D-PUF: An Intrinsically Reconfigurable DRAM PUF for Device Authentication in Embedded Systems*

Soubhagya Sutar, Arnab Raha, and Vijay Raghunathan
School of Electrical and Computer Engineering, Purdue University
{ssutar,araha,vr}@purdue.edu

ABSTRACT

Physically Unclonable Functions (PUFs) have proved to be an effective and low-cost measure against counterfeiting by providing device authentication and secure key storage services. Memory based PUF implementations are an attractive option due to the ubiquitous nature of memory in electronic devices and the requirement of minimal (or no) additional circuitry. DRAM based PUFs are particularly advantageous due to their large address space and multiple controllable parameters during response generation. However, prior works on DRAM PUFs use a static response generation mechanism making them vulnerable to security attacks. Further, they result in very slow device authentication, are not applicable to off-the-shelf DRAM modules, or require DRAM power cycling prior to authentication.

In this paper, we propose D-PUF, an *intrinsically reconfigurable* DRAM PUF based on refresh pausing. A key feature of the proposed DRAM PUF is that it can be reconfigured by varying the refresh-pause interval, which changes it's challenge-response behavior, making it robust against various attacks. We use the PUF to design a secure, low-overhead methodology for performing device authentication. The design is implemented and validated using an Altera Stratix IV GX FPGA based Terasic TR4-230 development board and several off-the-shelf 1GB DDR3 DRAM modules. Our experimental results demonstrate a `4.3X-6.4X` reduction in authentication time, compared to previous work. Using controlled temperature and accelerated aging tests, we also demonstrate the robustness of our authentication mechanism to temperature variations and aging effects.

CCS Concepts

•**Security and privacy** → **Embedded systems security;** *Hardware security implementation;* •**Hardware** → **Dynamic memory;** *Integrated circuits;*

Keywords

Hardware Security, Physically Unclonable Function, Memory PUF, Authentication

1. INTRODUCTION

The last decade has seen a rapid proliferation of embedded computing devices, fueled in part by the advent of the Internet-of-Things (IoT) era. The increasingly network-connected nature of these devices, coupled with their ability to access potentially sensitive or confidential information, has given rise to a plethora of new security and privacy concerns. A particularly worrisome trend is the growing number of counterfeit Integrated Circuits (ICs) and Intellectual Property (IP) cores in these devices, which not only has profound security and reliability implications, but has also led to a tremendous revenue loss (estimated to be in excess of $100 billion in annual global revenue) to the electronics industry [1]. To combat this challenge, researchers have proposed the use of various hardware-intrinsic security mechanisms such as Physically Unclonable Functions (PUFs) [2], which have proved to be a secure, low-cost, and robust authentication measure against issues of counterfeiting and information leakage. PUFs exploit the random physical variations inherent to any manufacturing process to generate device-specific and unclonable fingerprints. These unique fingerprints can be used as the basis for a challenge-response mechanism for device authentication or secure key generation.

Figure 1: Examples of embedded systems containing DRAM

While a large variety of PUF implementations have been proposed, memory-based PUFs [3, 4, 5], in particular, are an attractive candidate due to the ubiquitous presence of memory in virtually every embedded system. Moreover, these memory-based PUFs do not require any additional circuitry for their operation, giving them a distinct advantage over other PUF implementations. One such example of a memory-based PUF, which we focus on in this paper, is based on Dynamic Random Access Memory (DRAM). DRAM is used as main memory in a large number of modern embedded systems (as illustrated in Figure 1) due to its high density and low cost. As is well-known, DRAM cells must be refreshed periodically to preserve the stored data. This need for refresh operations also makes DRAM an attractive candidate for use as a PUF, especially for challenge-response based authentication [3, 4, 6]. The key idea of a DRAM PUF based on refresh pausing is as follows (a more detailed explanation is given in Section 3). If data is stored in a large (say 64KB) block of DRAM cells and refresh operations to the entire block are paused for an extended amount of time (henceforth referred to as the *refresh-pause interval*), some of the DRAM cells in the block will lose their data. *How many* and *which* cells in the DRAM block lose their contents (for a given refresh-pause interval) is unique to a device and can, therefore, be used as the basis for implementing a PUF.

Prior approaches to DRAM-based PUFs suffer from several shortcomings such as low speed of authentication [3], non-applicability to commercial off-the-shelf (COTS) devices [6], and the need for

*This work was supported in part by the National Science Foundation under grants CNS-0953468 and CNS-1527829.

CASES '16, October 01-07, 2016, Pittsburgh, PA, USA

DOI: http://dx.doi.org/10.1145/2968455.2968519

power cycling the DRAM module prior to authentication [4]. Moreover, the near static nature of the response generation mechanism in some of the works [3, 4] makes them vulnerable to various security attacks [7]. To address these limitations, we propose a DRAM PUF based on refresh pausing that not only supports a very large number of challenge-response pairs (CRPs) through the variation of different parameters, but is also *intrinsically reconfigurable*, *i.e.*, its challenge-response behavior can be substantially modified without the use of any additional circuitry. Hence, the PUF can be easily implemented in most off-the-shelf systems and provides considerable protection from various security attacks. Specifically, this paper makes the following contributions:

- We perform a comprehensive error characterization of DRAM modules by varying different parameters (refresh-pause interval, data patterns, and temperature) to gain a deep insight into DRAM behavior. This insight allows us to systematically select DRAM blocks that are best suited for use as a PUF.
- We propose an *intrinsically reconfigurable* DRAM PUF (D-PUF), based on refresh pausing, for device authentication. Reconfiguration is achieved through variation of the refresh-pause interval and changes the challenge-response behavior of the PUF, making it robust against various attacks. We use D-PUF to design a secure, low-overhead methodology for performing device authentication. The methodology operates robustly even in the presence of environmental and temporal variations.
- We implement D-PUF and our proposed authentication mechanism in a real system using off-the-shelf DRAM modules and evaluate it thoroughly. In particular, we demonstrate a `4.3X-6.4X` reduction in authentication time, compared to previous work. Using controlled temperature and accelerated aging tests, we demonstrate the robustness of our authentication mechanism to temperature and aging effects. Results demonstrate `100%` true-positive (successful authentication) rate for a 10°C temperature variation with `0%` false-positive rate. For a nine-month old DRAM module, our authentication mechanism also ensured `100%` true-positive rate and `0%` false-positive rate.

The remainder of this paper is organized as follows. Sec. 2 gives a brief description of DRAM operation and how PUFs work. Sec. 3 explains the design of the proposed reconfigurable DRAM PUF and the associated authentication methodology. Sec. 4 and Sec. 5 describe our experimental setup and present the results of our experiments, respectively. A discussion about implementation-related considerations and a summary of prior work in the PUF domain are provided in Sec. 6 and Sec. 7, respectively. Finally, Sec. 8 concludes the paper.

2. BACKGROUND

Before describing our design, we briefly explain the principles behind PUFs and DRAM refresh operation.

2.1 Physically Unclonable Functions

A PUF [2] maps a set of challenges to a set of responses based on random physical variations during the manufacturing of a device (containing the PUF). As a result, the challenge-response behavior of the PUF is highly unpredictable. In addition, the fact that it is impossible to manufacture a PUF with the same behavior as another, makes it unclonable and unique. These features make PUF an ideal candidate for generation of secret keys and authentication. Secret keys are used by various cryptographic applications such as keyed-hash message authentication code (HMAC), encryption/decryption, *etc.,* besides serving as unique fingerprints or signatures that could be used to identify a device. PUFs enable the generation of secret keys on demand rather than permanently storing them in non-volatile memory, drastically reducing the implications of physically invasive attacks. Authentication can be considered an extension of the above key generation process but involves a multiple challenge-response mechanism in order to authenticate the target device. This is further explained in Sec. 3.

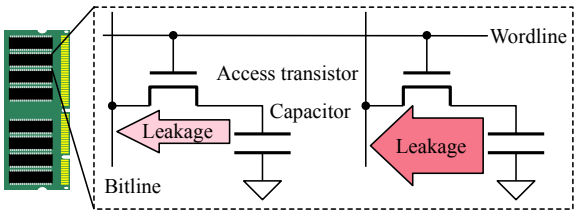

Figure 2: Variation in DRAM cell leakage

PUFs have been divided into two broad categories [8] - strong PUFs and weak PUFs. Strong PUFs can support a very large number of CRPs and are well suited for authentication. Weak PUFs, on the other hand, are primarily used for secret key generation as they support a relatively much smaller number of CRPs. As will be clear in the following sections, our proposed DRAM PUF closely resembles a strong PUF.

2.2 Dynamic Random Access Memory

The fundamental building block of a DRAM module is a DRAM bit-cell, which consists of a single capacitor and an access transistor, as shown in Figure 2. The bit-cells are arranged in a two-dimensional matrix structure and carry a binary value of '0' or '1', depending on whether the capacitors are in a fully charged or discharged state. The capacitors, however, leak charge over time due to various factors related to the non-ideality of the access transistors, *e.g.*, sub-threshold leakage, gate-induced drain leakage, *etc*. This charge leakage will eventually result in loss of data stored in the DRAM bit-cell after a certain time interval and hence, requires the bit-cell to be refreshed (charge replenishment) periodically. The memory controller, in charge of the DRAM module, uses a single refresh rate for simplicity and refreshes each row every 64 ms (standard refresh interval) for guaranteeing data integrity.

3. DESIGN

Authentication involves the use of CRPs in order to authenticate/identify a device, *e.g.,* a client device trying to connect to a restricted network first needs to be authenticated by a trusted authority such as the network gateway. Towards this objective, we present the design of our proposed *intrinsically reconfigurable* DRAM PUF (D-PUF). Subsequently, we describe a secure, low-overhead methodology that uses D-PUF and performs device authentication without the need for additional cryptographic resources that are used in traditional encrypted-password based methods. We also provide a low-complexity algorithm for selecting the blocks in D-PUF that ensure the minimum required entropy at the lowest refresh-pause intervals.

3.1 D-PUF

We first describe the motivation behind our choice of DRAM for implementing a PUF and then present our proposed design of an *intrinsically reconfigurable* DRAM PUF.

3.1.1 Motivation

Memory-based PUFs have a distinct advantage over other PUF implementations as they use components (SRAM, DRAM, *etc.*) that are inherent to most modern embedded systems. Hence, they require minimal or no additional circuitry for their operation and could enable energy-efficient designs for emerging IoT devices [9]. Over the past few years, SRAM-based PUFs have been widely studied and used as security primitives for various state-of-the art systems [5, 10]. However, these PUFs suffer from several shortcomings such as limited entropy, requirement of power cycling, high cost, *etc.*, and are hence, limited to applications based on secure key generation only. Authentication, however, demands a

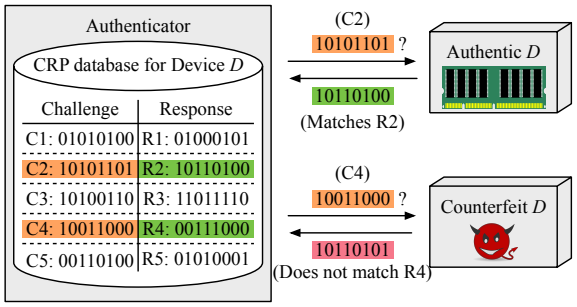

Figure 3: Authentication using D-PUF

challenge-response mechanism utilizing multiple CRPs. DRAM based PUFs, with their large address space, high density and dynamic challenge-response behavior, can generate a very large number of CRPs for use during authentication.

A light weight approach towards implementing DRAM PUFs is to use refresh pausing. Refresh pausing involves stopping the refresh operation in a DRAM module for a specified interval. In other words, the refresh operation is carried out at an interval higher than the standard 64 ms leading to improved memory performance and refresh energy savings at the cost of bit-errors [11, 12]. In DRAMs, inter-die and intra-die variations lead to highly variable bit-cell strengths distributed randomly across different modules as well as within a module. During refresh pausing, these variations cause DRAM cells to leak charge at different rates (as shown in Figure 2) resulting in highly random yet unique bit-errors in the data stored in the DRAM module. We exploit this randomness (entropy) to derive a fingerprint or secret key out of a DRAM module, as well as generate CRPs for authentication. Moreover, the existence of bit-cells experiencing both '1' $\rightarrow$ '0' bit-flips (true-cells) and '0' $\rightarrow$ '1' bit-flips (anti-cells) [13] enables us to extract more randomness out of the DRAM PUF.

3.1.2 *Intrinsically reconfigurable DRAM PUF*

Most DRAM PUF implementations are not strong PUFs and are vulnerable to various sophisticated attacks [7]. One way of guarding against such attacks is altering the challenge-response behavior of a PUF, in other words, *reconfiguring* the PUF [10, 14, 15]. The new challenge-response behavior is unpredictable and cannot be modeled based on the knowledge of the PUF behavior before reconfiguration. Towards this end, we propose an *intrinsically reconfigurable* DRAM PUF (D-PUF) based on refresh pausing. The reconfigurability is *intrinsic* in nature because our choice of refresh-pause interval as the reconfiguration parameter enables alteration of the PUF behavior without the requirement of any additional resource. Also, it must be noted that the refresh-pause intervals that are chosen ensure considerable entropy difference before and after reconfiguration. Hence, an attacker cannot predict the behavior of the reconfigured PUF completely even if he knows the behavior of the same before reconfiguration.

Next, we briefly describe the sequence of steps involved in the generation of a response from D-PUF. A random binary bit-stream of a particular size, referred to as challenge, is first written onto a specified memory address in D-PUF, following which, the memory controller pauses the refresh operation for a pre-decided time interval. Next, the data bit-stream is read out from the same memory address and processed for subsequent error-correction using some data already stored at the device containing the PUF. The error corrected bit-stream is then sent out as the response. Whenever needed, reconfiguration is carried out by simply changing the refresh-pause interval associated with the generation of the response.

3.2 Design Overview

The overall authentication methodology is divided into three phases - *(i) characterization* phase, *(ii) enrollment* phase, and *(iii) authentication* phase. Figures 6, 7, and 8 present flow diagrams of the different phases. The dotted lines between the device (containing the PUF) and the authenticator represent the interactions between them, while the solid lines show the actions inside each of these entities. The *characterization* phase ensures the usage of minimum refresh-pause interval for authentication by carrying out a coarse-grained error characterization of the DRAM module used as the PUF and simultaneously guarantees that it meets the minimum required entropy. During *enrollment* phase, the PUF responds to random challenges sent by an authenticator, generating a CRP database that is stored at the latter. It is followed by the *authentication* phase, where responses are generated by the PUF for one or more challenges picked from the CRP database. These responses are then compared against the ones stored in the database; if there is an exact match or the difference is within a small threshold, the device containing the PUF is authenticated. Note that, by varying different parameters associated with the generation of a response, a very large number of CRPs can be generated for carrying out CRP-based authentication. Hence, D-PUF closely resembles a strong PUF. Figure 3 provides a high-level overview of the entire process.

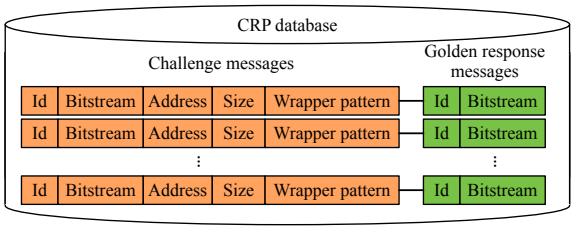

Figure 4: CRP database

Before delving into the details of the authentication methodology, we define a few terms and briefly discuss our assumptions. We also present the formats for the challenge and response used in our design below.

3.2.1 *Definitions and assumptions*

- *Device* (*D*): An untrusted client device that requests authentication and contains D-PUF. It is assumed to possess sufficient computational and memory resources to carry out error correction and store kilobytes of binary data. An example of such a device is a smartphone.

- *Reconfigurable DRAM PUF or D-PUF* (*P*): The DRAM module that implements PUF functionality in the device *D*. Note that, when we mention that a response is generated by *D*, it should be assumed that the same is actually generated by D-PUF (*P*) present in *D*.

- *Authenticator* (*A*): A trusted party which authenticates the device *D*. *A* is assumed to have access to the CRP database and limited information about the characteristics of the PUF present in *D*. It possesses relatively greater computational and memory resources than *D* in order to process and store gigabytes of data, *e.g.*, a server.

3.2.2 *Challenge and response message format*

In the proposed design, challenges and responses are represented as 5-tuple and 2-tuple messages respectively, as depicted in Figure 4. An entry in the CRP database comprises of a *challenge* message (*CM*) and a *golden response* message (*GRM*) (Sec. 3.2.4). The response generated during authentication is referred to as a *response* message (*RM*). *Id* refers to the index number assigned by *A* to an entry in the CRP database. *Bitstream* is a random binary sequence of *size* bytes generated by *A* (in a *CM*) or *D* (in an *RM* and a *GRM*). In a *CM*, *bitstream* is the data written onto *P* while in an *RM* and a *GRM*, it refers to the data read from *P*. *Address*

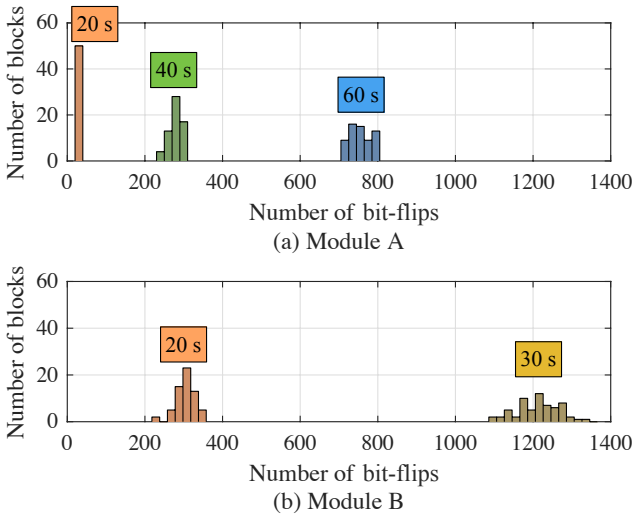

Figure 5: Variation of bit-flips across different DRAM modules

specifies the start memory address in *P* where this write or read occurs. *Wrapper pattern* represents a predefined binary sequence and is explained next.

Wrapper pattern: We observed that the number, position, and nature of bit-flips in a response generated by a memory block in *P* were influenced by the peripheral data-bits surrounding the block (Sec. 5.1.3). These peripheral data-bits, written just before the beginning and after the end of the block, are collectively referred to as wrapper data and are specified by a *wrapper pattern*. The *wrapper pattern* is a part of the challenge message and can be one of several predefined types, *e.g.,* all 1s, all 0s, checkered, *etc*. The challenge bitstream is padded with the corresponding wrapper data before it is written into the block. However, the wrapper data is not part of the response message that is sent back to *A*. *Wrapper pattern*, thus, serves as another variable parameter for CRP generation.

3.2.3 Characterization phase

Error characterization provides valuable insights into the behavior of DRAM modules, some of which are presented below. These insights enable D-PUF to have a lower authentication overhead compared to prior art, while meeting the specified entropy requirements. In addition, characterization carried out at different temperatures guides the design of a robust authentication methodology. Figure 5 shows the characterization results for two DRAM modules at different refresh-pause intervals with other parameters remaining the same. A few key observations from these characterization results that are leveraged in the D-PUF design are given below:

1. The choice of a conservative refresh-pause interval [3], though satisfies entropy requirements, can lead to very slow authentication in refresh pausing based DRAM PUFs.

2. Given a refresh-pause interval, some blocks in the DRAM address space may not contain the required entropy (bit-flips), making them unsuitable for PUF operation. For example, only some of the blocks in Module A satisfy an entropy requirement of 250 bit-flips at 40s refresh-pause interval, as shown in Figure 5(a).

3. The variation in entropy is more pronounced across modules, making a constant refresh-pause interval potentially unsuitable for some modules. For example, for the same entropy requirement of 250 bit-flips, a relatively lower interval of 20s is suitable for Module B, as shown in Figure 5(b), unlike in the case of Module A.

4. Higher temperatures will result in an exponential rise in the number of bit-flips, which can potentially hinder the authentication process, as described later in Sec. 5.3.1.

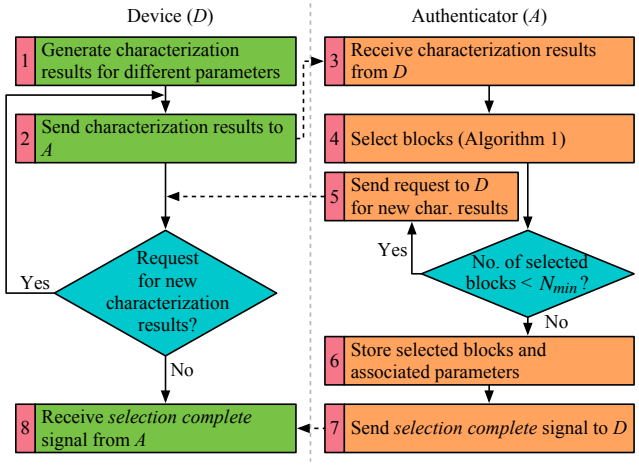

Figure 6: D-PUF *Characterization* phase

A coarse-grained characterization of a DRAM module (*P*) was carried out with standard input-data patterns and different refresh-pause intervals, the details of which are given in Sec. 4. Figure 6 provides the flow diagram of our proposed *characterization* phase where the numbers adjacent to each step specify the sequence of steps followed during a phase. It starts with the generation of characterization results as described in Sec. 4. It should be noted that DRAM manufacturers may choose to provide these results in the DRAM module itself, which could then be utilized by the PUF designer. The authenticator uses these characterization results to select blocks that meet the minimum entropy requirements at a refresh-pause interval. The block selection process is presented as pseudo-code in Algorithm 1 and is described next. Note that an attacker may be able to get some insights into the DRAM behavior if he has access to these characterization results. As a result, it needs to be ensured that the characterization results are shared with the authenticator securely. Therefore, we assume that the *characterization* phase occurs in a secure environment [10, 16, 17]. Also, the *characterization* phase is much less frequent than the other two phases as it occurs only before initial deployment or in case *D* runs out of address space (in *P*) for the PUF operation.

Block selection: The characterization results provide insights into the number, position, and nature of bit-flips within a particular module, which could then be utilized to choose the best blocks with respect to entropy requirements and at the optimum refresh-pause intervals. Algorithm 1 describes the proposed algorithm for the selection of blocks. During *enrollment* phase (Sec. 3.2.4), *D* sends the characterization results of a sub-address space (*S*) (in *P*) for a mutually-decided refresh-pause interval (*t*) and different input-data patterns (*C*) to *A*. Block selection at *A* starts by finding the bit-flip positions (represented as '1' in F_i) for each of the input-data patterns followed by combining them to generate all possible bit-flip positions (*F*). Beginning with the lowest specified block-size first, selection of blocks is carried out such that they meet the minimum entropy requirements (E_{min}). E_b represents the entropy of a block *b* and is specified by the number of '1's in *F* corresponding to that block. Moreover, the selected blocks are non-overlapping, *i.e.,* they do not share any memory address (and bit-flips) with each other. Though it may seem like a conservative approach, it ensures that the bit-flip positions are not shared among any two blocks and hence, selects blocks with sufficient number of unique bit-flips.

An aggressive (very low) choice for refresh-pause interval may lead to none (or very few) of the blocks in a particular sub-address space (*S*) meeting the entropy requirements. Also, as the characterization process is computation and time-intensive, sufficient number of blocks (greater then N_{min}) should be selected upfront to create enough CRPs without the need for re-characterizing anytime soon. In such a situation, *A* may request *D* to provide either the charac-

Algorithm 1: Pseudo-code for block selection

Input: $I = \{I_1, I_2, ..., I_n\}$: Set of input data-patterns for characterization,
$C = \{C_1, C_2, ..., C_n\}$: Set of characterization results of a sub-address space S at a refresh-pause interval t for different input data-patterns,
$Z = \{Z_1, Z_2, ..., Z_m\}$: Set of specified block-sizes in increasing order,
E_{min} = Minimum required entropy specified in terms of number of bit-flips
Output: $L = \{L_1, L_2, ..., L_m\}$: Set of list of selected blocks where L_i is the list of blocks with size Z_i

```
F = ϕ, L = {ϕ}   // F represents combined bit-flips for S
for i = 1 to n do
    F_i = C_i ⊕ I_i   // Bitwise XOR
    F = F + F_i   // Bitwise OR
for j = 1 to m do
    B_j = Get_All_NonOverlapBlocks(S, Z_j);
for k = 1 to m do
    foreach b ∈ B_k do
        E_b = Get_Entropy(b, F)
        if E_b > E_min then
            OL = Check_Overlap(b, L)
            if OL = False then
                L_k = L_k ∪ b
    L = L ∪ L_k
```

terization results for a new sub-address space (S') at t or those for S at a higher refresh-pause interval (t'). The former will enable fast authentication but will also exhaust the address space fast. This may be suitable for PUFs that demand fast authentication but require a small number of CRPs or have a large address space. The latter, on the contrary, will result in relatively slower authentication but will also exhaust the address space slowly. This may be suitable for devices that can tolerate slower authentication but require a large number of CRPs or have a small address space. The PUF designer can take the decision based on the *authentication speed vs. available address space* trade-off.

3.2.4 Enrollment phase

The *enrollment* phase primarily involves the generation of the CRP database (Figure 4) and is also assumed to be carried out in a secure environment [10, 16, 17]. Figure 7 presents the flow diagram for the *enrollment* phase. First, a refresh-pause interval t is mutually decided by A and D. Initially, t is the minimum interval that meets entropy requirements but may assume higher values later due to reconfiguration. Note that during an attack by an adversary or scenarios such as exhaustion of usable CRPs, authentication involving multiple authenticators, *etc.*, P can start operating at a different interval, generating a new and unpredictable response. Next, a sub-address space S (in P) is selected by D to be used for PUF operation. This choice is based on available sub-address spaces as well as results obtained from the *characterization* phase. The CRP database is then generated by A by sending challenge messages (*CM*s) to D, which responds by sending back golden response messages (*GRM*s). The golden responses together with the corresponding challenges form entries in the CRP database against which subsequent responses are compared during the *authentication* phase. The helper data, for error correction, is also generated during the *enrollment* phase (described next) and is stored at D. Note that the effects of variable retention time (VRT)

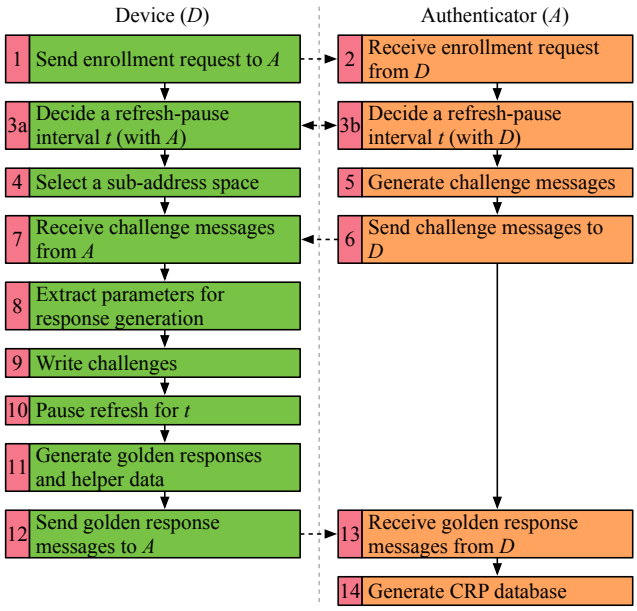

Figure 7: D-PUF *Enrollment* phase

during the generation of golden responses could be mitigated by acquiring multiple instances of the golden responses (for the same challenge) and taking a bit-wise intersection of the same.

Helper data for error correction: PUFs are governed by random physical processes, which are affected by numerous environmental and temporal variations. Error correction can be utilized to suppress some of these variations and enhance the reliability of the response generation process. During the *enrollment* phase, some redundant information is also generated from the golden responses that is capable of correcting the subsequent responses from D. This redundant information, known as *helper* data, is stored at D. In our work, helper data was generated by implementing a lightweight error correction algorithm - (31,26) Hamming Encoder/Decoder in software. Our experiments showed that this was enough to correct most of the errors at room temperature (20°C) and ensured successful authentication. Although more complex error correction [10] can also be utilized, the choice is finally left to the PUF designer and is orthogonal to the core idea of this work.

3.2.5 Authentication phase

Actual authentication of D is assumed to happen in an insecure environment during the *authentication* phase and is depicted in Figure 8. It starts with a request for authentication from D, which is followed by a challenge sent by A. A response is then generated by D, corrected for errors, and sent to A. Upon receiving the response, A calculates its hamming distance (HD) with the golden response stored in the CRP database. This HD is then compared with the *match threshold* (described next) to determine the authentication outcome.

Match threshold: The existing error-correction infrastructure in D may not be sufficient to correct all the errors in the response generated during the *authentication* phase. This is more pronounced in DRAMs due to their high entropy (bit-flips) and susceptibility to several environmental and temporal variations. In such scenarios, an exact match of the generated response with the CRP database may not happen even if D is authentic. Hence, we follow a fuzzy authentication strategy and define a *match threshold* (*MT*), the maximum HD between the *GRM bitstream* and *RM bitstream* beyond which D is not authenticated. Refs. [5, 16] use a similar technique for unique identification of devices.

In order to set the appropriate value of *MT*, we refer to Figure 13 that shows a probability distribution of the HD for five different DRAM modules at three temperatures and 50 different *CM*s. *MT*

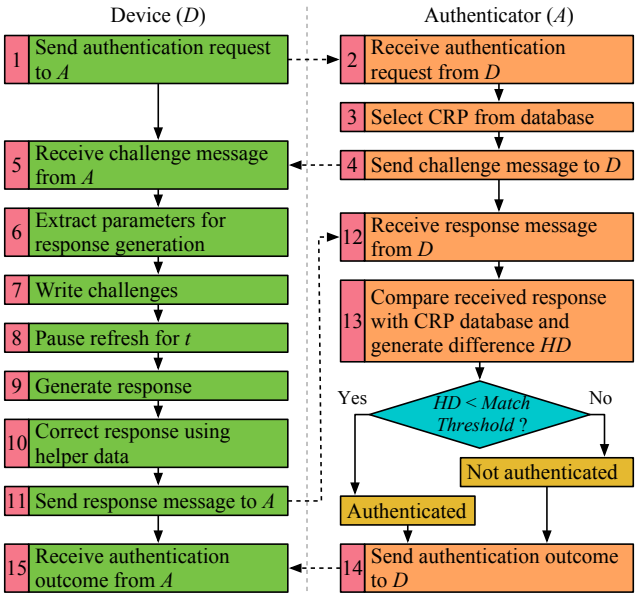

Figure 8: D-PUF *Authentication* phase

is given as $MT = \mu + \sigma + \tau$, where μ and σ represent the *mean* and *standard deviation* of the distribution (*same-module* comparisons). We include τ to accommodate for environmental (temperature) and temporal (aging, variable retention time, *etc.*) variations. This enables us to perform robust authentication even under high variations. Sec. 5.3.2 explains this in detail. Note that setting the match threshold can be done by the authenticator during the *characterization* phase on the basis of characterization results and, thus, represents a one-time overhead.

4. EXPERIMENTAL METHODOLOGY

4.1 Experimental Setup

This section provides a brief description of the experimental setup used to validate the D-PUF design. All experiments were performed using an Altera Stratix IV GX FPGA based Terasic TR4-230 development board [18], consisting of a 1GB SODIMM DDR3 DRAM. The entire experimental setup is shown in Figure 9.

The FPGA was programmed with a soft Nios II processor [19] along with an Altera UniPHY DDR3 memory controller for controlling the DRAM module. This controller provides the control signals required to pause the refresh operations. In modern embedded systems, these refresh control signals are usually exposed to the lower layers of the operating system. A custom slave running on the processor was also created, which can instruct the memory controller to start and stop the refresh operations. A total of six COTS 1GB DRAM DDR3 SODIMMs belonging to five different manufacturers were used for the experiments. The temperature and aging experiments were carried out by operating the DRAM modules inside the Quincy Lab 12-140E Incubator. Note that, during validation, error-correction was performed using software running on the Nios II processor, while the authentication was done on a local computer connected directly to the FPGA.

4.2 Characterization Methodology

We now describe the details of the DRAM error characterization process, which was performed in a number of sequential steps:

1. First, an input-data pattern was written throughout the selected DRAM sub-address space. Subsequently, the DRAM was refreshed normally (at 64 ms) so that 100% data is retained.

2. Next, the custom slave (implemented on the FPGA) disabled the refresh and waited for the selected refresh-pause interval, which was maintained by a precise timer controlled directly by the FPGA hardware.

3. The data from the DRAM was then read out and normal refresh operation was restored. The acquired data was compared with the input-data to determine the number, position, and nature of bit-flips.

4. This process was repeated for different data patterns (*e.g.,* all '1's, all '0's, checkered pattern (alternate '0's and '1's), *etc.*,) as well as different refresh-pause intervals (20s, 30s, 40s, 60s, 80s, and 90s) and temperatures (20°C, 30°C, and 40°C).

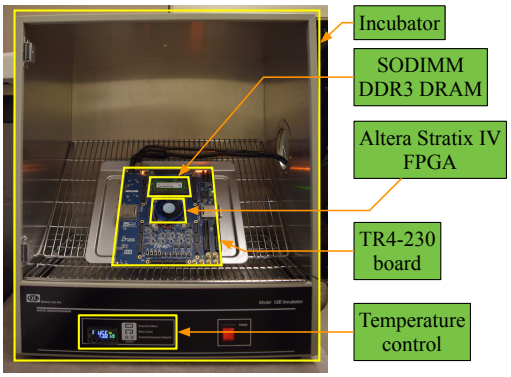

Figure 9: Experimental setup

In a real system, we envision that a fixed segment of the DRAM (say 5%) will be dedicated for PUF functionality so that the DRAM can be shared simultaneously with other tasks running on the device. For generic DDR DRAMs, this segment can be selected either randomly or by an initial lightweight characterization process, where the PUF section will be refreshed at a much higher refresh interval than the rest of the module. The Partial Array Self-Refresh functionality [20] in LPDDRs inherently meets this requirement and can refresh a portion of the DRAM at a different interval than the standard 64 ms.

5. EXPERIMENTAL RESULTS

This section presents the results obtained from experiments conducted to validate our work. It is divided into four parts. First, we show the effects of variation of different parameters (refresh-pause interval, block-size, and wrapper pattern) on the number of bit-flips in one of the DRAM modules. An analysis of the results obtained provides useful insights for setting design parameters associated with the proposed authentication methodology. An uniqueness analysis of the responses obtained from different DRAM modules as well as from different blocks belonging to the same DRAM module is presented next and shows the effectiveness of D-PUF for authentication. We then analyze the robustness of our proposed authentication methodology under temperature and aging effects using five DRAM modules. Finally, we present the test results for the authentication of a sample DRAM module.

5.1 Effects of Parameter Variations

We characterized 512 memory blocks in a DRAM module using the process described in Sec. 4.2. The results are presented in Figures 10, 11, and 12. Table 1 provides a summary of the parameter values used for each characterization.

5.1.1 Variation of bit-flips with refresh-pause interval

As described in Sec. 3.1.2, reconfigurability is achieved in the presented design by modifying the refresh-pause interval. Figures 10(a) and 10(b) show the variation of bit-flips with refresh-pause interval across 512 blocks in a DRAM module. Figure 10(a)

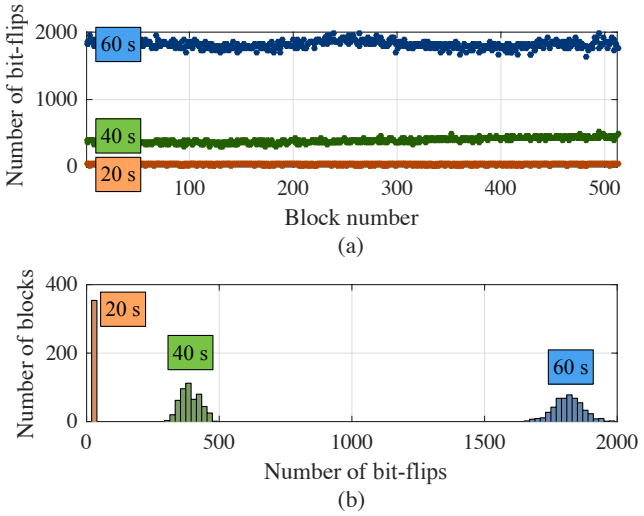

Figure 10: Variation of bit-flips with refresh-pause interval

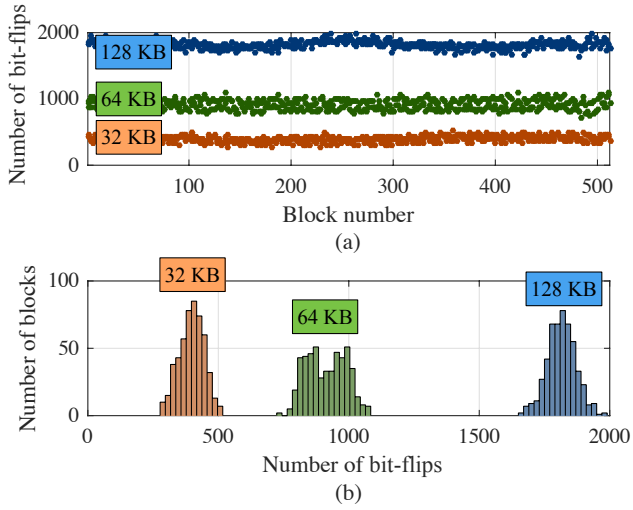

Figure 11: Variation of bit-flips with block-size

Table 1: Parameter values used in our evaluation

Figure	Pause-int.	Blk. size	Address	Wrp. Pattern
Fig. 10	Varied	128 KB	0x38000000 – ···	All '0's
Fig. 11	60s	Varied	0x38000000 – ···	All '0's
Fig. 12	60s	128 KB	0x38000000 – ···	Varied

depicts the number of bit-flips across different blocks while Figure 10(b) shows the number of blocks having a particular number of bit-flips. Assuming a minimum required entropy of 400 bit-flips, it can be clearly seen that a refresh-pause interval of 20s is unsuitable for PUF operation. On the other hand, an interval of 60s generates entropy in excess of 1200 bit-flips across all the blocks while 40s does it across some of the blocks, hence, both are suitable refresh-pause intervals. The entropy variation across suitable intervals is quite high, thus, reinforcing our intuition for the use of refresh-pause interval as the parameter for reconfiguration. Moreover, Figure 10 provides us the minimum interval (for the given DRAM module) that meets the entropy requirements. As compared to Ref. [3], our choice of the minimum refresh-pause interval (60s) reduces the authentication time by `4.3X` for an entropy requirement of 512 bit-flips. Note that, the minimum interval may vary from one module to another, *e.g.,* generation of the same entropy for some of the other modules required a minimum interval of 40s.

5.1.2 Variation of bit-flips with block-size

Blocks of different sizes starting at the same address can contain widely varying bit-flips. Figures 11(a) and (b) show the variation of bit-flips with block-size across 512 blocks in a DRAM module. The block-size, which is a part of the challenge message, can be varied to generate more CRPs for authentication. Also, for a relatively resilient (to bit-flips) DRAM module, a designer may need to use higher block-sizes at a given refresh-pause interval in order to meet the minimum entropy requirements. For example, an entropy of 1000 bit-flips is met by all 128 KB blocks and some 64 KB blocks but not by any of the 32 KB blocks, as shown in Figure 11. Characterization provides us with valuable insights for choosing the minimum block-size for a given interval.

5.1.3 Variation of bit-flips with wrapper pattern

An interesting observation of the characterization process is the variation of bit-flips in a block with the wrapper data (or pattern) surrounding the block. Figure 12 shows the variation of bit-flips with the wrapper pattern across 512 blocks in a DRAM module. The wrapper pattern serves as an additional parameter that can be varied to extract more entropy (and hence, more number of CRPs) from a DRAM module. However, not all blocks respond to the variation in wrapper pattern, as is evident in the first half of Figure 12(a). Thus, a careful block-specific utilization of this parameter is required. An off-shoot of this analysis is the revelation of the pattern that generates maximum bit-flips (all '0's here), enhancing the entropy of the block.

5.2 Uniqueness Analysis

In order to demonstrate the uniqueness offered by D-PUF, we generated response bit-streams (or fingerprints) from five different DRAM modules using the same refresh-pause interval and other parameters. This is shown in Figure 15, where different colors represent the degree of variation in the number and position of bit-flips in the generated bit-streams. A dark (blue) color represents a smaller number of bit-flips in the corresponding location within a block. Figure 16 gives a pictorial representation of the different responses generated using the same refresh-pause interval and other parameters (except address) from different blocks belonging to a single DRAM module. It can be seen that the responses are not only unique across different modules but also across different blocks within a module. This goes to show the viability of D-PUF for authentication.

5.3 Robustness Analysis

The reproducibility of responses (to the same challenge) under varying operating conditions such as temperature and aging is important to PUF operation and is referred to as robustness. We quantify robustness as the average HD resulting from the comparisons between golden responses and the responses generated by D-PUF during authentication. These comparisons are referred to as *same-module* comparisons. Also, in order to demonstrate the uniqueness of responses generated from different DRAM modules, we compared the golden response generated by one module (for a particular challenge) with the corresponding responses (for the same challenge) generated by all the other modules during authentication. We refer to such comparisons as *different-module* comparisons. While *same-module* comparisons produce the true-positive (genuine authentication) rate, *different-module* comparisons give an estimate of the probable false-postive (false authentication) rate.

We performed a robustness analysis of D-PUF under varying environmental and temporal conditions and present our results below. The relative frequency refers to the fraction of total comparisons, either *same-module* or *different-module*, that yields a particular HD.

5.3.1 Robustness under temperature variations

Figure 13 shows the relative frequency versus HD corresponding to 50 different challenges, each applied to five different DRAM

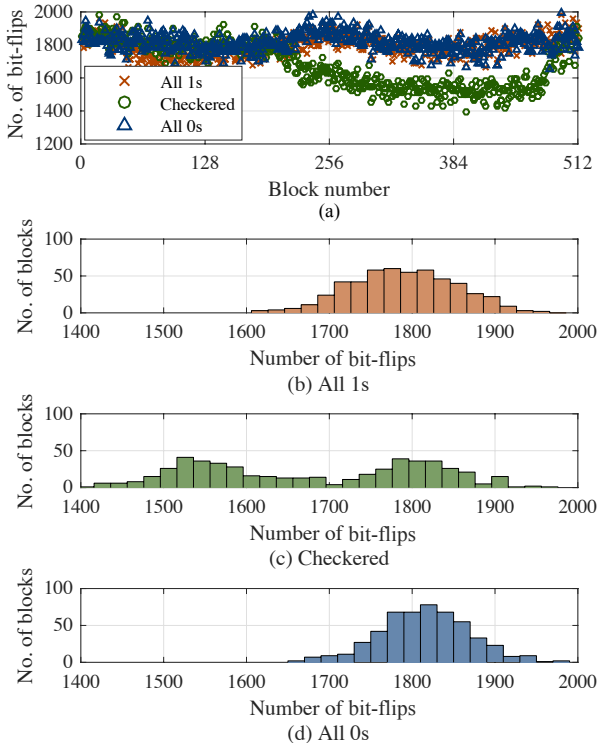

Figure 12: Variation of bit-flips with wrapper pattern

modules at three different temperatures. The golden-responses were generated at 20°C (during *enrollment* phase) while the responses for authentication were generated at 20°C, 30°C, and 40°C (during *authentication* phase). A total of 250 *same-module* comparisons and 1000 *different-module* comparisons were made altogether for the five modules. As shown, the maximum HD for *same-module* comparisons was less than 27 at 30°C but rose exponentially at 40°C. Setting an *MT* of 27 authenticated all the five modules at 30°C for every challenge. However, at 40°C this led to a number of false-negatives. Hence, our design is robust under a temperature variation of ±10°C. Note that this range also depends on the error correction algorithm utilized which happens to be (31,26) Hamming Encoder/Decoder in our case. Beyond this range, either re-enrollment needs to be performed at the new temperature or a more powerful error correction algorithm needs to be used. The HD margin between the *same-module* comparisons and the *different-module* comparisons re-emphasizes the uniqueness of the responses generated by different modules. The margin also plays an important role in setting the appropriate value for *MT* at a particular temperature, as is described below.

5.3.2 Setting match threshold (MT)

Sec. 3.2.5 defines μ, σ, and τ that are used for determining *MT*. μ and σ correspond to the temperature at which enrollment happens (20°C in our case) as shown in Figure 13(a). At 20°C, $\mu+\sigma$ was observed to be ~ 2. τ represents the maximum allowable HD between responses (golden responses and responses during authentication) due to temperature variations only and guarantees that authentication is carried out with a high true-positive rate but, more importantly, 0% false-positive rate. We consider τ to be a fraction of the minimum entropy (bit-flips) observed in a block and set it at 25% of the same. Hence, the *MT* equals 27 ($=2+25$) for a minimum entropy of 100 bit-flips observed at 20°C. This value of τ, in turn, allows us to determine the temperature range within which the CRP database generated at a particular temperature is valid. For the choice of (31,26) Hamming Encoder/Decoder, this temperature range came out to be close to ±10°C when averaged over five different modules, as depicted in Figure 13. This enabled us to use the same CRP database (generated at 20°C) for 30°C too, achieving a 100% true-positive rate while also ensuring 0% false-positive rate. However, aggressively setting the τ value for incorporating higher temperature ranges may lead to false positives during authentication, as shown in Figure 13(c), and should be avoided.

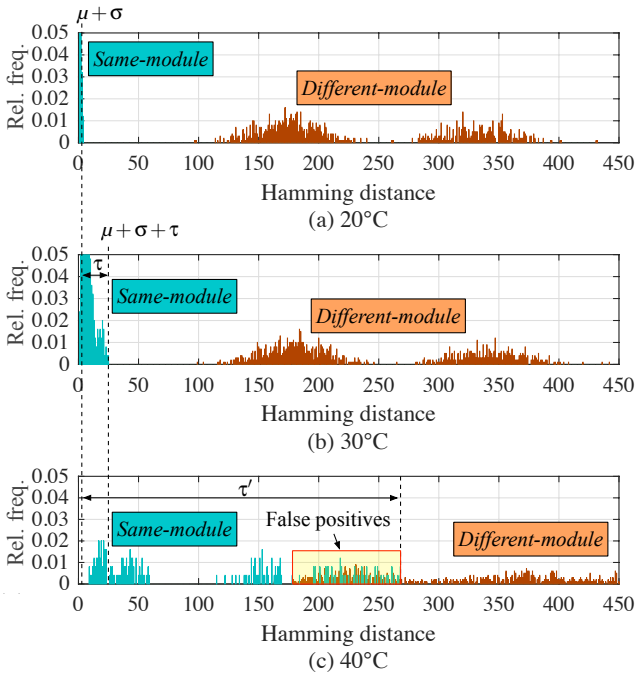

Figure 13: Robustness across different temperatures

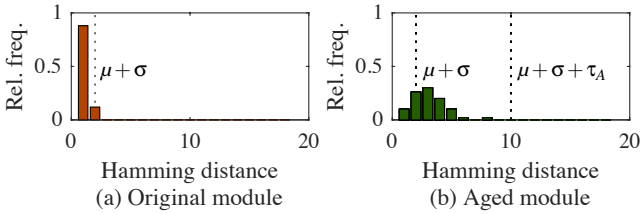

Figure 14: Robustness under aging

5.3.3 Robustness under aging

In order to approximate nine months of aging, one of the DRAM modules was subjected to a temperature of 85°C for 48 hours (according to the Arrhenius equation [21]).

Figure 14 gives the relative frequency (of *same-module* comparisons) versus HD before and after aging of a DRAM module for 50 different challenges at a constant temperature of 20°C. In an approach similar to addressing temperature variation, we also define τ with respect to aging (τ_A) and set the *MT* equal to 10. Note that the enrollment was done at 20°C before the aging process was carried out. As shown, aging seemed to generate much lesser HD as compared to temperature variations. Setting an *MT* of 10 was enough to successfully authenticate the module for every challenge.

5.4 Authentication of a sample DRAM module

In order to verify our approach, we carried out authentication of a sample DRAM module using 50 different challenges at three different temperatures. Figure 17 shows the results for the authentication test. We also used four other DRAM modules as false devices (or attacker devices) and subjected them to the same challenges. Their responses were compared against the golden responses of the sample DRAM module and the corresponding HDs are depicted as red bars in the figure. The enrollment was done at 20°C. We en-

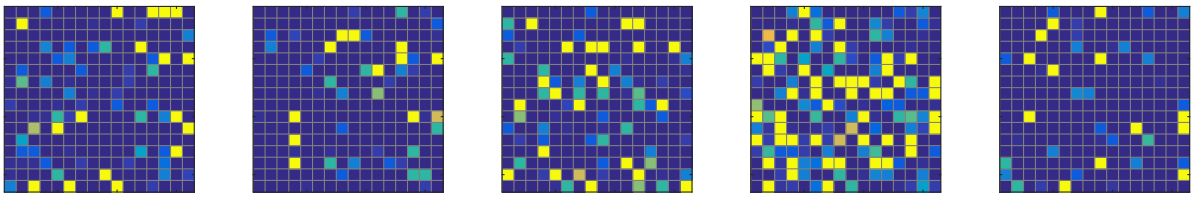

Figure 15: Unique responses generated from different DRAM modules (Address = 0x8000000, Size = 128 KB, Wrapper pattern = All '0's, Refresh-pause interval = 90 s); Average HD = 216, Minimum HD = 113

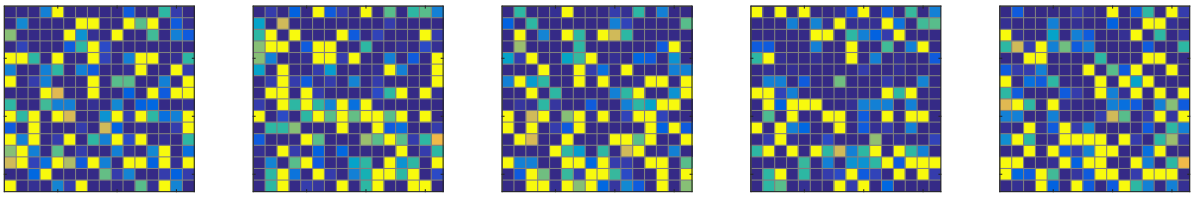

Figure 16: Unique responses generated from different blocks of DRAM Module 4 (Size = 128 KB, Wrapper pattern = All '0's, Refresh-pause interval = 60 s); Average HD = 503, Minimum HD = 486

sured a minimum entropy of 100 bit-flips again, which resulted in an *MT* of 27 and an operational temperature range of $\pm 10^{\circ}$C. As expected, we observed 100% true-positive rate at 20°C and 30°C. 40°C yielded a true-positive rate of 94%. Note that, by re-enrolling at 30°C or by employing a more powerful error correction algorithm, a 100% true-positive rate could have been achieved at 40°C. At all temperatures, the wide margin between the *MT* and the HDs from false modules guaranteed that there were no false authentications, *i.e.,* 0% false-positive rate.

6. DISCUSSIONS

6.1 Attack Scenarios and Assumptions

Before describing our attacker model, we state our assumptions for D-PUF (*P*). First, the responses generated by *P* are assumed to be unpredictable and *P* itself is unclonable. The randomness in bit-flips coupled with the reconfigurability of *P* validate this assumption. Second, the characterization results of *P* are only shared with the authenticator in a secure environment and are inaccessible to an external entity.

We envision three major types of attack that could be mounted on *P* – snooping-based, physical invasion-based, and replay attacks. The most probable type, *i.e.,* snooping-based attack, happens when an attacker learns a subset of the CRPs corresponding to a particular block and refresh-pause interval, by passively listening to the communication between *D* and *A* during the *authentication* phase. The attacker could then employ sophisticated techniques such as machine learning [7] to predict the behavior of *P*. However, the probability of success for such an attack is very low due to the following reasons. First, if a different block is used for subsequent authentications, it is near impossible for an attacker to predict the response of the block based on the response of a known one due to random yet unique bit-flips across different blocks (Figure 16). Second, if the PUF is reconfigured, it becomes even more difficult for the attacker. To explain this, we refer to Figure 10. Suppose, passive snooping allows the attacker to have complete information about the number, position, and nature of all the possible bit-flips of a certain block (128 KB size) in *P* for the 40s refresh-pause interval. The same block when used during another *authentication* phase, employing 60s interval, would generate ∼1400 new bit-flips. So, the probability of predicting the correct response is extremely low (close to $\frac{1}{128\times1024\times8 C_{1400}}$), given the limited number of times the attacker can request authentication. Third, even if the same block is continued to be used for authentication, such an attack could be prevented by generating (and using) challenges (Sec. 6.3) that utilize some minimum *unique* entropy (or bit-flips).

The second type of attack requires a highly skilled and well equipped attacker. Through physically invasive means, he/she can access the DRAM module in *D* and characterize it to learn about all the possible bit-flips in the module. However, such an attack is unlikely due to the following reasons. First, the attacker needs to possess the DRAM module for a sufficiently long period of time to be able to characterize it exhaustively. In the worst-case scenario, he may be able to characterize a portion of the DRAM for a few values of the parameters. But, the PUF can be easily reconfigured preventing the attacker from predicting the DRAM behavior completely. Second, though the attacker may extract the helper data (usually stored in the public NVM of *D*) that leaks some information about the expected response, it is unusable as each challenge (and the corresponding response and helper data) is used only once during authentication.

Finally, replay attacks cannot be mounted on a PUF as a CRP is never used twice during authentication.

6.2 Mitigation of Authentication Latency

Sometimes D-PUF may not be able to generate the response for a challenge instantaneously due to a high refresh-pause interval or non-availability of a sub-address space (due to the sharing of DRAM among multiple applications). One potential way of mitigating this latency is by piggybacking future challenges with the current one and caching the corresponding responses within *D*, which could be then sent during the next authentication cycle.

6.3 Generation of Challenge Bit-Stream

The randomly generated challenge may contain a bit-stream such that none of the bits flip during response generation. For example, a memory block containing only *true*-cells ('1' → '0' bit-flips), when subjected to a challenge bit-stream containing all 0s, would generate a response same as the challenge itself. To prevent this, the challenge bit-stream can be constructed with the help of the characterization results obtained prior to enrollment. By putting `'1'` and `'0'` at the *true*-cell and *anti*-cell positions respectively in a challenge bit-stream, the inherent entropy of the module can be properly utilized.

6.4 Temperature-Specific CRP Database

In order to ensure robust authentication across varying tempera-

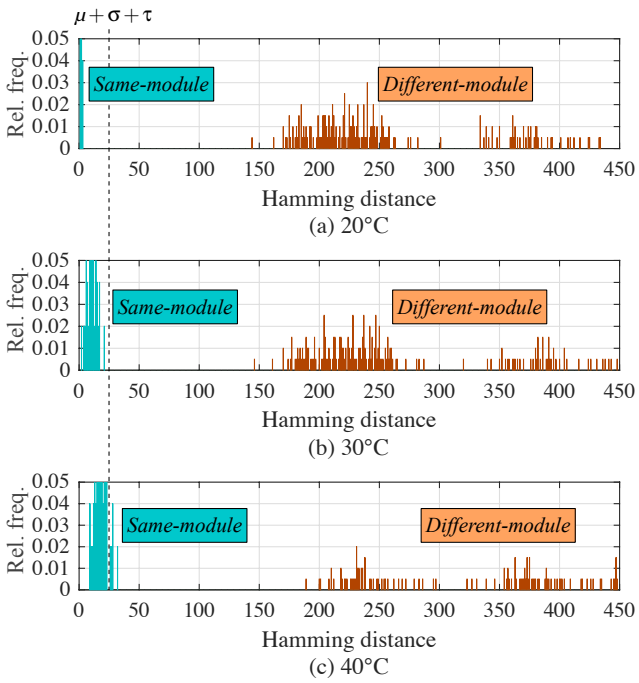

Figure 17: Authentication of a sample DRAM module (Mod. 6)

tures, temperature-specific CRP databases could be utilized in conjunction with the *match threshold* technique. During authentication, the device can send its operating temperature as a part of the authentication request. The authenticator can accordingly select the challenge from the appropriate CRP database.

7. RELATED WORK

One of the earliest works on PUFs was carried out in Ref. [22], resulting in an optical PUF based on the scatter pattern of a laser beam. Ref. [2] introduced the concept of silicon PUFs and provided various circuit realizations that could be integrated into an electronic circuit. Due to this ease of integration, silicon PUFs have become extremely popular in present day implementations. Some of the notable works on silicon PUFs are mentioned next.

Ring Oscillator and Arbiter PUFs [23] exploit the inherent delay characteristics in IC components for authentication and generation of secret keys. However, both these implementations require dedicated circuitry that is added solely for the PUF operation and present an area overhead. Memory-based PUFs, on the other hand, utilize the memory module already present on the IC/SoC and do not require any dedicated circuitry. For example, Ref. [5] used the start-up state of an SRAM module to identify its fingerprint and generate true random numbers. The start-up state approach has also been used to derive a fingerprint out of a DRAM module, as presented in Ref. [4]. However, both these approaches have an inherent disadvantage, *i.e.,* they require power cycling of the memory module for generating the responses, which may hinder the continuous execution of applications.

7.1 DRAM PUFs

Reduction of the write-duty cycle in a DRAM module is employed in Ref. [6] to implement a strong PUF. However, the reduction is achieved by adding a delay generator to the write-circuitry of the DRAM module. This not only requires very precise control over the *write* signal but is also not applicable to off-the-shelf DRAM modules. Ref. [3] utilizes refresh pausing to generate unique identifiers and random numbers from a DRAM module. However, the work employs a constant refresh-pause interval of 256 - 8192 s that may be considered too slow for authentication.

7.2 Reconfigurable PUFs

A parallel approach to the development of strong PUFs has focused on reconfiguration. Reconfigurable PUFs (rPUFs) [14, 15] have a mechanism to transform themselves, generating a new and unpredictable challenge-response behavior. Ref. [10] implements a *logically* reconfigurable SRAM PUF for secure key storage by hashing the start-up state of the SRAM with a stored bit-stream, referred to as the logical *state*.

We proposed the design of an *intrinsically reconfigurable* strong PUF based on refresh pausing in DRAM. Unlike Ref. [10], our design enables reconfiguration without the requirement of any additional hardware resource (*e.g.*, private NVM) and hence, can be applied to any commercially available DRAM module (in contrast to Ref. [6]). It also alleviates the problem of a large refresh-pause interval which was required in Ref. [3]. The refresh pausing approach also ensures that there is no need for power cycling, unlike Ref. [4].

8. CONCLUSION

In this paper, we proposed an *intrinsically reconfigurable* DRAM PUF based on refresh pausing and also presented a secure, low-overhead methodology that uses the PUF for device authentication. We validated our work on a real system using off-the-shelf DRAM modules and evaluated it thoroughly. The overall design performed robustly under various environmental and temporal variations and achieved a `4.3X-6.4X` reduction in authentication time compared to prior work. We envision that our work will pave the way for the wide adoption of DRAM PUFs into a large number of modern embedded devices.

9. REFERENCES

[1] U. Guin *et al.*, "Counterfeit integrated circuits: A rising threat in the global semiconductor supply chain," *Proc. of the IEEE*, 2014.

[2] B. Gassend *et al.*, "Silicon physical random functions," *CCS*, 2002.

[3] C. Keller *et al.*, "Dynamic memory-based physically unclonable function for the generation of unique identifiers and true random numbers," *ISCAS*, 2014.

[4] F. Tehranipoor *et al.*, "DRAM based intrinsic physical unclonable functions for system level security," *GLSVLSI*, 2015.

[5] D. Holcomb *et al.*, "Power-up SRAM state as an identifying fingerprint and source of true random numbers," *IEEE Trans. on Computers*, 2009.

[6] M. Hashemian *et al.*, "A robust authentication methodology using physically unclonable functions in DRAM arrays," *DATE*, 2015.

[7] U. Rührmair *et al.*, "Modeling Attacks on Physical Unclonable Functions," *CCS*, 2010.

[8] C. Herder *et al.*, "Physical unclonable functions and applications: A tutorial," *Proc. of the IEEE*, vol. 102, 2014.

[9] H. Jayakumar *et al.*, "Energy-efficient system design for IoT devices," *ASP-DAC*, 2016.

[10] I. Eichhorn *et al.*, "Logically reconfigurable PUFs: Memory-based secure key storage," *STC*, 2011.

[11] P. Nair *et al.*, "A case for refresh pausing in DRAM memory systems," *HPCA*, 2013.

[12] A. Raha *et al.*, "Quality-aware data allocation in approximate DRAM," *CASES*, 2015.

[13] J. Liu *et al.*, "An experimental study of data retention behavior in modern DRAM devices: Implications for retention time profiling mechanisms," *ISCA*, 2013.

[14] D. Lim, "Extracting secret keys from integrated circuits," *Master's thesis, MIT*, 2004.

[15] K. Kursawe *et al.*, "Reconfigurable physical unclonable functions - enabling technology for tamper-resistant storage," *HOST*, 2009.

[16] W. Che *et al.*, "PUF-Based Authentication," *ICCAD*, 2015.

[17] U. Kocabaş *et al.*, "Converse PUF-based authentication," *TRUST*, 2012.

[18] Terasic, "TR4 FPGA development kit," March 2015. [Online]. Available: http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&No=683

[19] Altera, "Nios II processor," March 2015. [Online]. Available: https://www.altera.com/products/processors/overview.html

[20] S. Liu *et al.*, "Flikker: Saving DRAM refresh-power through critical data partitioning," *ASPLOS*, 2011.

[21] JEDEC, "Acceleration factor, temperature." [Online]. Available: https://www.jedec.org/standards-documents/dictionary/terms/acceleration-factor-temperature

[22] R. Pappu *et al.*, "Physical one-way functions," *Science*, 2002.

[23] G. E. Suh *et al.*, "Physical unclonable functions for device authentication and secret key generation," *DAC*, 2007.