE Commun. Surv. Tutor.* **2014**, *16*, 414–454.
64. Chang, C.H.; Zheng, Y.; Zhang, L. A Retrospective and a Look Forward: Fifteen Years of Physical Unclonable Function Advancement. *IEEE Circuits Syst. Mag.* **2017**, *17*, 32–62.
65. Maes, R. *Physically Unclonable Functions: Constructions, Properties and Applications*, 1st ed.; Springer: Berlin, Germany, 2013.
66. Maes, R.; Verbauwhede, I. Physically Unclonable Functions: A Study on the State of the Art and Future Research Directions. In *Towards Hardware-Intrinsic Security*; Springer: Berlin, Germany, 2010; pp. 3–37.

67. Rostami, M.; Wendt, J.B.; Potkonjak, M.; Koushanfar, F. Quo Vadis, PUF?: Trends and Challenges of Emerging Physical-Disorder Based Security. In Proceedings of the Conference on Design, Automation & Test in Europe, Dresden, Germany, 24–28 March 2014.
68. Rührmair, U.; Holcomb, D.E. PUFs at a Glance. In Proceedings of the Conference on Design, Automation & Test in Europe, Dresden, Germany, 24–28 March 2014.
69. Böhm, C.; Hofer, M. *Physical Unclonable Functions in Theory and Practice*; Springer Science & Business Media: New York, NY, USA, 2012.
70. Busch, H.; Sotáková, M.; Katzenbeisser, S.; Sion, R. The PUF Promise. In *International Conference on Trust and Trustworthy Computing*; Springer: Berlin, Germany, 2010; pp. 290–297.
71. Eiroa, S.; Baturone, I.; Acosta, A.J.; Dávila, J. *Using Physical Unclonable Functions for Hardware Authentication: A Survey*. In Proceedings of the XXV Conference on Design of Circuits and Integrated Systems (DCIS), Lanzarote, Las Palmas, Canary Islands, Spain, 12–14 November 2010.
72. Tehranipoor, M.; Koushanfar, F. A Survey of Hardware Trojan Taxonomy and Detection. *IEEE Des. Test Comput.* **2010**, *27*, 10–25.
73. Karri, R.; Rajendran, J.; Rosenfeld, K. Trojan Taxonomy. In *Introduction to Hardware Security and Trust*; Springer: Berlin, Germany, 2012; pp. 325–338.
74. Karri, R.; Rajendran, J.; Rosenfeld, K.; Tehranipoor, M. Trustworthy Hardware: Identifying and Classifying Hardware Trojans. *Computer* **2010**, *43*, 39–46.
75. Schaller, A. Lightweight Protocols and Applications for Memory-Based Intrinsic Physically Unclonable Functions Found on Commercial Off-The-Shelf Devices. Ph.D. Thesis, Technische Universität Darmstadt, Darmstadt, Germany, 2017.
76. Ravishankar, Y. PUFs – An Extensive Survey. Ph.D. Thesis, George Mason University, Fairfax, VA, USA, 2017.
77. Sahoo, D.P. Design and Analysis of Secure Physically Unclonable Function Compositions. Ph.D. Thesis, Indian Institute of Technology Kharagpur (IIT KGP), Kharagpur, West Bengal, India, 2017.
78. Витенко, А. Использование DRAM-PUF для Идентификации Мобильных Устройств под Управлением ОС Android (in Russian). Апробация **2016**, *1*, 26–29.
79. Пучков, А.В.; Иванюк, А.А. Применение Запоминающих Устройств в качестве Криптографических Примитивов для Интегральных Схем Программируемой Логики (in Russian). In Proceeding of the 2016 International Conference on Information Technologies and Systems (ITS 2016) – Информационные технологии и системы 2016 (ИТС 2016), Minsk, Belarus, 26 October 2016; pp. 210–211.
80. Rosenblatt, S.; Fainstein, D.; Cestero, A.; Safran, J.; Robson, N.; Kirihata, T.; Iyer, S.S. Field Tolerant Dynamic Intrinsic Chip ID Using 32 nm High-K/Metal Gate SOI Embedded DRAM. *IEEE J. Solid-State Circuits* **2013**, *48*, 940–947.
81. Rosenblatt, S.; Chellappa, S.; Cestero, A.; Robson, N.; Kirihata, T.; Iyer, S.S. A Self-Authenticating Chip Architecture Using an Intrinsic Fingerprint of Embedded DRAM. *IEEE J. Solid-State Circuits* **2013**, *48*, 2934–2943.
82. Liu, W.; Zhang, Z.; Li, M.; Liu, Z. A Trustworthy Key Generation Prototype Based on DDR3 PUF for Wireless Sensor Networks. *Sensors* **2014**, *14*, 11542–11556.
83. Liu, W.; Zhang, Z.; Li, M.; Liu, Z. A Trustworthy Key Generation Prototype Based on DDR3 PUF for Wireless Sensor Networks. In Proceedings of the 2014 International Symposium on Computer, Consumer and Control (IS3C), Taichung, Taiwan, 10–12 June 2014; pp. 706–709.
84. Zhang, Z. Design and Implementation of DRAM PUF (in Chinese). Master's Thesis, Huazhong University of Science and Technology, Wuhan, Hubei, China, 2015.
85. Rahmati, A.; Hicks, M.; Holcomb, D.E.; Fu, K. Probable Cause: The Deanonymizing Effects of Approximate DRAM. In Proceedings of the 2015 ACM/IEEE 42nd Annual International Symposium on Computer Architecture (ISCA), Portland, Oregon, 13–17 June 2015; pp. 604–615.
86. Sutar, S.; Raha, A.; Raghunathan, V. D-PUF: An Intrinsically Reconfigurable DRAM PUF for Device Authentication in Embedded Systems. In Proceedings of the 2016 International Conference on Compilers, Architectures, and Sythesis of Embedded Systems (CASES), Pittsburgh, PA, USA, 2–7 October 2016.
87. Sutar, S.; Raha, A.; Raghunathan, V. Memory-Based Combination PUFs for Device Authentication in Embedded Systems. *arXiv* **2017**, arXiv:1712.01611.

88. Tehranipoor, F.; Karimian, N.; Yan, W.; Chandy, J.A. Investigation of DRAM PUFs Reliability Under Device Accelerated Aging Effects. In Proceedings of the 2017 IEEE International Symposium on Circuits and Systems (ISCAS), Baltimore, MD, USA, 28–31 May 2017.
89. Kumar, R.; Xu, X.; Burleson, W.; Rosenblatt, S.; Kirihata, T. Physically Unclonable Functions: A Window into CMOS Process Variations. In *Circuits and Systems for Security and Privacy*; Sheikh, F., Sousa, L., Iniewski, K., Eds.; CRC Press: Boca Raton, FL, USA, 2017.
90. Kirihata, T.; Rosenblatt, S. Dynamic Intrinsic Chip ID for Hardware Security. In *VLSI: Circuits for Emerging Applications*; Wojcicki, T., Iniewski, K., Eds.; CRC Press: Boca Raton, FL, USA, 2017.
91. Raha, A.; Jayakumar, H.; Sutar, S.; Raghunathan, V. Quality-Aware Data Allocation in Approximate DRAM. In Proceedings of the 2015 International Conference on Compilers, Architecture and Synthesis for Embedded Systems, Amsterdam, The Netherlands, 4–9 October 2015; pp. 89–98.
92. Raha, A.; Sutar, S.; Jayakumar, H.; Raghunathan, V. Quality Configurable Approximate DRAM. *IEEE Trans. Comput.* **2017**, *66*, 1172–1187.
93. Anagnostopoulos, N.A.; Schaller, A.; Fan, Y.; Xiong, W.; Tehranipoor, F.; Arul, T.; Gabmeyer, S.; Szefer, J.; Chandy, J.A.; Katzenbeisser, S. 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, 1–2 June 2017.
94. Yan, W.; Tehranipoor, F.; Chandy, J.A. PUF-Based Fuzzy Authentication Without Error Correcting Codes. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst. (CAD)* **2017**, *36*, 1445–1457.
95. Yan, W.; Tehranipoor, F.; Chandy, J.A. A Novel Way to Authenticate Untrusted Integrated Circuits. In Proceedings of the 2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD 2015), Austin, TX, USA, 2–6 November 2015; pp. 132–138.
96. Maes, R.; Tuyls, P.; Verbauwhede, I. A Soft Decision Helper Data Algorithm for SRAM PUFs. In Proceedings of the 2009 IEEE International Symposium on Information Theory (ISIT 2009), Seoul, South Korea, 28 June–3 July 2009; pp. 2101–2105.
97. Anagnostopoulos, N.A. Optical Fault Injection Attacks in Smart Card Chips and an Evaluation of Countermeasures Against Them. Master's Thesis, University of Twente, Enschede, The Netherlands, 2014.
98. Karimian, N.; Wortman, P. A.; Tehranipoor, F. Evolving Authentication Design Considerations for the Internet of Biometric Things (IoBT). In Proceedings of the 2016 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS 2016), Pittsburgh, PA, USA, 2–7 October 2016; pp. 1–10.