Fast DRAM PUFs on Commodity Devices

Jack Miskelly and Máire O'Neill, *Senior Member, IEEE*

***Abstract*—Intrinsic physical unclonable functions (PUFs), which derive hardware identifiers from components already present in a system without modification, are an appealing way to add a layer of hardware rooted security into a system. This is evidenced by the fact that the majority of PUF designs in commercial use today are intrinsic. However, as each intrinsic PUF design is reliant on specific hardware their use is limited to a subset of systems. It is therefore desirable to have practical intrinsic PUF designs for as wide a range of underlying hardware as possible. Most intrinsic PUF designs to date have used memory as the entropy source, with the most well studied type being based on SRAM. More recently designs based on DRAM have been proposed, an appealing prospect considering the ubiquity of that technology. While previous research has demonstrated that entropy can be extracted from DRAM there has not yet been a substantive demonstration of such a PUF operating in real-time on a commodity system. In this article, we present a novel set of algorithms for deriving PUF responses in-runtime from DRAM by altering timing parameters using only software. These algorithms reduce the critical period of system disruption by 96% from 88 ms to 3 ms on average compared to existing designs. We present a large scale dataset derived from 1824 DRAM chips characterized using the proposed design on commodity off-the-shelf desktop hardware running a Linux OS. An analysis of the data shows that in addition to the speed improvements the proposed design shows near ideal (>44%) uniqueness and good (>88%) reliability.**

***Index Terms*—DRAM physical unclonable function (PUF), hardware derived identifier, hardware security, intrinsic PUF, memory PUF, PUF.**

I. Introduction

PHYSICAL unclonable functions (PUFs) are a form of hardware-based security primitive which derive entropy from low level physical variation in components. A PUF typically consists of a circuit that can accept some set of input challenges and derive an output response based on the challenge and entropy extracted from the low level hardware variances in the circuit components. The PUF response acts as a unique "fingerprint" for the circuit as the low-level variations that dictate the response cannot be selected without access to a process orders of magnitude more precise than that used to produce the PUF itself.

Despite close to two decades of research on PUFs there has been minimal adoption of the discrete circuit PUFs which comprise the majority of research to date. In part, this is due to the difficulty in implementation. PUFs are often touted as being well suited to resource constrained device and in particular IoT devices but introducing new and unfamiliar hardware components into existing designs requires a cost in development resources that on balance does not seem to appeal to many device manufacturers.

It is perhaps because of this that the most widely used subcategory of PUFs at present are Intrinsic PUFs. These PUFs do not use a dedicated circuit to produce the PUF response but rather extract the entropy from components in circuitry already present in the target system. In many senses, this is ideal as it avoids the overheads of discrete circuit PUFs with many Intrinsic PUFs able to be implemented entirely through software on existing devices. However, such PUFs are restricted to devices that contain the necessary base hardware. Due to this, the development of practical Intrinsic PUF designs for as broad a range of base hardware as possible is vital.

A. PUF Fundamentals

The underlying entropy sources for a given PUF design vary substantially. The variance in almost any component property can be used as the entropy source with the right extraction mechanism. For example, in Arbiter PUFs [6], two nominally identical sets of delay stages are used, with an arbiter at the end outputting a 1 or 0 depending on whether the upper or lower path executes faster, which is determined by low-level variance in the delay elements themselves. Similarly, ring-oscillator (RO) PUFs [3], [4] use loops of delay elements to generate frequencies determined by the delay component variances. Comparing two of these frequencies generates a single bit of the response depending on which frequency is higher. Other entropy sources include the startup state of SRAM cells [1], the point-of-failure frequency in overclocked processor cores [5], and the stable state of cross coupled latch cells on FPGA [7].

While the specifics of the hardware implementations are highly varied between PUF designs, at a higher level all PUFs can be conceptualised as a some kind of physically rooted function which cannot be easily cloned. It is sufficient to understand that the PUF function accepts some set of challenges as input and produces a corresponding set of responses as output. Any instance of a PUF should produce a unique response set to the set of challenges. This property is called uniqueness and is derived from the interdevice Hamming distance. However, within a given PUF instance the same challenge should consistently produce the same response under repeated measurements. This property is called reliability and is calculated from the Hamming distance between repeated

Manuscript received April 17, 2020; revised June 17, 2020; accepted July 6, 2020. Date of publication October 2, 2020; date of current version October 27, 2020. This article was presented in the International Conference on Embedded Software 2020 and appears as part of the ESWEEK-TCAD special issue. *(Corresponding author: Jack Miskelly.)*

The authors are with the Centre for Secure Information Technologies, Queen's University Belfast, Belfast BT7 1NN, U.K. (e-mail: jmiskelly08@qub.ac.uk; maire.oneill@qub.ac.uk).

Digital Object Identifier 10.1109/TCAD.2020.3012218

measurements of the same PUF. Ideally, a PUF should also show minimal bias toward binary 0 or 1 in the response set.

PUFs in practice are most commonly used in two applications. The first and simpler is as a source of secret information for use in cryptographic algorithms. In this case, the PUF, which may have only a single valid challenge and response, provides some secret hardware derived value when prompted which is fed into a standard cryptographic function. In effect, the PUF serves as an alternative to storing the secret information in ROM or similar techniques.

The second use case is for device ID and authentication. In this case, it is generally assumed that this process is centrally managed in some way, with a central device controlling the authentication of many other devices. This requires the enrolment of each PUF instance on creation. Enrolment involves the characterization of the full set of challenges and responses, such that the controlling device has a record of the expected response of each PUF instance to each possible challenge. When a device needs to be authenticated one of this set of challenges is selected and sent to that device, which passes it to the PUF circuit and returns the response. If the response matches that derived at enrolment then the identity of the target device is verified. This challenge-response verification process is referred to as a PUF query.

B. PUF Classification

When discussing PUFs it is often useful to classify them as "Weak" or "Strong." This is not a measure of the attack resistance of a PUF but is rather based on the size of the challenge-response space—that is, how many unique challenge-response pairs (CRPs) that are possible for a given PUF design. Weak PUFs have only a single or small amount of CRPs, while strong PUFs have a large CR space with the number of CRPs ideally increasing exponentially as the number of challenge bits is increased.

Strong PUFs, due to the large set of CRPs, can in theory be used for device authentication without additional cryptographic hardware. However, they are also vulnerable to certain attacks, particularly machine learning (ML)-based attacks. Many of the most well known PUF designs, such as the RO PUF [3], [4] or the Arbiter PUF [6] are strong PUFs. While there have been substantial efforts to mitigate the issues with strong PUFs they have yet to see adoption in the industry on the level of weak PUFs.

Weak PUFs are more limited in application but less vulnerable to ML attacks due to the small CRP space not allowing an attacker to build a large enough training dataset. Weak PUFs are mainly used for the generation of small amounts of secret information, as True Random Number Generators, and to generate hardware derived identifiers. Weak PUFs have seen substantial adoption in industry, particularly the SRAM PUF [1], a variant of which is deployed in many commercial FPGA accelerator cards.

C. Intrinsic PUFs

Intrinsic PUFs are a subset of PUF designs that form the PUF response by extracting entropy from hardware already present in a system. Truly intrinsic PUFs require no modification or addition of hardware whatsoever instead using existing software and hardware functions to extract the entropy from the underlying hardware. This has obvious benefits as it can add a layer of hardware rooted security into a system while requiring no additional hardware resources. Intrinsic PUFs extract additional security functionality from components that will be used in the system regardless. The main drawback to Intrinsic PUFs is that by their very nature they can only be implemented in systems that have specific hardware already present. While it is entirely possible to add hardware to allow for an Intrinsic PUF, this negates most of the benefits of these PUFs over discrete circuit PUFs.

The most well studied intrinsic PUF design is the SRAM PUF [1]. When powered on SRAM cells enter into an unstable state which will fall into either a 1 or 0, with the bias of a cell toward either value determined by variances in the cell components. The PUF response is derived from the set of power-on values of the memory cells. In the most basic form this bit pattern is simply used unaltered as the response with more advanced methods also being used to improve the uniqueness and reliability of the PUF response. As the single PUF challenge in this case is to power cycle the memory module the viable use cases of this design are limited. Designs for memory intrinsic PUFs based on other memory technologies, such as DRAM have also been proposed. These are discussed in greater detail in Section II. As with the SRAM PUF specific conditions are required for the PUF to be viable in a given system.

D. Contributions and Outline

In this article, we introduce a novel sets of algorithms for extracting PUF responses from commodity-off-the-shelf (COTS) DRAM memory modules by leveraging internal memory timing parameters. These algorithms provide a purely software-based in-runtime intrinsic DRAM PUF. Further, they improve upon existing comparable designs by greatly lowering the system disruption required to query the PUF. We demonstrate the proposed design on unaltered COTS computing systems running a live Linux OS. From this, we present a dataset based on 1820 discrete DRAM chips, which is the largest of its kind to date to the best knowledge of the authors at the time of writing. This dataset is provided freely for the use of the research community. Finally, we use this dataset to analyse the quality of the proposed PUF design, demonstrating that it exhibits near ideal uniqueness and good reliability. Further, we demonstrate that at the cost of increased memory overheads the reliability can be improved to a near ideal value. The contributions of this article can be summarized as follows.

1) A novel implementation of the latency-based DRAM PUF concept with a new set of algorithms for PUF enrolment and query which reduce the critical period of system disruption by up to 96% to below 31 ms in the worst case and 3 ms on average.
2) A demonstration of the proposed design on live desktop systems, using only COTS hardware and a widely

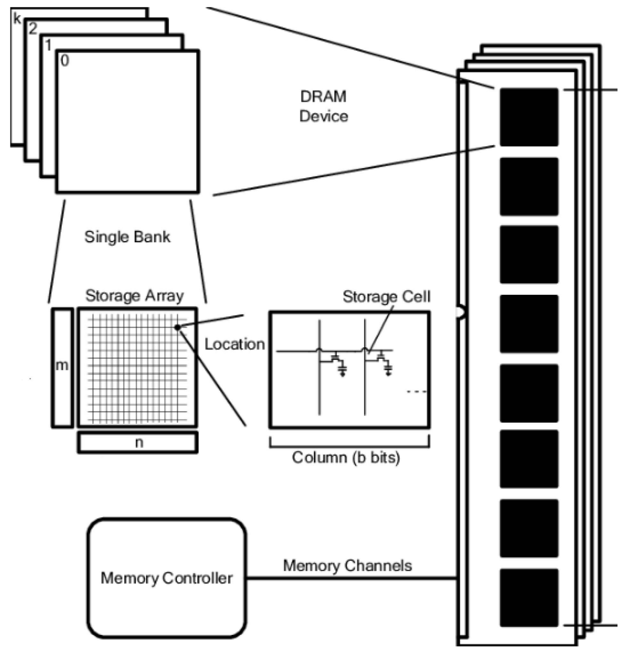

Fig. 1. DRAM structure for a standard DIMM [8].

used OS (Ubuntu Linux). Results from six individual machines are presented.

3) A large scale dataset using a measurements from over 1800 COTS DRAM chips from three manufacturers and in two form factors [dual in-line memory module (DIMM) and small outline DIMM (SODIMM)]. This is by a substantive amount the largest dataset of its kind to date and is provided for the use of the research community.
4) An analysis of the performance of the proposed design taking into account the standard metrics of uniqueness and reliability, the bias of individual responses, the required query time, and the overheads required to implement the proposed design.

II. Background and Limitations of Existing DRAM PUFs

A. DRAM Structure

The fundamental storage medium of DRAM is an array of capacitive cells, gated by transistors with each cell storing a single bit of data. The data bit stored depends on the charge of the capacitor—in general a charged capacitor equates to a binary 1 and a discharged capacitor to a binary 0. This is a simplification of the hardware level operations, where in practice there exist both cells and anti-cells which are inverted in terms of what level of charge equates to each value.

These cells are arranged into a grid of rows and columns referred to as a bank. There will usually be many such banks in a single DRAM chip. Often, multiple chips are combined into a rank which share a single chip select. While individual chips are deployed the most recognizable form factor for DRAM is the DIMM which incorporates multiple chips. Another common form factor is the SODIMM which is used in smaller form factor devices.

B. DRAM Volatility and the Refresh Cycle

DRAM cells are inherently volatile. Over time due to leakages the value stored in a cell will decay and eventually a bit flip may occur depending on the stable state of that particular cell. In order to retain the memory values the entire memory must be refreshed periodically. This is done by reading the value in each cell and writing it back. In modern DRAM this is controlled and carried out by dedicated refresh cycle circuitry.

C. DRAM Operations and Timings

Due to the multiple steps involved in internal memory operations and the inherent latency of some of these steps certain delays are introduced to ensure correct functionality. These delays are dictated by timing parameters imposed by the memory controller. The safe range for each parameter is specified by the device manufacturer. In most systems these parameters can be changed at the BIOS level, and can be adjusted to improve either the performance or the stability of the memory in a system. If timings are lowered past the lower limit given by the manufacturer the stability of memory operations can no longer be guaranteed resulting in undefined behavior, though depending on the exact circumstances the system may still be able to function with "unsafe" values for some timings.

A timing value that is critical in this work is *tRCD*. This is the required delay between the row address strobe (RAS) signal which selects a row to perform the operation on, and the column address strobe (CAS) signal which selects a column within the row. When *tRCD* is given a value below the recommended limit the memory enters an undefined state in which read operations return incorrect values for some or all of the bits being read. This property is discussed in greater detail in later sections.

D. DRAM-Based PUFs

As a ubiquitous technology found in a wide range of devices the idea of using DRAM as the core hardware for an Intrinsic PUF has been explored in several works. While some designs exploiting more obscure properties of DRAM have been proposed, such as the Rowhammer Effect PUF [10] the majority of DRAM PUF designs proposed to date fall into one of two categories: 1) the more well studied Retention PUFs which use the refresh cycle to extract entropy and 2) the recently proposed Latency PUFs which use the latency of operations to do so.

E. DRAM Retention PUFs

The DRAM Retention PUF proposed by Tehranipoor *et al.* [2] exploited the volatile nature of DRAM cells to extract entropy based on either the drainage rate or the stability of cells. Nominally the charge of a cell containing a 1 and the rate at which that cell discharges should be fixed. In practice this is not the case as due to process variation in the various components the real values vary within an acceptable range centred around the nominal value. This variance is marginal and hence does not impact the normal functions of the memory.

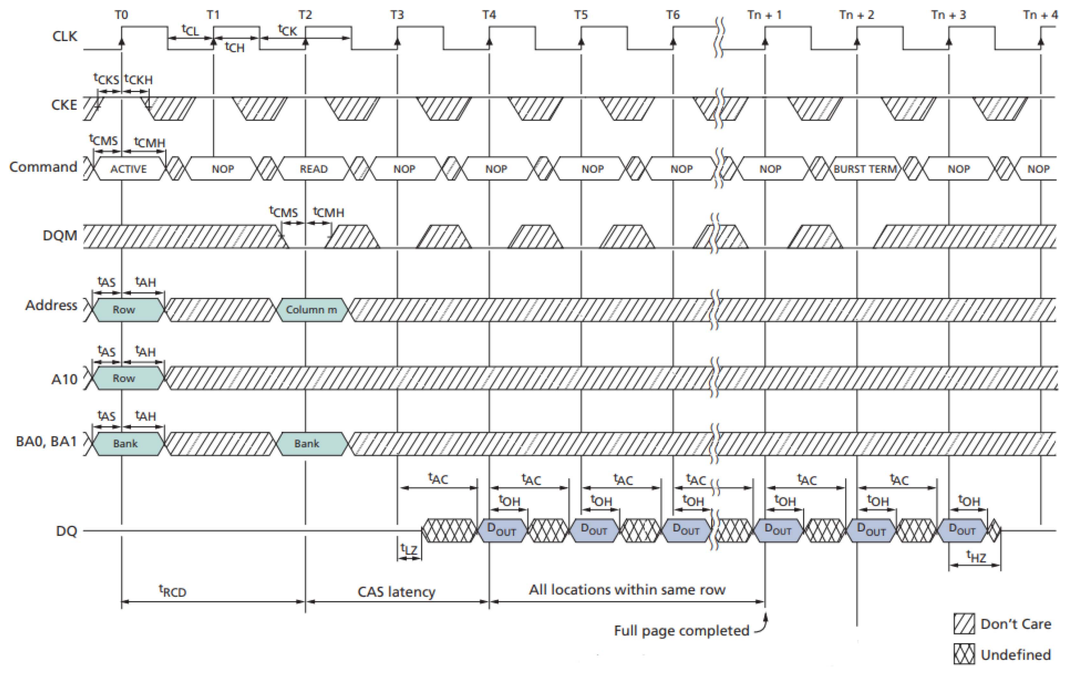

Fig. 2. Diagram showing the timings used in a burst read operation [9].

The Retention PUF exploits this by filling a segment of DRAM with a bit pattern and then disabling the refresh cycle of the memory. If the components all had the exact nominal values this would result in a period of normal operation until a fixed period had passed, at which point the charge leakage would be enough that all the bits would flip. In practice, however, each cell leaks at a slightly different rate with some cells flipping after only a short period and others retaining their value for much longer. Further, the distribution of the fast flipping cells and the slow flipping cells is highly random across commercial DRAM giving this type of PUF high uniqueness. By stopping the refresh cycle and reading the memory back after a fixed period a bit pattern unique to that memory segment is generated. In a following work it was also shown that a PUF of this type is fairly resistant to degradation due to ageing [12].

The initial proof-of-concept Retention PUF designs suffered from the requirement for a lengthy period in the order of minutes to allow for the decay of cells before the response could be read. In addition, it was unclear if the PUF response could be generated in-runtime on a real system without modification (i.e., as a truly intrinsic PUF). Sutar *et al.* later proposed means by which the query time could be reduced to between 20 and 60 s [13] while Schaler, Xiong *et al.* [14] and Schaller *et al.* [15] demonstrated the viability of using a Retention PUF on a commodity system, either by generating the response at boot or by generating the response in-runtime while manually refreshing key parts of memory to prevent a system crash.

Despite the advances from the original proposal in terms of practical implementation and query time, there are still serious drawbacks to the Retention PUF, either the response must be generated at boot (and may not be regenerated during runtime) or the response can be generated in-runtime but with substantial limitations on the system for the query duration due to the need to reserve sections of memory and the computational cost of manually refreshing key areas of memory to prevent instability. To date in most designs this period is in the order of several minutes and in all designs it is at minimum 20 s.

F. DRAM Latency PUFs

As a response to the technical challenges inherent to DRAM Retention PUFs Kim *et al.* [11] proposed the DRAM Latency PUF. Rather than using cell decay as the entropy source this design exploits the latency-stability tradeoff of modern DRAM to extract entropy by placing the memory into a state of undefined behavior through the manipulation of timing parameters. By lowering specific timing parameters, such as *tRCD*, beyond the normal lower bound the memory is placed into a state where read operations do not return the values held in memory but instead return a pattern derived from both the memory values and the low-level variances in the DRAM circuitry. The resultant error patterns are repeatable within a given piece of hardware and form the PUF response.

This PUF has several advantages over the Retention PUF. Like some of the more recent Retention PUF proposals it is in theory viable for use in-runtime so long as the system allows in-runtime changing of timing parameters. This means it can be used for authentication within a standard challenge-response framework. Like the Retention PUF it exhibits high uniqueness. Most importantly it requires only a relatively short query time in the order of 88 ms to generate an 8 KiB response, a substantial improvement over the Retention PUF.

In this work, we propose a novel PUF based on the latency PUF concept which works to further minimize the query time. We will demonstrate that by the careful selection of bit patterns and a more efficient query scheme that the key period of system disruption can be reduced to just 3 ms while retaining excellent uniqueness and high reliability.

III. Proposed PUF Design

In this section, we present a set of novel algorithms that allow for the extraction of PUF responses from COTS DRAM memory by exploiting the behavior of memory with extremely reduced timing parameter values. This new approach leads to drastic reductions in the critical period during which memory must be reserved for PUF use thus minimizing system disruption while maintaining near ideal performance in other metrics.

The proposed design is a Weak PUF. The design can produce a CRP set equal in size to twice the available memory size. While in many cases this will be relatively large it increases only linearly with PUF segment size. As such the operating assumption is that the proposed PUF will be used as a Weak PUF to generate hardware rooted identifiers. The enrolment and query algorithms and the threat model have been chosen with this in mind.

A. Challenge-Response Mechanism

In the proposed design the PUF challenge consists of a set of values to set each timing parameter to combined with a series of offsets used to target sections of the PUF segment. By default, the timing parameter changes are simply to set *tRCD* to the lowest possible value which is sufficient to extract entropy from the hardware. Supplementary to this is the start address of the memory segment reserved for the PUF and a bit-pattern which will be written into it prior to query. As this data is static and does not reveal any information about the PUF response to an adversary without root access it can be stored on the target device in order to reduce the size of the challenge, or provided as part of the challenge for additional security. The PUF response consists of a number of bits specified in the challenge returned as a series of 32 b segments. These segments are concatenated without further post-processing to produce the final response.

The enrolment process and challenge-response mechanism used have been designed with the aim of minimizing the query time (and hence, the amount of system disruption) while maximizing the entropy of the PUF. The reasoning for how the challenge values, memory segment, and bit-pattern are selected is detailed in the following sections.

B. Enrolment

The bit pattern held in memory when the PUF is queried has a significant impact on the generated response. In previous works fixed bit patterns have been used in which the memory segment is filled with all 1's, all 0's, or some other regular pattern, such as alternating 1's and 0's. This approach is relatively simple but does not extract the maximum available entropy from each memory cell, as we observed that the responses generated using the same memory segment but inverted bit patterns were distinct and could exhibit markedly different characteristics in terms of the probability of bit flip and the reliability of bit flips. This observation and how it was reached is discussed in further detail in Section V.

As such the key goal and contribution of the novel enrolment algorithms presented here is in the derivation of nonregular mixed bit patterns which optimize the PUF response in terms of reliability such that fewer repeated queries are needed to acquire error-corrected responses, with a corresponding improvement in the critical period of system disruption when querying the PUF. This new approach also has the advantage of exhibiting similar near ideal uniqueness values to comparable works, despite optimizing for reliability and speed rather than other metrics.

1) Changes to the Method of Entropy Extraction: Unlike the approach in [11] where a count of the number of read failures is used to determine the PUF response bits, here the output of the read operations is used directly as the PUF response and all error correction and post-processing is performed after the query has finished. While the end result is fairly similar, our approach allows for a further reduction in the amount of system disruption by performing any processing on the data after reserved memory has been released and normal operation restored. In Section V—PUF analysis it will be shown that this approach can produce a sufficiently robust PUF with the only tradeoff being an increase to memory overheads.

2) Initial Characterisation: The enrolment process starts with characterizing the response of the PUF segment for a bit pattern of all 1's and also for an all 0 b pattern over a large number of repeated measurements, *n*. The result of this will be a set of *n* responses for each solid bit pattern.

3) Granularity of Characterisation: It may seem intuitive that in deriving a mixed it pattern that the responses should be analysed on a bitwise basis. We found that this approach while certainly possible produced no better results in practice than analysing each 32-b subsegment while requiring a substantially more complex enrolment process. Further, if this bitwise derivation is used, then a bit pattern equal in size to the PUF memory segment must be stored on device, or transmitted as part of the PUF challenge. As such the 32-b response from each single read operation is taken as a discrete unit and kept or discarded as a whole. This means the mixed bit pattern will consist of alternating blocks of 32-b solid patterns of 1 or 0 and the mixed pattern can be stored using only 1-b per 32-b of the output response.

4) Calculating the Reliability of Subsegments: The key metric to select for in the mixed bit pattern is reliability as this allows for minimal repetition during query and corresponding reductions in query time. In addition, the reliability of subsegments is sufficiently randomly distributed that selecting for reliability does not lower the uniqueness of PUF responses significantly from what can be achieved using a regularly alternating pattern of 0 and 1. This is detailed further in Section V. To calculate reliability for each subsegment first simple error correction is performed on the *n* repeat measurements to derive a reference value. Reliability for each subsegment is then calculated in reference to this value using the equation in Section V-B. This operation is performed on every characterized sub segment for both solid bit patterns.

5) Filtering of Zero Entropy Responses: We observed that in certain devices there would be segments of memory that produce no bit-flips under any of the possible PUF setups for one or both of the inputs. Including these segments would lower the entropy of the response. Further, if selecting for reliability a disproportionate number of these segments will be included as they will be near or at 100% reliability. In cases where one input produces bit flips and the other does not one value can be discarded immediately and the reliability calculation skipped for that segment. If neither input produces any bit flips then the location must be flagged and excluded from the final CRP set. Again this improves the quality of the response at the expense of further memory overheads.

6) Deriving the Optimal Bit Pattern: For all subsegments which are not yet determined the value of that 32 b subsegment in the mixed bit pattern is simply the value of whichever bit pattern produced the more reliable response. Thus, the final pattern should consist of only subsegments in which at least one pattern exhibited entropy. If only one pattern exhibited entropy the value of the subsegment in the mixed pattern is whichever solid pattern produced entropy. If both solid patterns exhibited entropy then the value in the mixed bit pattern subsegment is whichever solid pattern produced the most reliable response.

Optionally a further step can be taken to filter out unreliable responses by repeating some of the previous step using the derived bit pattern. This allows a certain degree of control over the reliability of the overall CRP set but at the expense of lowering the size of the set. This can be compensated for by increasing the PUF segment size if the increased overheads can be tolerated. In Section V—PUF analysis it is shown that it is possible to use this technique to produce a PUF with a minimum reliability of 99% for all CRPs.

Through burst reads of memory offsets found to exhibit high reliability and the use of a derived bit pattern which further enhances reliability the throughput of the PUF is maximized and the critical period of system disruption minimized. The result is a PUF which has a greatly reduced query time while maintaining high performance on key metrics.

Algorithm 1: Characterise PUF for *0* and *1*

```
Reserve Memory Channel B;
Reserve Segment of size (PUF Segment)*2 in Memory
  Channel A;
Fill PUF segment S in B with 0xFFFFFFFF;
Set timings to LOW;
for <large number of reads> do
    while NOT end of segment do
        Read 32 bit double word from next location;
        Store value in array in Memory Channel A;
    end
end
Restore original timings;
Fill PUF segment S in B with 0x00000000;
Set timings to LOW;
for <large number of reads> do
    while NOT end of segment do
        Read 32 bit double word from next location;
        Store value in array in Memory Channel A;
    end
end
Restore original timings;
```

Algorithm 2: Generate Optimal Bit Pattern

```
for <all segments> do
    if bit flips in both responses then
        Value = input of
          max{rel(0x00000000),rel(0xFFFFFFFF)};
    else
        if bit flips in one response then
            Value = input of bit-flip response;
        else
            Exclude segment from CRP set;
        end
    end
end
```

C. Query

Much of the complexity in the operation of the proposed design is in the initial enrolment phase. Once the PUF has been enrolled the PUF query that will be performed on the target system is much simpler and lightweight. Once the bit pattern and map of memory locations is enrolled to query the PUF all that is needed is to reserve the PUF memory segment, write the known bit-pattern into the segment, lower the timing parameters to the specified values, and read the memory location at each specified offset. As the timing parameters associated with write operations remain unchanged the PUF response can be stored in any part of memory, including the same area of memory as the PUF segment if desired.

D. Selection of Parameter Values

The selection of the timing values is a key factor in getting a useful CRP set from the proposed design. In all cases, the *tRCD* timing parameter must be lowered enough to initiate an unstable state. The point at which this state is initiated is heavily dependant on the specifics of the memory controller and the DRAM itself. In cases where this point is unknown or uncertain it is viable to simply select the lowest possible value for *tRCD*. Other timing parameters may be lowered in conjunction with *tRCD* to varying effect but the exact results of this and the means of selecting these parameters is outside the scope of this work.

During the enrolment stage the number of repeated measurements used to characterize the PUF can be varied, with smaller numbers of repetitions speeding up the enrolment process but higher numbers providing more accurate values for reliability. In the experiments used in this work 100 repetitions were used. Likewise, when querying the PUF the number of repetitions presents a similar tradeoff between query speed and the accuracy of the error corrected response. In all

Algorithm 3: Filter CRP Set

```
Reserve Memory Channel $B$;
Reserve Segment of size (PUF Segment)*2 in Memory
  Channel $A$;
Fill PUF segment $S$ in $B$ with derived bit pattern;
Set timings to $LOW$;
for <large number of reads> do
    while NOT end of segment do
        Generate PUF response;
    end
end
Restore original timings;
for <all response segments> do
    Calculate reliability;
    if reliability is above threshold $T$ then
        Include segment in final CRP set;
    else
        Exclude segment from final CRP set;
    end
end
```

measurements taken in this work ten repeated measurements were used.

IV. Experimental Setup and Methodology

The following section details the hardware and process used to derive the dataset used in the analysis in Section V.

A. Testing Platforms

The platforms used to perform the experiments consisted of a set of three COTS desktop systems for testing DRAM in the DIMM form factor and a further three laptop systems for testing SODIMM form factor memory. Each device in the testbed contained an AMD A-series processor and was running a live instance of Ubuntu Linux 16.04.

The PUF itself was implemented as a set of Linux drivers which could be activated to perform individual queries or to perform a full enrolment of the PUF. As all test devices contained related models of AMD processors using the same memory controller architecture this driver was able to be ported to each device with only minimal changes. A more detailed description of the technical challenges of implementing the PUF is provided in Section V.

The datasets were taken from a set of 1824 discrete DRAM chips from three different manufacturers and in two form factors. Of these 120 chips were from Apacer DRAM in SODIMM form factor, 904 were from QUMOX DRAM in DIMM form factor, and the remaining 800 were from Kingston DRAM in DIMM form factor.

B. Experimental Methodology and Dataset

When extracting data from the test devices the following methodology was used. From each DRAM chip a representative 4 KB key was generated for the solid bit patterns 0xFFFFFFFF and 0x00000000, respectively. This was taken from a contiguous 4KB segment with the same starting offset relative to each chip. This segment was queried under eleven parameter sets and repeated ten times within each set, producing an array of $11\times10\times8$ KB for each chip. All tests were performed at nominal supply voltage and at room temperature.

In addition to more accurately measure the timing of the critical period of system disruption a single set of ten 40 KB segments was generated from each chip starting at the same base offset under the default timing parameters of $tRCD = 0$.

As a key part of this work is the reduction in system disruption in order to measure this accurately the experiments were performed while the system was live and running the same set of background processes.

The full dataset is made openly available for the use of the research community and can be downloaded at 10.6084/m9.figshare.12149799.

V. PUF Analysis

In this section, we present an analysis of the gathered data, focusing on the key metrics of reliability, uniqueness, and query time. Further, we give an analysis of the bias of response bits across the gathered responses. Finally, an analysis of the overheads required in implementing the design is provided.

A. Uniqueness

Uniqueness is a metric that measures how distinct the PUF response of a given instance is likely to be from all the other instances in a population. It is calculated by measuring the interchip Hamming distance between each the devices in a population of m devices. The ideal value of this metric is 50%.

The calculation used in this article is as follows, where a key of n bits has been generated from a population of m devices:

$$U = \frac{2}{m(m-1)} \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} \frac{HD(R_i, R_j)}{n} \times 100\%. \qquad (1)$$

While the memory used in these experiments is all standards compliant DDR memory, the precise specifications and layout of memory components contributing to the entropy of the PUF will not be identical across chips made by different manufacturers. As such we have calculated both the overall uniqueness for the entire population of devices from all manufacturers, and on a per manufacturer basis. As can be seen in Fig. 3 and Table I there is a substantial variance in PUF uniqueness from manufacturer to manufacturer.

It can be observed from Fig. 3 that when using a solid bit-pattern in memory (0x00000000 or 0xFFFFFFFF) the uniqueness is generally poor and varies substantially between manufacturers, with the median value of Kingston being above 25% while the median value for Apacer is below 10%. However, when using a mixed bit-pattern (see Algorithm 4) the uniqueness increases substantially, becoming close to the ideal value and generally more consistent, as can be seen in Fig. 4.

This can be attributed to several factors. First the solid bit-pattern responses are heavily biased toward the pattern

Algorithm 4: Query PUF

```
Receive challenge;
From challenge extract timing values;
From challenge extract offset list;
Reserve Memory Channel $B$;
Reserve Segment of size (PUF Segment)*2 in Memory
  Channel $A$;
Write bit pattern into $B$;
for <x repetitions> do
    Set timings to $LOW$;
    for <offset 1:n> do
        Read 32 bit dword;
    end
    Restore default timings;
end
Perform error correction;
Concatenate segments into response;
Send response;
```

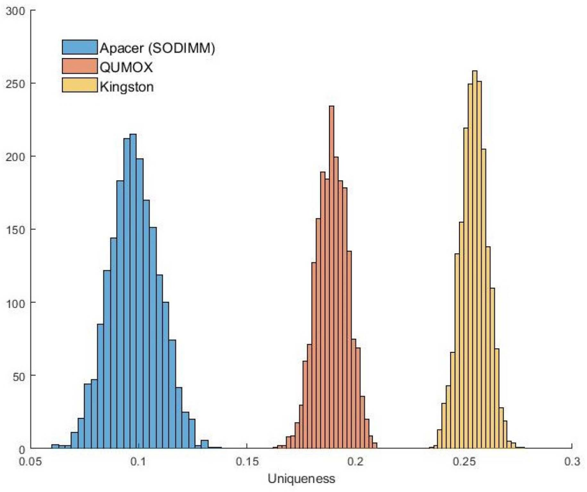

Fig. 3. Uniqueness of solid bit-pattern 32-b responses.

TABLE I
INTERCHIP UNIQUENESS FOR EACH MANUFACTURER WITH SOLID AND MIXED BIT-PATTERNS

	Solid Bit-Pattern	Mixed Bit-Pattern
OVERALL	18.05%	47.95%
QUMOX	18.91%	49.55%
KINGSTON	25.46%	49.79%
APACER	9.8%	44.53%

value. This reduces uniqueness as all of the response bits are weighted toward a certain value meaning the odds of two chips having similar responses similarly increases. However, as can be seen in Fig. 5 the bias for each bit-patterns is proportional to the other but inverted.

This means so long as the selection criteria used to generate the mixed bit-pattern results in roughly equal amounts of segments with each bit-pattern and that there is no correlation between which segments use each bit-pattern in a given PUF

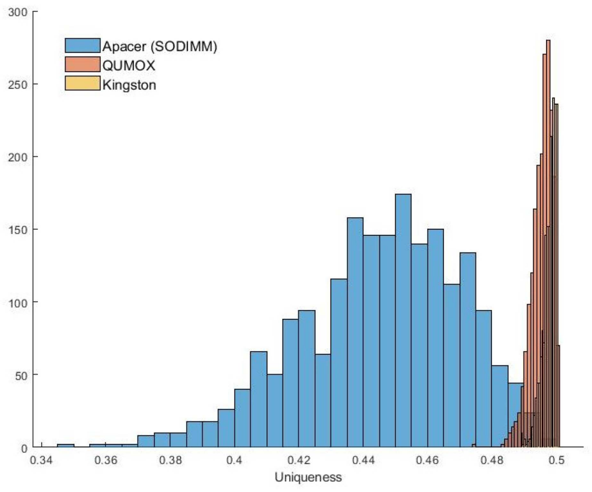

Fig. 4. Uniqueness of mixed bit-pattern 32-b responses.

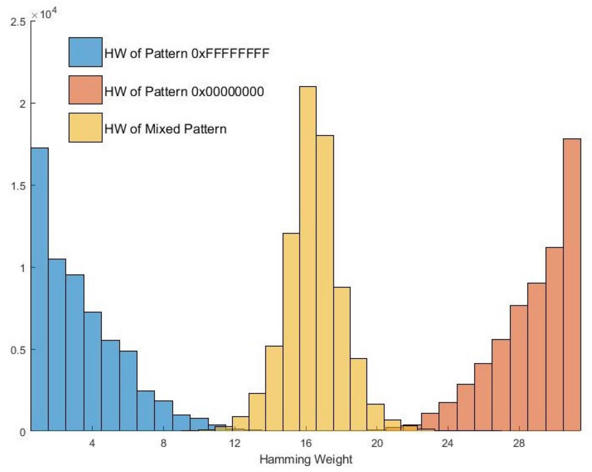

Fig. 5. Hamming Weights of 32-b responses within Apacer chips using various input values.

instance then the biases are canceled out, resulting in minimal bias in the overall PUF responses with mixed bit-patterns. It can be seen in Fig. 5 that this does occur in practice.

The variances which dictate the nature of the PUF response seem to be distributed across the memory structure with a high degree of randomness. This is consistent with the findings of previous works albeit with a more complex set of properties contributing to the overall result. As the PUF is exploiting latency in the internal memory commands some part of the entropy likely comes from the circuitry relating to these operations. Further, some variance was observed between the response of the same chips on different test devices under the same test conditions, implying that the memory controller itself may act as a partial entropy source.

Despite these factors playing some role we propose that the primary entropy source is from the DRAM cells, more specifically in the charge and discharge rates of the cell capacitors. From the data we observed no obvious correlation between the probability of bit-flips in cells when they contain a 1 or 0. That is, a cell which reliably flips when containing one value is no more or less likely to exhibit the same behavior when

TABLE II
RELIABILITY AND PROPORTION OF HIGH RELIABILITY (ABOVE 99%) 32-B SEGMENTS FOR EACH MANUFACTURER

	Reliability	Ideal Segments
OVERALL	93.59%	16.09%
QUMOX	88.66%	2.67%
KINGSTON	95.37%	12.26%
APACER	96.75%	33.35%

TABLE III
MEAN QUERY TIME AND OVERHEAD FOR AN 8 KIB SEGMENT

	Query Time	Overhead
OVERALL	3.03ms	129KiB
QUMOX	2.81ms	299KiB
KINGSTON	3.19ms	65KiB
APACER	3.1ms	24KiB

containing the inverse value. From this the most likely explanation is that the cell charge and discharge rate are the main entropy source for each value, respectively, and that these two variables are independent of one another.

B. Reliability

The ability of the PUF to generate keys consistently is of paramount importance. Reliability is a metric which measures the ability of a PUF design to generate the same response to a given challenge over repeated measurements. In this article, reliability has been calculated as follows, for m repeat measurements of n bit keys:

$$\text{Rel} = \frac{1}{m}\sum_{i=1}^{m}\frac{HD(R_{\text{ref}}, R_i)}{n} \times 100\%. \tag{2}$$

We observed that the manner in which the memory is read during a PUF query has a significant impact on the stability of the PUF. The larger the contiguous segment queried the more unstable the responses became. This is not a product of the DRAM cells as querying the same cells in smaller sections produced stable results. In practice, we found that querying more than a few KB continuously led to a degradation of results. Where larger keys are desired they can be broken down into smaller queries of a sustainable size.

Reliability is generally high with values ranging from 80%–100%. The worst reliability is found in the QUMOX chips with a mean value of 88%. As an unreliable response requires more error correction when optimizing for minimal query time it is best to use only the most reliable segments. As this excludes some percentage of the CRP set a larger PUF segment is needed to generate a response of the desired size. As can be seen in Table II if a cutoff of 99% is chosen the proportion of suitable segments varies highly between manufacturers but remains above 2.5%. These values have been used when calculating the required memory overheads in Section V-C.

C. Speed and Overheads

In order to minimize the disruption to the overall system the critical period during which memory must be reserved for PUF use must be made as short as possible. In the design proposed in [11] the average time for generating an 8 KiB key reliably is 88 ms. As can be seen in Table III our proposed design reduces this critical period to an average of 3 ms per query, a reduction of 96%. Even if ten repeated measurements are taken for the purposes of error correction as in the experiments in this work the query time is only on average 30.3 ms, a reduction of 65%.

This improvement comes mainly from two factors. First, by using the results of the read operations directly to form the response and by performing the processing for this formation after the PUF memory has been released and timings restored the period in which the PUF disrupts the system is kept to a minimum. Additionally, using the new proposed enrolment and query scheme to create a set of CRPs with near ideal reliability allows for a reduction in the number of read operations required to produce a response.

The main drawback of this approach is that it substantially increases the memory needed for the PUF. In the worst case only 2.5% of segments provide sufficient reliability to be considered ideal meaning that, as can be seen in Table III, to generate an 8 KiB response requires almost 300 KiB of memory for PUF use. However, in comparison to the capacity of most DRAM chips these overheads are still manageable and the balance of overhead to query time can be controlled by adjusting the threshold at which a segment is considered ideal. Furthermore, these overheads are required only while the PUF is operating and do not need to be reserved exclusively for PUF use. It is entirely possible to move data out of the PUF segment, perform the query, and restore the previous values, though this does incur some computational costs. It is also important to note that while a segment of 8 KiB has been evaluated for the purposes of timing analysis in practice the response size needed for device authentication and the corresponding overheads will be much smaller.

D. Potential Vulnerabilities and Security Concerns

The threat model for fully intrinsic PUFs, such as the proposed design is not identical to that models used with more conventional PUF designs. As such it is useful to discuss some of the contingencies which would need to be considered if using a PUF of this type in practice.

The major difference is the method of accessing PUF entropy—namely, the lowering of timing parameters in memory. In systems where this functionality exists it is a normal, if somewhat obscure, function of the memory controller. This means that if an attacker gains root access they can easily trigger the PUF. This is largely consistent with any PUF integrated into a complex system. If an adversary has control of the controlling system, they can access the PUF. The PUF, at least in the case of the proposed design, is a mean to authenticate devices, not a means in itself to secure the device from illicit access. What is different in fully intrinsic PUFs is that we cannot add any hardware which might limit the rate of this access without rendering them nonintrinsic.

Should an attacker gain full root access they can perform the same enrolment procedure as the device owner did to derive the full set of CRPs. This will take a considerable amount of time (in the order of 100 or 1000 times longer than generating the keys from a known challenge) and if they do not have helper data like the area of memory used for the PUF they may have to also perform this process for the entire memory of the device. While this is being done device operation will be limited by necessity and this may allow the attack to be detected if it disrupts device operations to the point that other nodes can detect the discrepancy. However, a clever adversary will perform this attack piecemeal so as to avoid detection. Once the full CRP set is collected the attacker has a digital model of that particular device that can be used to impersonate it, but this would not be transferable to another device of the same type as PUF CRP sets are unique to a specific item of hardware.

Each cell provides a distinct entropy source, meaning that acquiring the CRP data for a given memory segment does not allow the prediction through modeling of other segments in the same chip. We can conceptualize the proposed design as a weak PUF in the vein of SRAM PUF, with two CRPs that have very large responses and can be divided and recombined to form a response of the desired size. ML modeling attacks are therefore not applicable to the proposed design.

If an attacker were to gain root access to the system it is entirely possible to use the PUF query mechanism without the careful segregation of memory channels in order to destabilize the system and force a crash. It is worth noting this is not actually a weakness of the PUF design specifically, and could be done in any system with the right memory controller regardless of whether it was being used for PUF response generation. Likewise, a sufficiently knowledgeable adversary with root access has many options for crashing a system, many of which are easier to perform than this method.

If we assume that the attacker only has root access for a limited window then acquiring the CRP set becomes a nontrivial challenge. The enrolment process takes considerable time, and the attacker must possess helper data to know which memory segment to target. It would be possible for an attacker to gather the CRP set relatively quickly if the challenge set is known, but as this set is not held on device they attacker would have to first compromise the authentication server or perform some kind of man-in-the-middle attack to gradually gather the challenge set by listening in on normal operations.

In general, as is the case with any individual security primitive, there are means by which a sufficiently knowledgeable and resourced adversary can compromise a PUF of this type. It is important, therefore, to keep in mind that this PUF provides only an element of hardware derived authentication and cannot be relied on as a means of device security in and of itself, but rather should be integrated into existing security mechanism as an additional assurance of device authenticity. Hardware rooted security is not a silver bullet. It is simply another hurdle for adversaries to overcome before they can compromise a system, and critically which requires different knowledge to attack than other aspects of system security.

E. Technical Challenges of Implementation

There are several difficulties inherent in implementing a DRAM latency PUF on commodity hardware. In this section we discuss the technical challenges which must be overcome when integrating a PUF of this type into a COTS system.

1) Memory Controller Requirements: A dynamic memory controller (DMC) which has in-runtime access to timing parameters is needed. This is not insurmountable as many existing architectures use such a memory controller. Snapdragon processors for mobile phones have this capability, as do the majority of server and HPC processor architectures. At the level of desktop computing AMD processors use a DMC on all multichannel systems. The testbeds used in this article all use AMD processors for this reason. Nonetheless, it limits the viable platforms for implementing a PUF of this type. In architectures, where this functionality is present it is a low level operation of the memory controller and so must be implemented at a level where direct control of the memory controller is possible. A major factor in how well a PUF of this type can be implemented is the granularity of the memory controller. Ideally the timing parameters would be modified for only a single bank of memory, so that disruption to the system is minimal. In practice depending on the specifics of the memory controller the granularity may in fact only be to the level of a chip, a rank, or an entire memory channel. In the case of our experimental testbeds this function was only possible at the level of memory channels, meaning that during PUF operation an entire memory channel had to be rendered temporarily inaccessible.

2) Maintaining System Stability: In the case where the smallest section of memory that can have timing values altered is relatively large there is a real technical challenge in lowering these parameters without destabilizing the entire system. This is because when the timings are lowered to the required levels, while the data in memory remains unaltered, read operations during this period will return junk data. If a critical process attempts to do this it will cause severe problems. This means that before the PUF is triggered it must be ensured that there is no data in the same channel being used by the PUF which is required by any critical process. It must also be ensured that noncritical processes either do not have any necessary data held in this part of memory, or that those processes are paused for the duration of the PUF query. This is not a trivial task, though it is worth noting that the query time as listed in Section V-C is for a much larger response than would be used in practice. The query time required to generate an authentication ID in a real system would likely be no more than 100 ms (for a 2048 b ID).

Similarly, there are challenges in regards to the management of the data held in the PUF segment and of the response data as it is read. The PUF has to write a specific bit pattern into a specific segment of memory as part of the query procedure. If data is already present it must be shuffled out of this part of memory and restored after the PUF query is complete. This could be avoided by reserving a small amount of memory exclusively for PUF usage depending on how much memory is available. Some area of memory must also be used to store the response as it is generated. Here, there is less of a problem, as the alteration of timing parameters used in this design does not affect the stability of write operations. Hence, the memory used for this can be in the same memory being used by the PUF already.

VI. Conclusion

In this article, we have presented a novel method for extracting PUF responses from DRAM memory modules in-runtime on live COTS systems. We propose a new enrolment and query system for latency-based DRAM PUFs and have used these algorithms to generate a large scale dataset of PUF challenge-responses under various timing parameter adjustments which is provided for the use of the research community. Our analysis of this data shows that the proposed design exhibits near ideal uniqueness and high reliability. Furthermore, the proposed design reduces the critical period of system disruption by up to 96% relative to comparable designs though at the cost of increased memory overheads.

Acknowledgment

The data from the experiments in this article is available for the use of the research community and can be downloaded at 10.6084/m9.figshare.12149799.

References

[1] J. Guajardo, S. S. Kumar, G.-J. Schrijen, and P. Tuyls, "FPGA intrinsic PUFs and their use for IP protection," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst. (CHES)*, 2007, pp. 63–80. doi: 10.1007/978-3-540-74735-2_5.

[2] F. Tehranipoor, N. Karimian, W. Yan, and J. A. Chandy, "DRAM-based intrinsic physically unclonable functions for system-level security and authentication," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 3, pp. 1085–1097, Mar. 2017.

[3] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *Proc. 44th ACM/IEEE Des. Autom. Conf.*, 2007, pp. 9–14.

[4] A. Maiti and P. Schaumont, "Improved ring oscillator PUF: An FPGA-friendly secure primitive," *J. Cryptol.*, vol. 24, pp. 375–397, Oct. 2010.

[5] A. Maiti and P. Schaumont, "A novel microprocessor-intrinsic physical unclonable function," in *Proc. FPL*, 2012, pp. 380–387. [Online]. Available: https://doi.org/10.1109/FPL.2012.6339208

[6] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Silicon physical random functions," in *Proc. 9th ACM Conf. Comput. Commun. Security (CCS)*, 2002, p. 148.

[7] S. Kumar, J. Guajardo, R. Maes, G.-J. Schrijen, and P. Tuyls, "Extended abstract: The butterfly PUF protecting IP on every FPGA," in *Proc. IEEE Int. Workshop Hardw. Orient. Security Trust (HOST)*, 2008, pp. 67–70, doi: 10.1109/HST.2008.4559053.

[8] L. A. Bathen, M. Gottscho, N. Dutt, P. Gupta, and A. Nicolau, "ViPZonE: OS-level memory variability-driven physical address zoning for energy savings," in *Proc. CODES+ISSS*, 2012, pp. 33–42, doi: 10.1145/2380445.2380457.

[9] *"256Mb: x4, x8, x16 SDRAM" MT48LC64M4A2/MT48LC32M8A2/MT48LC16M16A2 Datasheet*, Micron, Boise, ID, USA, Nov. 1999.

[10] A. Schaller *et al.*, "Intrinsic Rowhammer PUFs: Leveraging the rowhammer effect for improved security," in *Proc. HOST*, 2017, pp. 1–7, doi: 10.1109/HST.2017.7951729.

[11] J. Kim, M. Patel, H. Hassan, and O. Mutlu, "The DRAM latency PUF: Quickly evaluating physical unclonable functions by exploiting the latency-reliability tradeoff in modern commodity DRAM devices," in *Proc. HPCA*, 2018, pp. 194–207, doi: 10.1109/HPCA.2018.00026.

[12] F. Tehranipoor, N. Karimian, W. Yan, and J. A. Chandy, "Investigation of DRAM PUFs reliability under device accelerated aging effects," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Baltimore, MD, USA, 2017, pp. 1–4, doi: 10.1109/ISCAS.2017.8050629.

[13] S. Sutar, A. Raha, and V. Raghunathan, "D-PUF: An intrinsically reconfigurable DRAM PUF for device authentication in embedded systems," in *Proc. Int. Conf. Compliers Archit. Syth. Embedded Syst. (CASES)*, Pittsburgh, PA, USA, 2016, pp. 1–10, doi: 10.1145/2968455.2968519.

[14] W. Xiong *et al.*, *Run-Time Accessible DRAM PUFs in Commodity Devices* (Lecture Notes in Computer Science), pp. 432–453, 2016, doi: 10.1007/978-3-662-53140-2_21.

[15] A. Schaller *et al.*, "Decay-based DRAM PUFs in commodity devices," *IEEE Trans. Depend. Secure Comput.*, vol. 16, no. 3, pp. 462–475, May/Jun. 2019, doi: 10.1109/TDSC.2018.2822298.

Jack Miskelly received the B.Eng. (First-Class Hons.) and M.Eng. degrees in software and electronic systems engineering from Queen's University Belfast, Belfast, U.K., where he is currently pursuing the Ph.D. degree with the Centre for Secure Information Technologies.

His research interests include hardware security, true random number generators, IoT security, and physical unclonable functions with a focus on intrinsic PUFs.

Máire O'Neill (Senior Member, IEEE) received the M.Eng. (with Distinction) and Ph.D. degrees in electrical and electronic engineering from Queen's University Belfast, Belfast, U.K., in 1999 and 2002, respectively.

She is the Chair of information security with Queen's University Belfast and received an EPSRC leadership fellowship to conduct research into next generation data security architectures. She previously held a United Kingdom Royal Academy of Engineering research fellowship from 2003 to 2008. She has authored two research books and has more than 100 peer-reviewed conference and journal publications. Her research interests include hardware cryptographic architectures, side channel analysis, physical unclonable functions, and post-quantum cryptography.

Dr. O'Neill was awarded a Royal Academy of Engineering Silver Medal in 2014, which recognizes outstanding personal contribution by an early or mid-career engineer that has resulted in successful market exploitation.