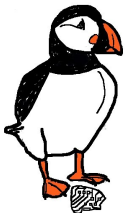

PUFFIN

Physically unclonable functions found in standard PC components

Project number: 284833
FP7-ICT-2011-C

D2.1

Scientific contribution of WP2, part 1 Analysis and Qualification

Due date of deliverable: 31. July 2013
Actual submission date: 30. September 2013

WP contributing to the deliverable: WP2

Start date of project: 1. February 2012 Duration: 3 years

Coordinator:
Technische Universiteit Eindhoven
Email: coordinator@puffin.eu.org
www.puffin.eu.org

Revision 1.0

Project co-funded by the European Commission within the 7th Framework Programme		
Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission services)	
RE	Restricted to a group specified by the consortium (including the Commission services)	
CO	Confidential, only for members of the consortium (including the Commission services)	

Scientific contribution of WP2, part 1
Analysis and Qualification

V. van der Leest (IID)
Ruben Niederhagen (TUE)
and the WP2 team

30. September 2013
Revision 1.0

The work described in this report has in part been supported by the Commission of the European Communities through the FP7 program under project number 284833.

Abstract

This document contains an overview of the work and results from Work Package 2 (WP2) of the PUFFIN project. The work in WP2 can be divided into two main parts: analysis of PUF measurements (from WP1) and development of new methodologies for evaluating PUF behaviour. This document describes what has been achieved in both of these areas during the first phase (18 months) of the PUFFIN project. The results of the work performed in WP2 will serve as input for WP3, since WP2 will show which PUFs (and therefore which devices) have the required properties to implement certain specific use cases.

Keywords: WP2, PUF analysis

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Work Package 2 (WP2) of the PUFFIN project focusses on analysis and qualification of the PUFs that have been found in WP1. Based on this goal, WP2 is divided into two main parts:

- The analysis of PUF measurements (from WP1), and
- the development of new methodologies for evaluating PUF behaviour.

This document describes what has been achieved in both of these areas during the first phase (18 months) of the PUFFIN project. The results of the work performed in WP2 serve as input for WP3, since WP2 shows which PUFs (and therefore which platforms) have the required properties to implement specific use cases from WP3.

Chapter 2 of this deliverable provides an overview of the tests that have been performed on the different PUF measurements from WP1. These tests have been used as a preliminary investigation into the suitability of the different PUFs from commercially available devices for actual use in PUF implementations. Due to the limited number of devices measured (and the limited set of environmental tests performed), the overview presented in this section does not offer a thorough qualification of the measured PUFs yet. However, the results can already be used to distinguish between platforms that will not be suitable for implementing PUF-based security primitives and those that do seem promising.

Chapter 3 describes which new methodologies for evaluating PUF behaviour have been developed in the PUFFIN project. These two methodologies each focus on one of the two basic properties of PUFs: reliability and uniqueness. This has resulted in the following new methodologies for evaluating PUF behaviour:

- A new approach for modelling noise behaviour of PUFs (reliability), and
- a new method for deriving extractable entropy for PUF instances (focussed on uniqueness, but also taking reliability into account).

Both of these new methodologies are described in detail in two scientific publications that are attached to this deliverable as appendices.

Chapter 2

Preliminary Analysis of PUFs

2.1 Introduction

In the first period of the PUFFIN project WP2 has received PUF measurements from WP1 for analysis. Up to now all measured PUF behaviour has been derived from SRAM memories of commercially available devices. A term generally used for these devices is *Commercial Off-The-Shelve* (COTS) devices. SRAM that has been analysed so far in the PUFFIN project originates from the following platforms:

- Ainol Novo 7 tablets,
- Texas Instruments MSP430F5308 microcontrollers,
- Microchip PIC16F1825 microcontrollers,
- ST STM32F100R8 microcontrollers,
- ST STM32F100RB microcontrollers,
- Atmel ATMega328p microcontrollers,
- NVIDIA GeForce GTX 295 graphics card, and
- Pandaboards (computer development platform containing either Texas Instruments OMAP4430 or 4460, see `www.pandaboard.org` for more information).

Deliverable D1.1 of the PUFFIN project explains which SRAM memories of these platforms have been used for these measurements and how data has been extracted.

Note: Some platforms from D1.1 have not been analysed yet, because the research and measurements performed in WP1 for these platforms are not yet considered to be sufficiently mature. Once these measurements are more mature, these platforms will be added to the overview.

2.2 Test Descriptions

The following tests have been performed to evaluate reliability and uniqueness of the PUFs from devices of the different platforms:

- Repeated Start-up Test (RST),
- Temperature Cycle Test (TCT),
- Between-class Hamming Distance Test (BCHDT), and
- Hamming Weight Test (HWT).

The following sections provide a description for each of these tests. The Temperature Cycle Test is the only test that has not been performed on all platforms for the following reasons:

- Some platforms (tablets, GPUs, and Pandaboards) cannot be measured under extreme temperature conditions, because some components of these platforms will not be able survive extreme heat or cold.
- Microcontrollers do survive exposure to extreme temperatures; we have decided to use one of the microcontroller platforms as an example in the Temperature Cycle Test.

Based on these reasons, TI's MSP430F5308 microcontroller has been chosen for the Temperature Cycle Test. All other devices have been tested with the remaining three tests.

2.2.1 Repeated Start-up Test

This basic test measures the noise characteristics of the PUF candidates by comparing several PUF responses *within-class*, i.e., of the same device. Each device of a platform containing a PUF (found in WP1) is measured repeatedly. The measurements are performed "on the desk" under room temperature and uncontrolled humidity conditions. The PUF response of each measurement is stored on a hard drive and later analysed by software.

One PUF measurement (usually the first one) of each device is considered as enrolment measurement. A Matlab script is used to compare (fractional) Hamming distances between the enrolment measurement and all other PUF responses of the device. The Hamming distances between the PUF measurements must be small in order to identify a device with high reliability.

2.2.2 Temperature Cycle Test

This within-class test measures noise characteristics and thus the reliability of the PUFs for a specific platform under different ambient temperatures. The PUF response of a device is measured repeatedly under well defined ambient conditions and each measurement is stored on a hard disk and finally analysed by software. Measurement files are sorted into folders according to the conditions at which they were taken (e.g., folder names 'Temp-30', 'Temp25', 'Temp90' indicate that measurements stored in these folders were taken at -30^oC, $+25^o$C and $+90^o$C temperature respectively).

An enrolment measurement of each device is taken at $+25^o$C. A Matlab script is used to compare the (fractional) Hamming distances between the enrolment measurement and the

other measured PUF responses of the device. A PUF is considered reliable if the Hamming distances to the enrolment measurement are small under all ambient conditions.

As described above, this test has only been performed on TI's MSP430F5308 microcontroller.

2.2.3 Between-Class Hamming Distance Test

This test investigates the uniqueness of PUF responses by comparing PUF enrolment measurements (from the Repeated Start-up Test) *between-class*, i.e., between several devices of the same platform.

A Matlab script is used to compare the Hamming distances between the enrolment measurements of the devices. Each device can only be uniquely identified if there is only a small correlation between PUFs from different devices, i.e., if the fractional Hamming distances between the enrolment measurements have a Gaussian distribution with a mean value close to 50%.

2.2.4 Hamming Weight Test

This test is performed using either the measurements from the Repeated Start-up Test or from the Temperature Cycle Test. It investigates whether PUF responses have a bias to either 0 or 1 during start-up (possibly at different temperatures).

A Matlab script is used to calculate the (fractional) Hamming weight of all measured PUF responses of a device. The Hamming weight is an indication for bias in the PUF responses: Non-biased PUF responses have an even distribution in zero and non-zero bits. Therefore, the fractional Hamming weight of the measurements should be close to 50%, indicating that about half the bits of the response are 0 and the other half are 1.

2.3 Test Results

2.3.1 Ainol Novo 7 Tablet

Information

Number of devices measured:	7
Number of measurements per device:	5
PUF type:	SRAM PUF
PUF size:	1KB

Repeated Start-up Test

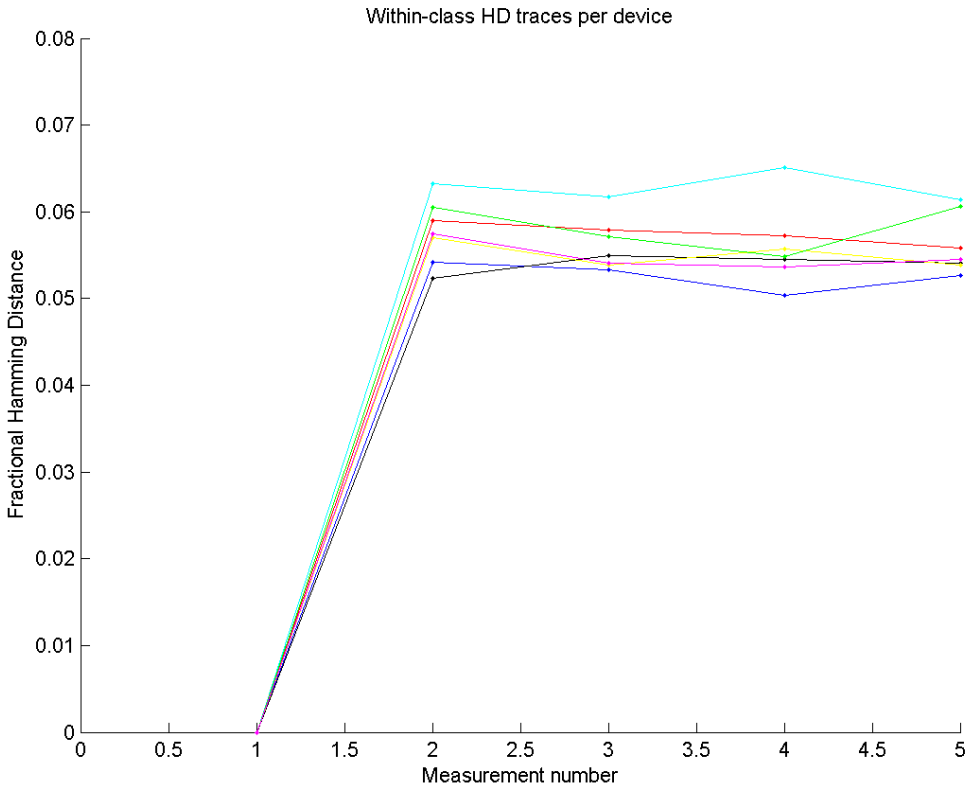

Figure 2.3.1: Within-class Hamming distance of SRAM in Ainol Novo 7 tablets.

Figure 2.3.1 shows the results from the 5 measurements of the Repeated Start-up Test for the seven Ainol Novo 7 tablets (each individual line representing one of the devices). For each device, the first measurement has been used for enrolment (so it has 0% noise); all other measurements are compared to this enrolment measurement. The figure shows that the maximum within-class Hamming distance for the devices of this platform (at room temperature) is less than 7%. This amount of noise can easily be corrected using commonly known Fuzzy Extractors. Therefore, **these devices pass the Repeated Start-up Test**.

Between-Class Hamming Distance Test

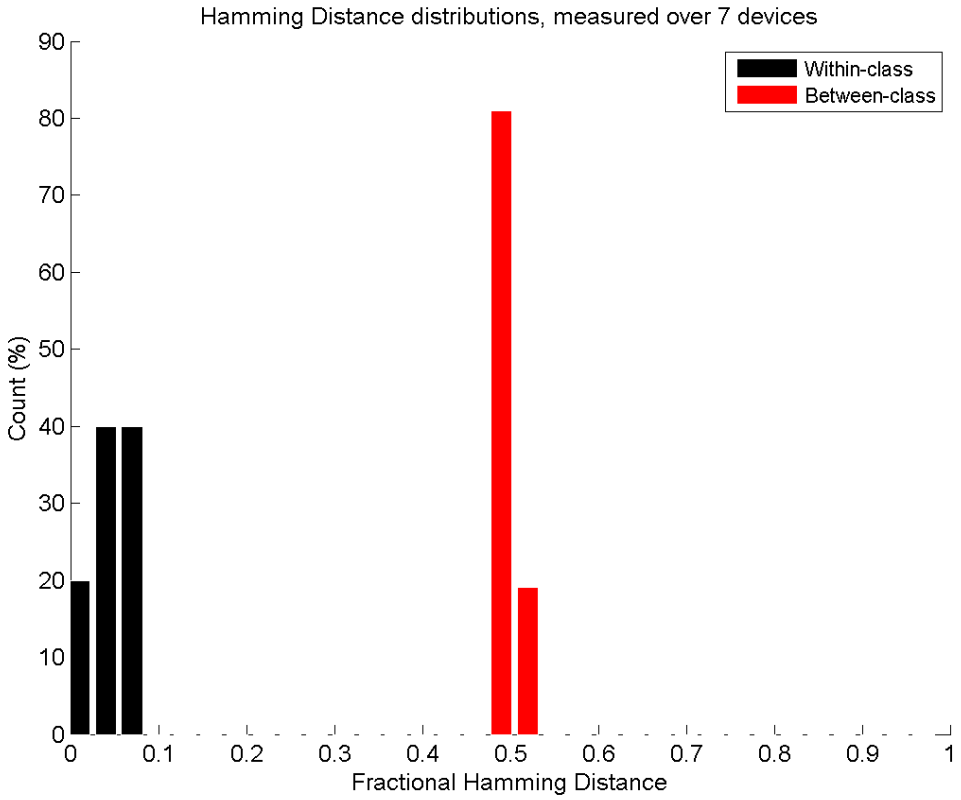

Figure 2.3.2: Between-class versus within-class fractional Hamming distances of SRAM in Ainol Novo 7 tablet measurements.

Figure 2.3.2 compares the results from the Repeated Start-up Test (in black) to the results of the Between-Class Hamming Distance Test (in red) for the seven Ainol Novo 7 tablets. The Hamming distance between different devices is much higher than the noise measured for each individual device. This indicates that each device can be uniquely identified based on its PUF responses.

In more detail: We calculated $7 \times 6 : 2 = 21$ fractional Hamming distances between the seven devices. These 21 distances fit to a Gaussian distribution with a mean value of 49.9%. This is an indication that there is very little correlation between the PUF responses from different devices, which makes them suitable as an input for commonly known Fuzzy Extractors. Therefore, **these devices pass the Between-Class Hamming Distance Test**.

Hamming Weight Test

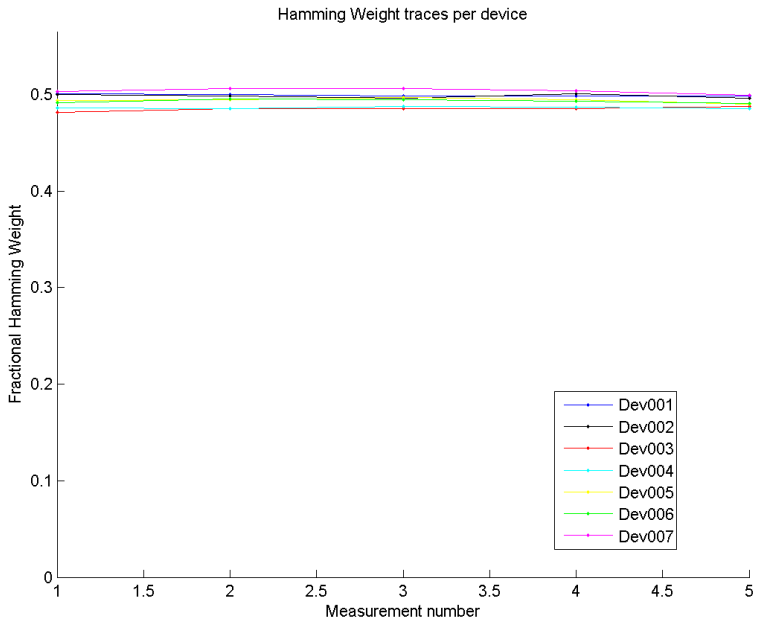

Figure 2.3.3: Fractional Hamming weight of SRAM PUFs in Ainol Novo 7 tablets.

Figure 2.3.3 shows the Hamming weight of 5 measurements from the Repeated Start-up Test for each of the seven Ainol Novo 7 tablets (each individual line represents one of the devices). For all devices, the Hamming weight of the measurements is close to 50%, indicating an equal number of 0's and 1's in the PUF responses (also visible in the plotted example PUF response in Figure 2.3.4). This indicates that these PUF responses are suitable inputs for commonly known Fuzzy Extractors. Therefore, **these devices pass the Hamming Weight Test**.

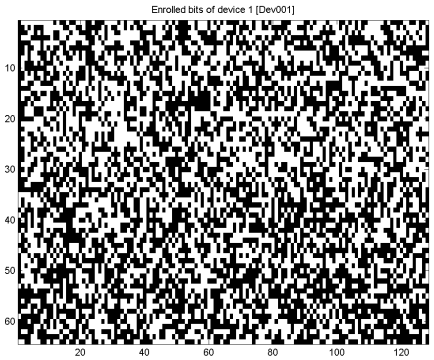

Figure 2.3.4: Example measurement of an SRAM PUF response from an Ainol Novo 7 tablet.

2.3.2 Texas Instruments MSP430F5308

Information

Number of devices measured:	15
Number of measurements per device:	> 1000 (varying per device)
PUF type:	SRAM PUF
PUF size:	6KB

Repeated Start-up Test

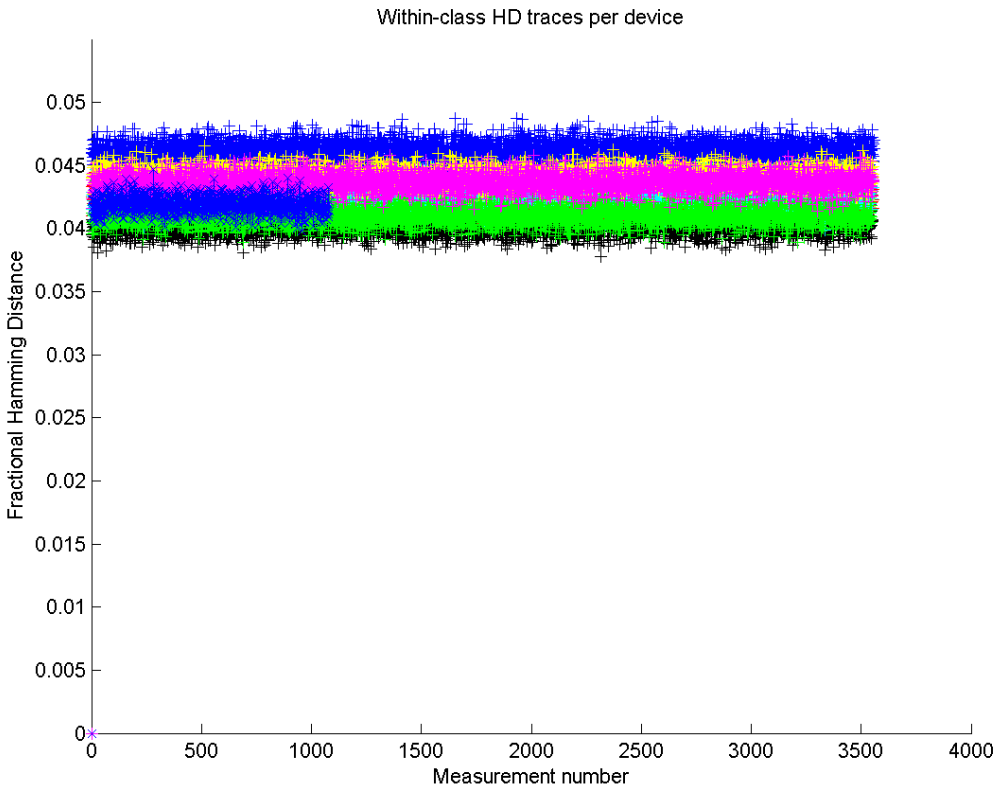

Figure 2.3.5: Within-class Hamming distance of SRAM in MSP430F5308 measurements.

Figure 2.3.5 shows the results from the measurements of the Repeated Start-up Test for the 15 Texas Instruments MSP430F5308 microcontrollers (each individual line representing one of the devices). For each device, the first measurement has been used for enrolment (so it has 0% noise); all other measurements are compared to this enrolment measurement. The figure shows that the maximum within-class Hamming distance for these devices (at room temperature) is less than 5%. This amount of noise can easily be corrected using commonly known Fuzzy Extractors. Therefore, **these devices pass the Repeated Start-up Test**.

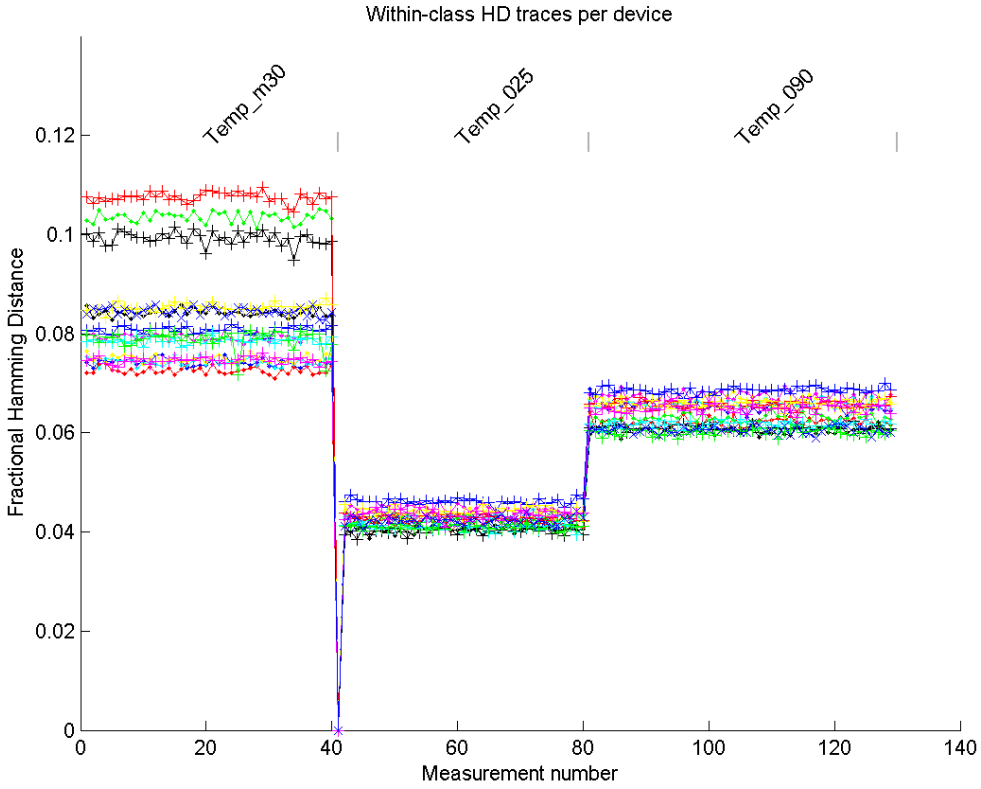

Figure 2.3.6: Within-class Hamming distance of SRAM in MSP430F5308 measured over different temperatures.

Temperature Cycle Test

Figure 2.3.6 shows the results from the measurements of the Temperature Cycle Test for the 15 Texas Instruments MSP430F5308 microcontrollers (each individual line representing one of the devices). PUF responses for all devices have been measured at three different temperatures: -30^oC, $+25^oC$ and $+90^oC$. For all devices the first measurement at $+25^oC$ has been used for enrolment (so it has 0% noise); all other measurements are compared to this enrolment measurement. The figure shows that the maximum within-class Hamming distance for these devices, over all tested temperatures, is less than 11%. This amount of noise can easily be corrected using commonly known Fuzzy Extractors. Therefore, **these devices pass the Temperature Cycle Test**.

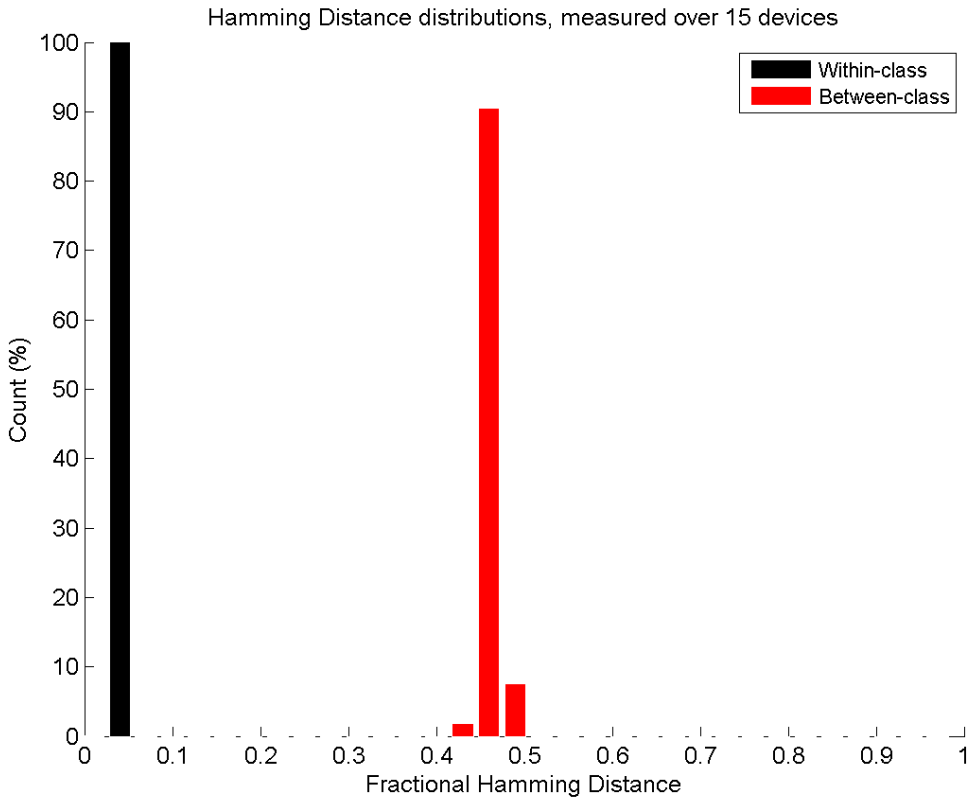

Figure 2.3.7: Between-class versus within-class Hamming distance of SRAM in MSP430F5308 measurements.

Between-Class Hamming Distance Test

Figure 2.3.7 compares the results from the Repeated Start-up Test (in black) with the results of the Between-Class Hamming Distance Test (in red) for the 15 Texas Instruments MSP430F5308 microcontrollers. This figure clearly shows that the Hamming distance between different devices is much higher than the noise measured for each individual device. This indicates that the devices can all be uniquely identified based on their PUF responses.

In more detail: We calculated $15 \times 14 : 2 = 105$ Hamming distances between the 15 devices. These fractional distances fit a Gaussian distribution with a mean value of 46.4%. Since this value is a bit lower than 50%, this is an indication that there is some correlation between the PUF responses from different devices, but they should still be suitable as an input for commonly known Fuzzy Extractors. Therefore, **these devices (weakly) pass the Between-Class Hamming Distance Test**.

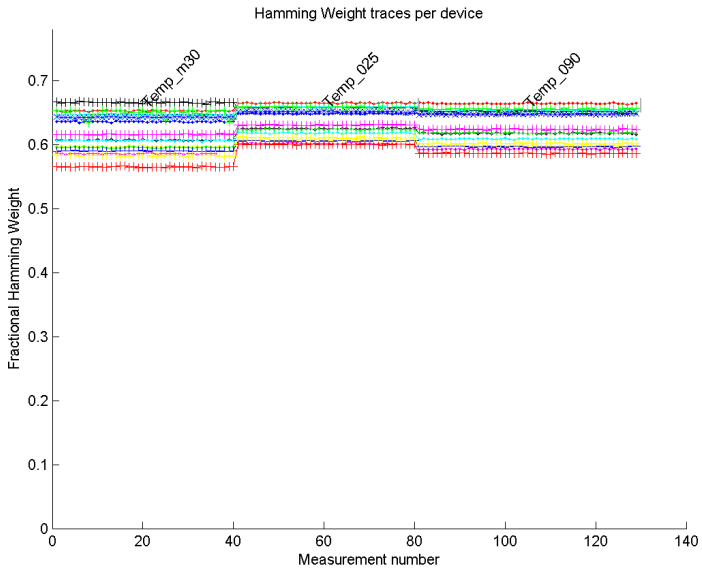

Figure 2.3.8: Hamming weight of SRAM in MSP430F5308 measurements.

Hamming Weight Test

Figure 2.3.8 shows the Hamming weight of the measurements from the Temperature Cycle Test for the 15 Texas Instruments MSP430F5308 microcontrollers (each individual line representing one of the devices). For all devices, the fractional Hamming weight of the measurements is significantly higher than 50%. This means that there are more 1's than 0's in the PUF responses (also visible in the plotted example PUF response in Figure 2.3.9). This indicates that these PUF responses will require significant pre-processing before they are suitable inputs for commonly known Fuzzy Extractors. This causes some overhead in the size requirements for the PUF response; however, these requirements can most likely be fulfilled. Therefore, **these devices (weakly) pass the Hamming Weight Test**.

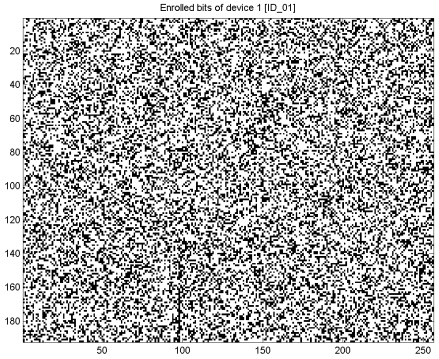

Figure 2.3.9: Example of SRAM PUF response from MSP430F5308 measurement.

2.3.3 Microchip PIC16F1825

Information

Number of devices measured:	16
Number of measurements per device:	5
PUF type:	SRAM PUF
PUF size:	1KB

Repeated Start-up Test

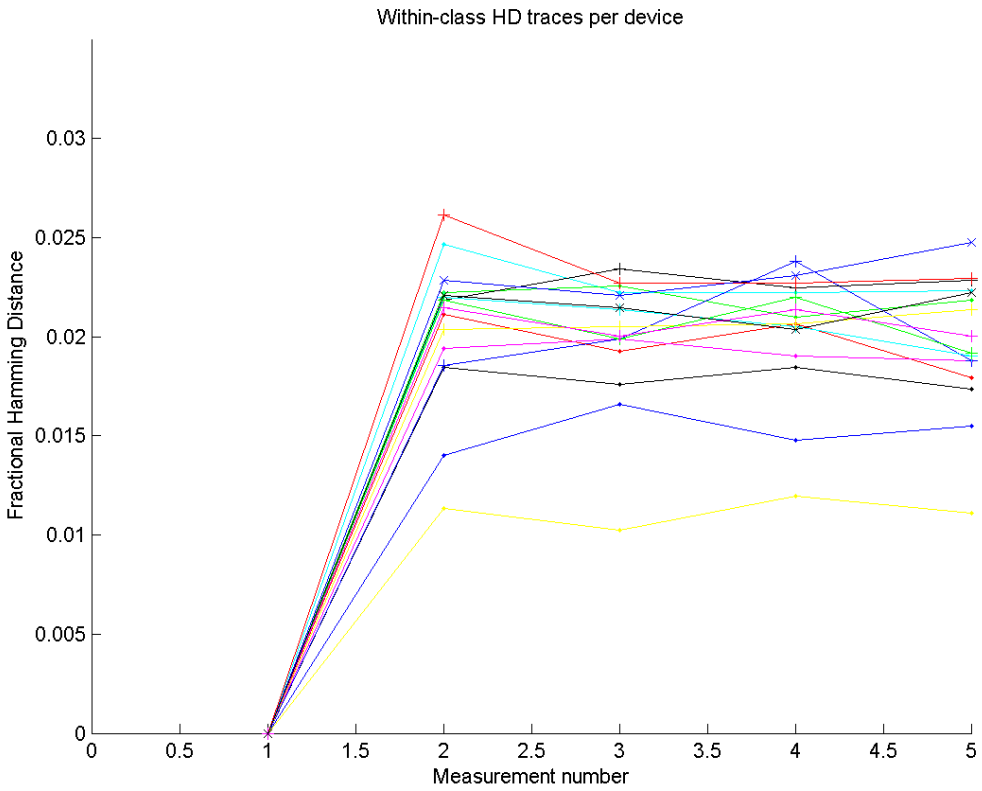

Figure 2.3.10: Within-class Hamming distance of SRAM in PIC16F1825 measurements.

Figure 2.3.10 shows the results from 5 measurements of the Repeated Start-up Test for each of the 16 Microchip PIC16F1825 microcontrollers (each individual line representing one of the devices). For all devices, the first measurement is used for enrolment (so it has 0% noise). All other measurements are compared to this enrolment measurement. The maximum within-class Hamming distance for these devices (at room temperature) is below 3%. This amount of noise can easily be corrected using commonly known Fuzzy Extractors. Therefore, **these devices pass the Repeated Start-up Test**.

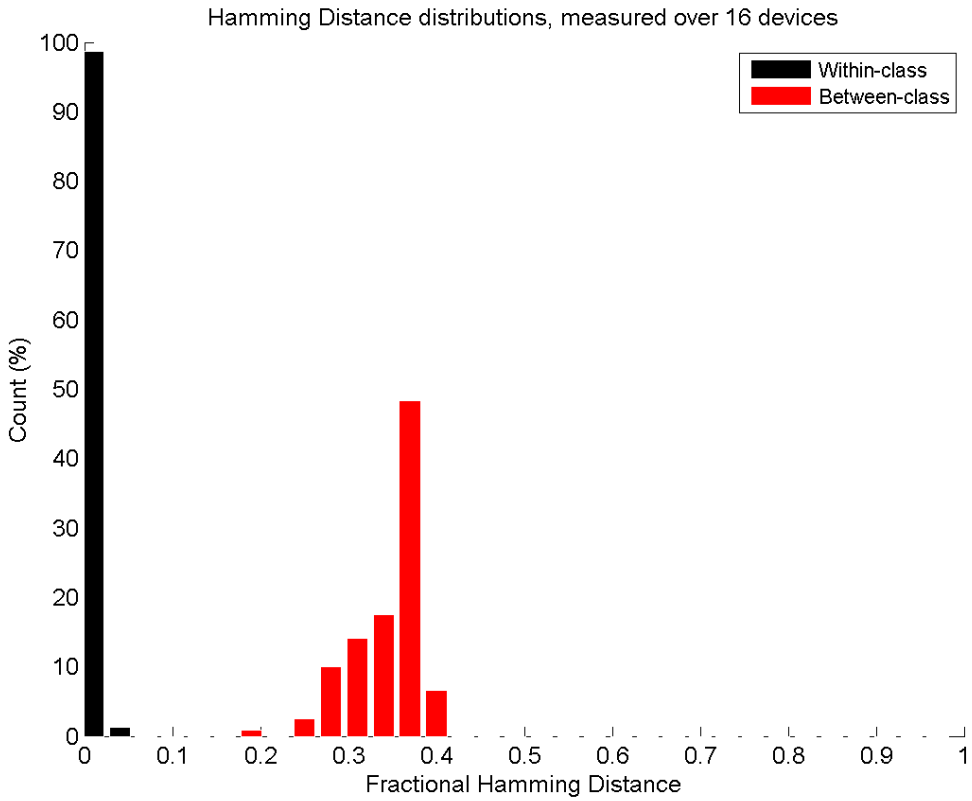

Figure 2.3.11: Between-class versus within-class Hamming distance of SRAM in PIC16F1825 measurements.

Between-Class Hamming Distance Test

Figure 2.3.11 compares the results from the Repeated Start-up Test (in black) to the results of the Between-Class Hamming Distance Test (in red) for the 16 Microchip PIC16F1825 microcontrollers. The Hamming distance between different devices is much lower than required for a PUF implementation.

In more detail: We calculated $16 \times 15 : 2 = 120$ Hamming distances between the 16 devices. These 120 fractional distances fit a Gaussian distribution with a mean value of 34.5%. This low mean value indicates that there is too much correlation between the PUF responses from different devices to be suitable as an input for commonly known Fuzzy Extractors. Therefore, **these devices fail the Between-Class Hamming Distance Test**.

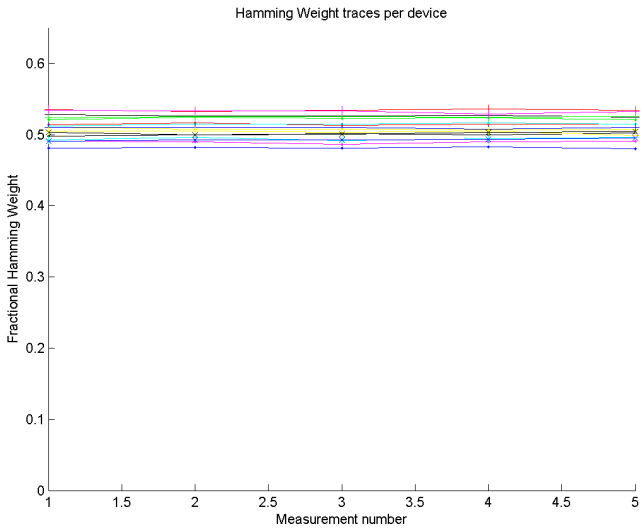

Figure 2.3.12: Hamming weight of SRAM in PIC16F1825 measurements.

Hamming Weight Test

Figure 2.3.12 shows the Hamming weight of the 5 measurements from the Repeated Start-up Test for 15 Texas Instruments MSP430F5308 microcontrollers (each individual line representing one of the devices). The fractional Hamming weight of the measurements is close to 50%. This does not give any indication for a bias that might have caused the failure of the Between-Class Hamming Distance Test. However, when examining the PUF responses visually (see the example PUF response in Figure 2.3.13), it turns out that there is clearly some bias present: each byte of a response has a preference to have either all bits 0 or all bits 1 (with an alternating pattern). This pattern explains the poor Hamming distance between devices and highlights the very biased Hamming weights per byte. Therefore, **these devices fail the Hamming Weight Test** due to this local biasing.

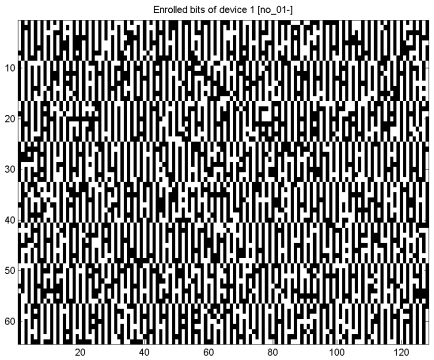

Figure 2.3.13: Example of SRAM PUF response from PIC16F1825 measurement.

2.3.4 ST STM32F100R8

Information

Number of devices measured:	11
Number of measurements per device:	144
PUF type:	SRAM PUF
PUF size:	8KB

Repeated Start-up Test

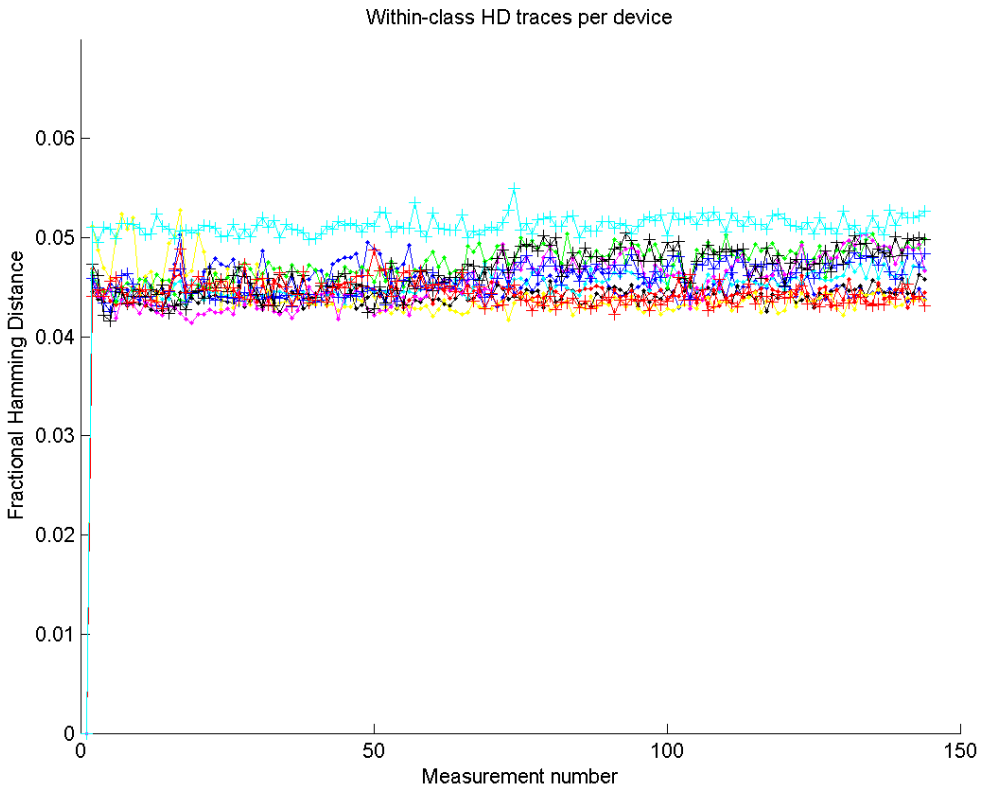

Figure 2.3.14: Within-class Hamming distance of SRAM in STM32F100R8 measurements.

Figure 2.3.14 shows Hamming distance from 144 measurements of the Repeated Start-up Test for the 11 ST STM32F100R8 microcontrollers (each individual line representing one of the devices). For all devices, the first measurement has been used for enrolment (so it has 0% noise); all other measurements have been compared to this enrolment measurement. The maximum within-class Hamming distance for these devices (at room temperature) is less than 6%. This amount of noise can easily be corrected using commonly known Fuzzy Extractors. Therefore, **these devices pass the Repeated Start-up Test**.

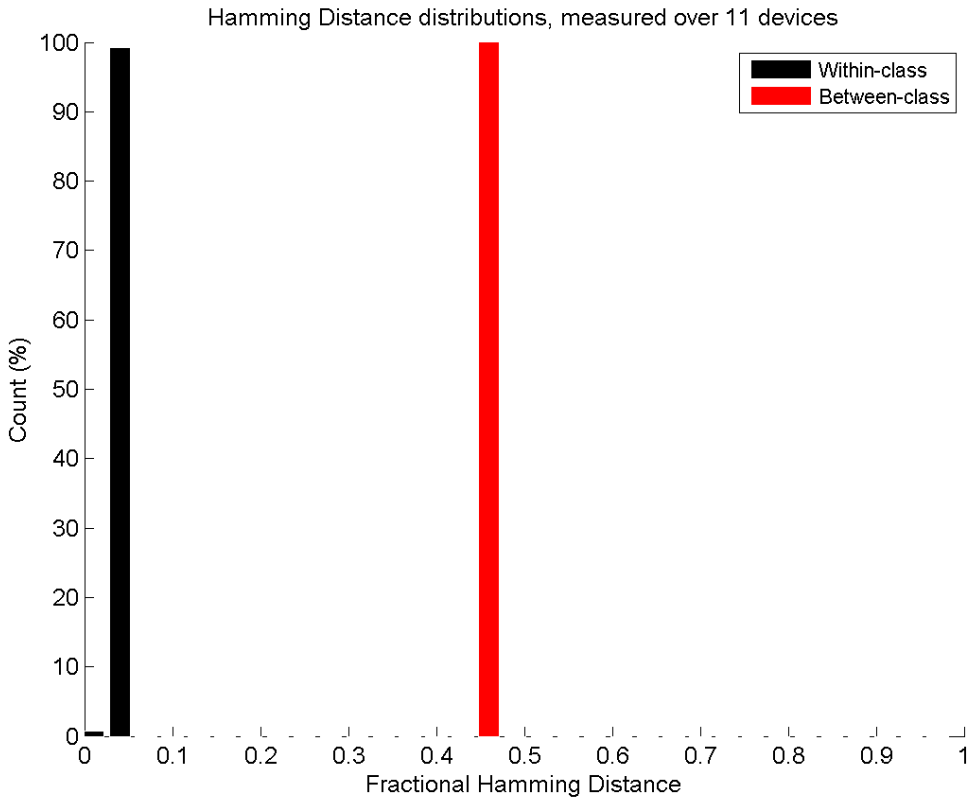

Figure 2.3.15: Between-class versus within-class Hamming distance of SRAM in STM32F100R8 measurements.

Between-Class Hamming Distance Test

Figure 2.3.15 compares the results from the Repeated Start-up Test (in black) to the results of the Between-Class Hamming Distance Test (in red) for the 11 ST STM32F100R8 microcontrollers. The Hamming distance between different devices is much higher than the noise measured for each individual device. This indicates that the devices can all be uniquely identified based on their PUF responses.

In more detail: We calculated $11 \times 10 : 2 = 55$ Hamming distances between the 11 devices. These 55 fractional Hamming distances fit a Gaussian distribution with a mean value of 46.7%. This is an indication that there is some correlation between the PUF responses from different devices, but they should still be suitable as an input for commonly known Fuzzy Extractors. Therefore, **these devices (weakly) pass the Between-Class Hamming Distance Test**.

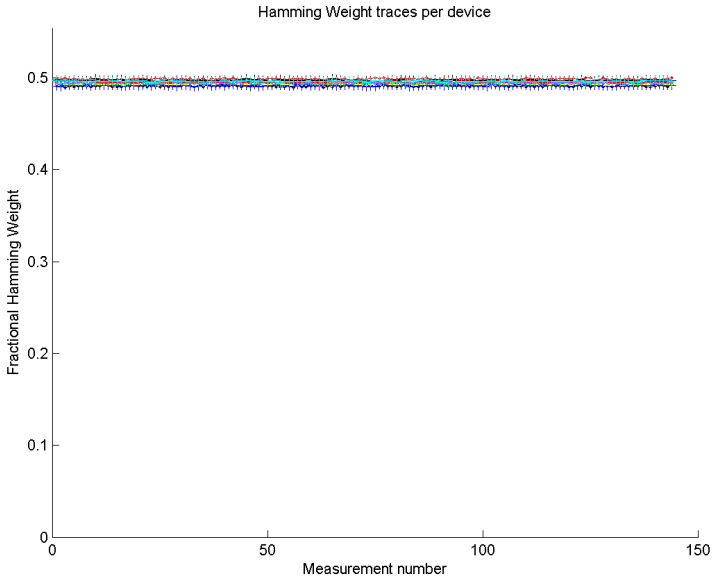

Figure 2.3.16: Hamming weight of SRAM in STM32F100R8 measurements.

Hamming Weight Test

Figure 2.3.16 shows the Hamming weight of the 144 measurements from the Repeated Start-up Test for the 11 ST STM32F100R8 microcontrollers (each individual line representing one of the devices). For all devices, the Hamming weight of the measurements is close to 50%, i.e., there is an equal number of 0's and 1's in the PUF responses (an example PUF response is plotted in Figure 2.3.17). This indicates that these PUF responses are suitable inputs for commonly known Fuzzy Extractors. Therefore, **these devices pass the Hamming Weight Test**.

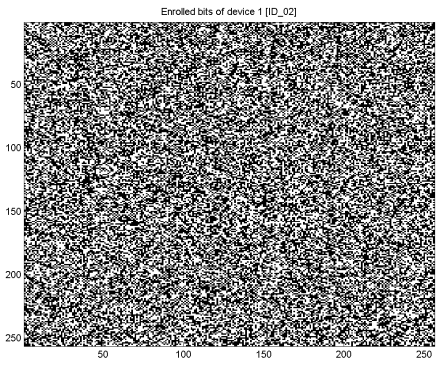

Figure 2.3.17: Example of SRAM PUF response from STM32F100R8 measurement.

2.3.5 ST STM32F100RB

Information

Number of devices measured:	11
Number of measurements per device:	51
PUF type:	SRAM PUF
PUF size:	8KB

Repeated Start-up Test

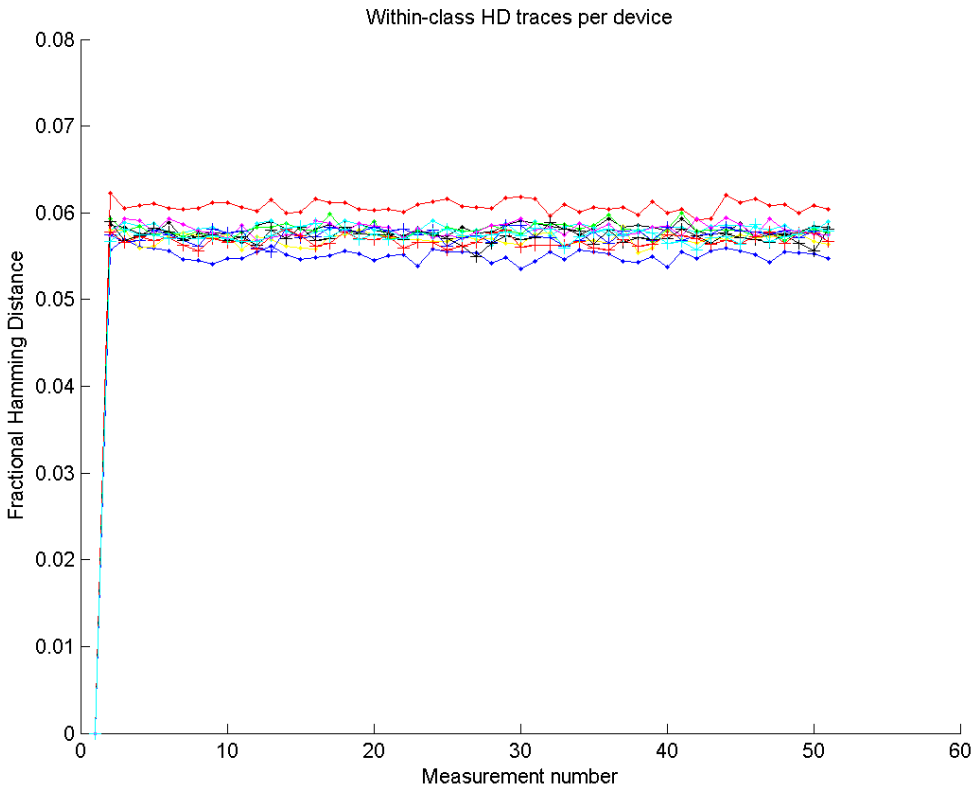

Figure 2.3.18: Within-class Hamming distance of SRAM in STM32F100RB measurements.

Figure 2.3.18 shows the Hamming distance from 51 measurements of the Repeated Start-up Test for the 11 ST STM32F100RB microcontrollers (each individual line representing one of the devices). For all devices, the first measurement is used for enrolment (so it has 0% noise); all other measurements are compared to this enrolment measurement. The maximum within-class Hamming distance for these devices (at room temperature) is less than 7%. This amount of noise can easily be corrected using commonly known Fuzzy Extractors. Therefore, **these devices pass the Repeated Start-up Test**.

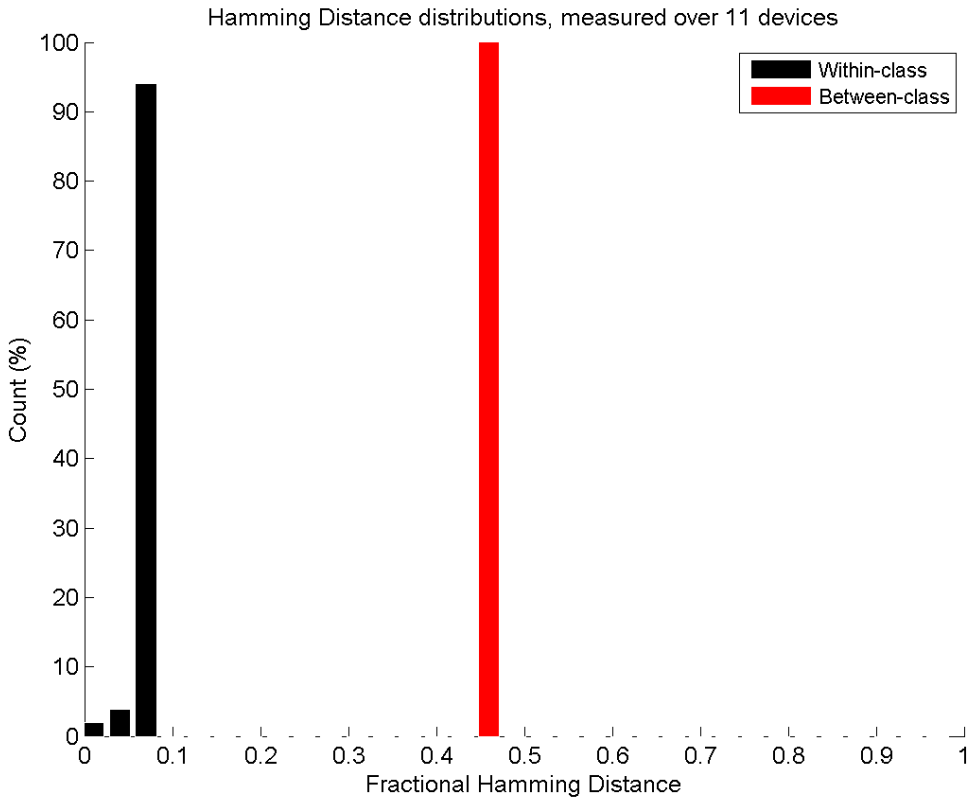

Figure 2.3.19: Between-class versus within-class Hamming distance of SRAM in STM32F100RB measurements.

Between-Class Hamming Distance Test

Figure 2.3.19 compares the results from the Repeated Start-up Test (in black) to the results of the Between-Class Hamming Distance Test (in red) for the 11 ST STM32F100RB microcontrollers. The Hamming distance between different devices is much higher than the noise measured for each individual device. This indicates that the devices can all be uniquely identified based on their PUF responses.

In more detail: We calculated $11 \times 10 : 2 = 55$ Hamming distances between the 11 devices. These 55 fractional distances fit a Gaussian distribution with mean value of 46.8%. This indicates that there is some correlation between the PUF responses from different devices, but they should still be suitable as an input for commonly known Fuzzy Extractors. Therefore, **these devices (weakly) pass the Between-Class Hamming Distance Test**.

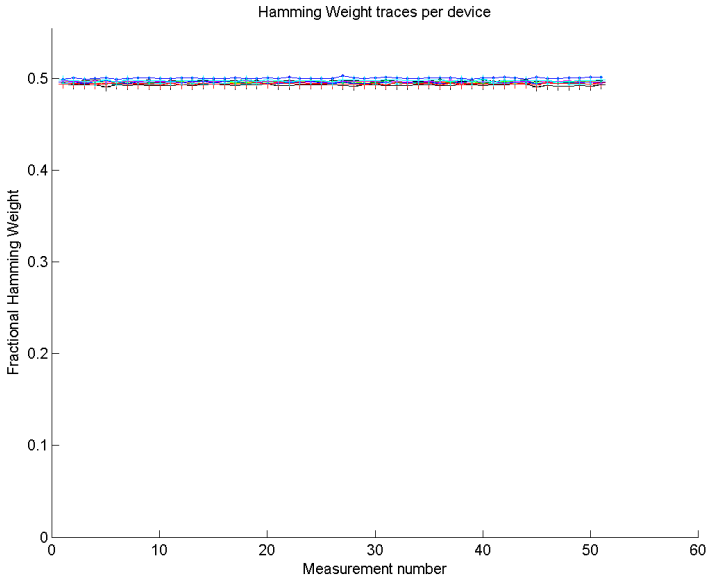

Figure 2.3.20: Hamming weight of SRAM in STM32F100RB measurements.

Hamming Weight Test

Figure 2.3.20 shows the Hamming weight of the measurements from the Repeated Start-up Test for the 11 ST STM32F100RB microcontrollers (each individual line representing one of the devices). For all devices, the Hamming weight of the measurements is close to 50%, i.e., there is an equal number of 0's and 1's in the PUF responses (an example PUF measurement is shown in Figure 2.3.21). This indicates that these PUF responses are suitable inputs for commonly known Fuzzy Extractors. Therefore, **these devices pass the Hamming Weight Test**.

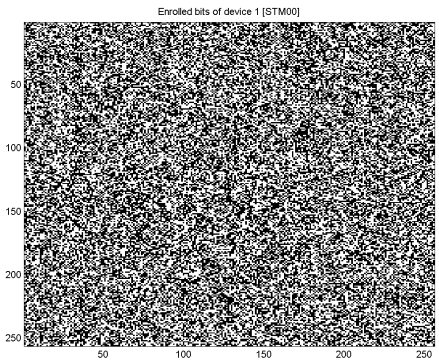

Figure 2.3.21: Example of SRAM PUF response from STM32F100RB measurement.

2.3.6 Atmel ATMega328p

Information

Number of devices measured:	16
Number of measurements per device:	50
PUF type:	SRAM PUF
PUF size:	2KB

Repeated Start-up Test

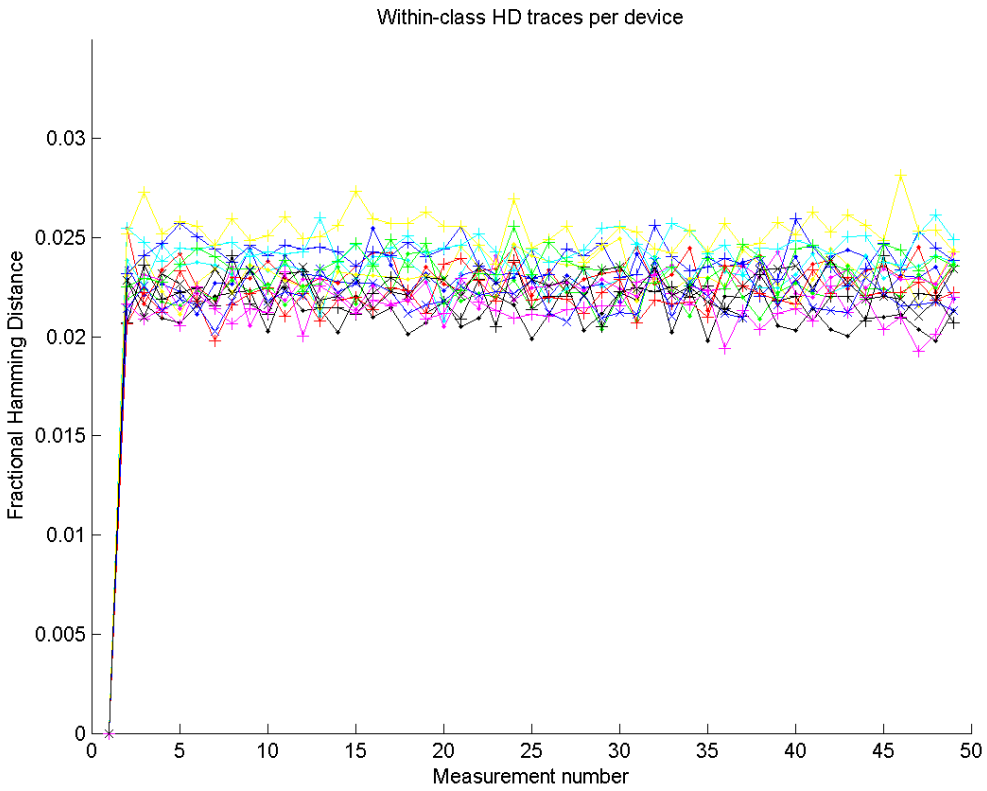

Figure 2.3.22: Within-class Hamming distance of SRAM in ATMega328p measurements.

Figure 2.3.22 shows 50 measurements of the Repeated Start-up Test for the 16 ATMega328p microcontrollers (each individual line representing one of the devices). For all devices, the first measurement has been used for enrolment (so it has 0% noise). All other measurements are compared to the enrolment measurement. The maximum within-class Hamming distance for these devices (at room temperature) is less than 3%. This amount of noise can easily be corrected using commonly known Fuzzy Extractors. Therefore, **these devices pass the Repeated Start-up Test**.

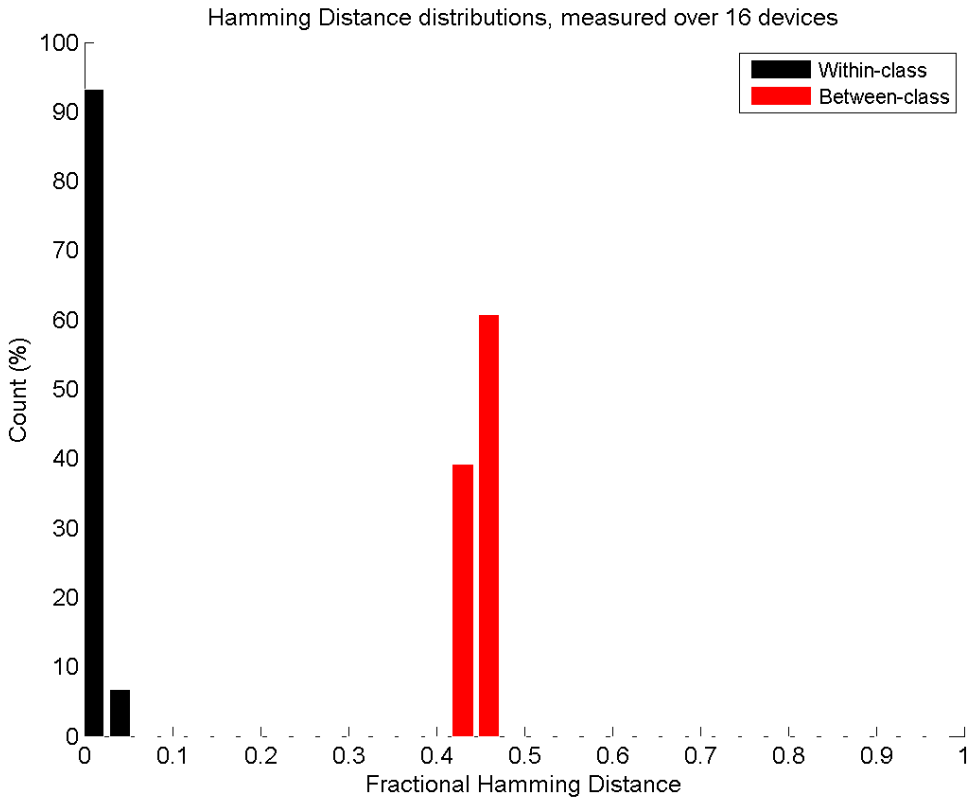

Figure 2.3.23: Between-class versus within-class Hamming distance of SRAM in ATMega328p measurements.

Between-Class Hamming Distance Test

Figure 2.3.23 compares the results from the Repeated Start-up Test (in black) to the results of the Between-Class Hamming Distance Test (in red) for the 16 ATMega328p microcontrollers. The Hamming distance between different devices is much higher than the noise measured for each individual device. This indicates that the devices can all be uniquely identified based on their PUF responses.

In more detail: We calculated $16 \times 15 : 2 = 120$ Hamming distances between the 16 devices. These 120 distances fit a fractional Gaussian distribution with a mean value of 44.7%. This is an indication that there is some correlation between the PUF responses from different devices, but they should still be suitable as an input for commonly known Fuzzy Extractors. Therefore, **these devices (weakly) pass the Between-Class Hamming Distance Test**.

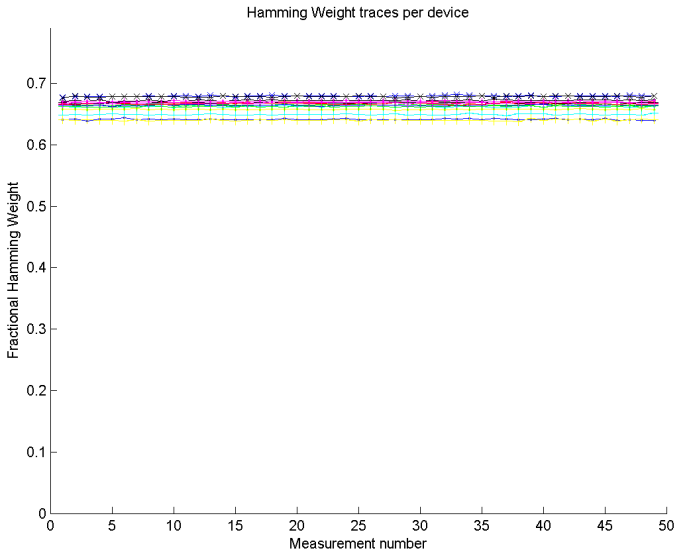

Figure 2.3.24: Hamming weight of SRAM in ATMega328p measurements.

Hamming Weight Test

Figure 2.3.24 shows the Hamming weight from the 50 measurements of the Repeated Start-up Test used for the 16 ATMega328p microcontrollers (each individual line representing one of the devices). For all devices, the Hamming weight of the measurements is significantly higher than 50% (around 65%), i.e., there are more 1's than 0's in the PUF responses (an example PUF response is shown in Figure 2.3.25). This indicates that these PUF responses require some pre-processing in order to be suitable inputs for commonly known Fuzzy Extractors. This causes some overhead in the size requirements for the PUF response; however, these requirements can most likely be fulfilled. Therefore, **these devices (weakly) pass the Hamming Weight Test**.

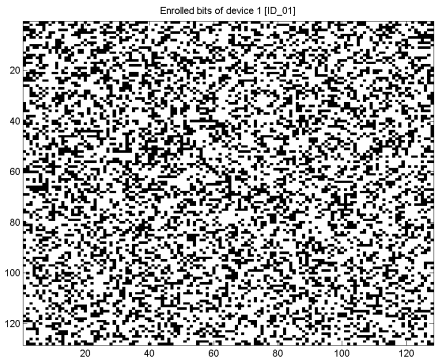

Figure 2.3.25: Example of SRAM PUF response from ATMega328p measurement.

2.3.7 NVIDIA GeForce GTX 295

Information

Number of devices measured:	4
Number of measurements per device:	12
PUF type:	SRAM PUF
PUF size:	7680 bits

Repeated Start-up Test

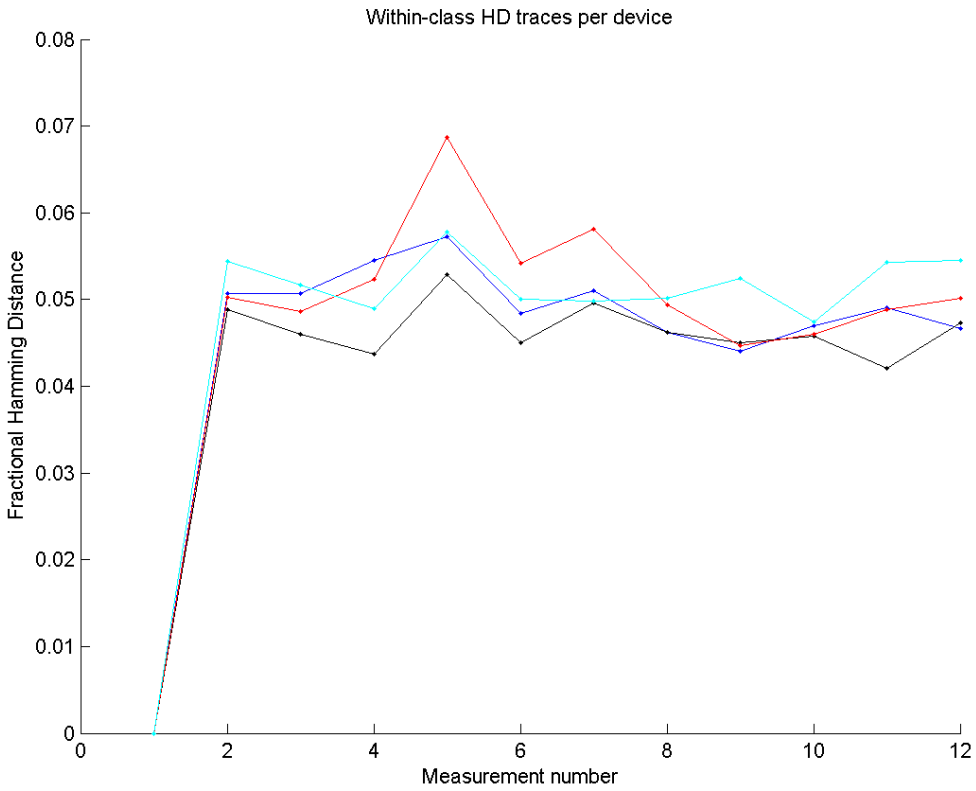

Figure 2.3.26: Within-class Hamming distance of SRAM in GTX 295 measurements.

Figure 2.3.26 shows the Hamming distances of 12 measurements of the Repeated Start-up Test on the 4 GTX 295 GPUs (each individual line representing one of the devices). The first measurement has been used for enrolment (so it has 0% noise); all other measurements are compared to this enrolment measurement. The maximum within-class Hamming distance for these devices (at room temperature) is less than 7%. This amount of noise can easily be corrected using commonly known Fuzzy Extractors. Therefore, **these devices pass the Repeated Start-up Test**.

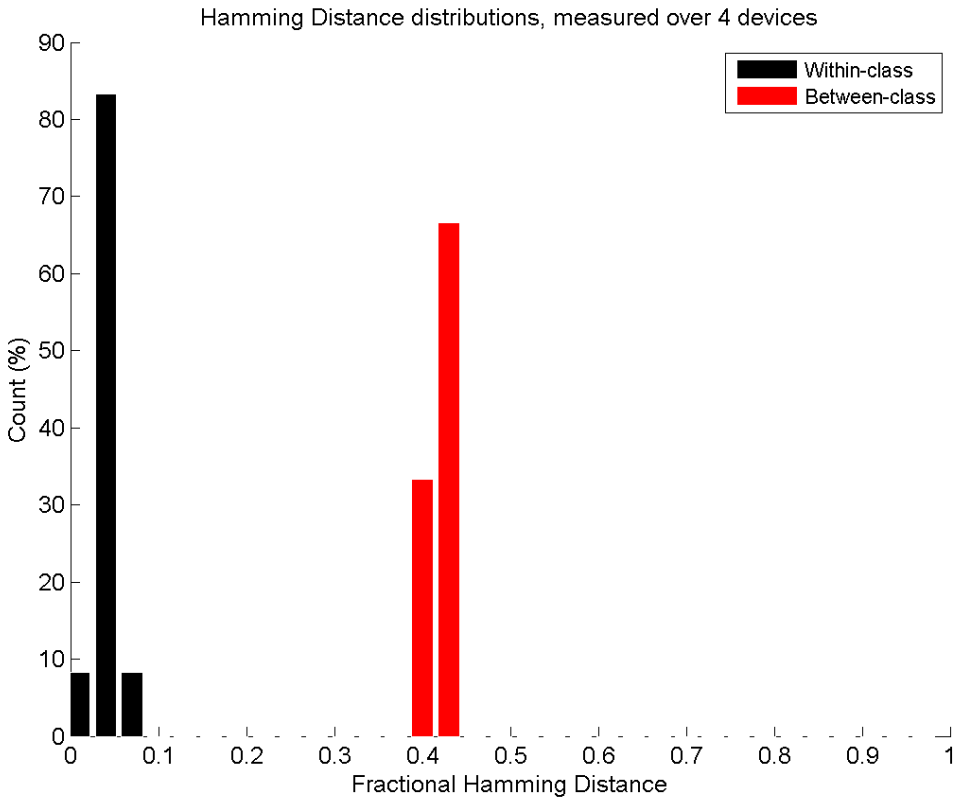

Figure 2.3.27: Between-class versus within-class Hamming distance of SRAM in GTX 295 measurements.

Between-Class Hamming Distance Test

Figure 2.3.27 compares the results from the Repeated Start-up Test (in black) to the results of the Between-Class Hamming Distance Test (in red) for the 4 GTX 295 GPUs. The Hamming distance between different devices is much higher than the noise measured for each individual device. This indicates that the devices can all be uniquely identified based on their PUF responses.

In more detail: We calculated $4 \times 3 : 2 = 6$ Hamming distances between the 4 devices. These 4 distances cannot be fitted by a Gaussian distribution, because the number of values is not large enough for a proper Gaussian fit. The distances between the devices vary from 39.8% to 44.0%. This is an indication that there is some correlation between the PUF responses from different devices, but they should still be suitable as an input for commonly known Fuzzy Extractors. Therefore, **these devices (weakly) pass the Between-Class Hamming Distance Test**.

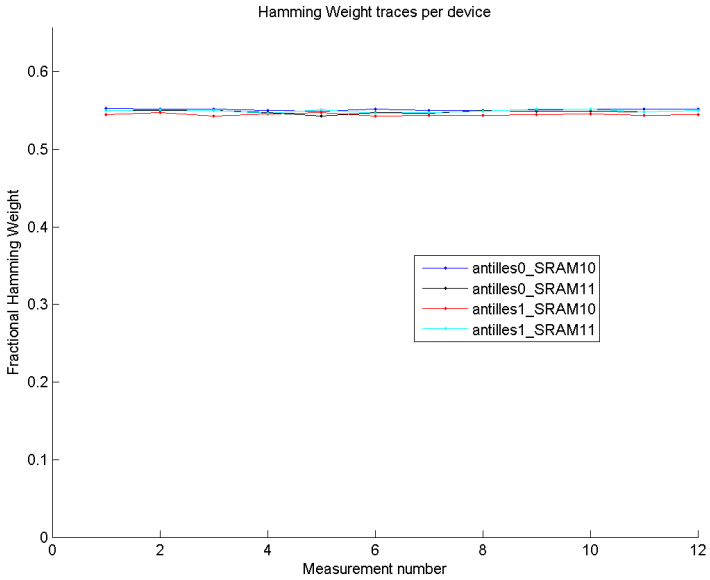

Figure 2.3.28: Hamming weight of SRAM in GTX 295 measurements.

Hamming Weight Test

Figure 2.3.28 shows the Hamming weight of the 12 measurements from the Repeated Start-up Test for the 4 GTX 295 GPUs (each individual line representing one of the devices). For all devices, the fractional Hamming weight of the measurements is higher than 50% (around 55%), i.e., there are more 1's than 0's in the PUF responses (an example PUF response is shown in Figure 2.3.29). This indicates that these PUF responses require some pre-processing in order to be suitable inputs for commonly known Fuzzy Extractors. This causes some overhead in the size requirements for the PUF response; however, these requirements can most likely be fulfilled. Therefore, **these devices (weakly) pass the Hamming Weight Test**.

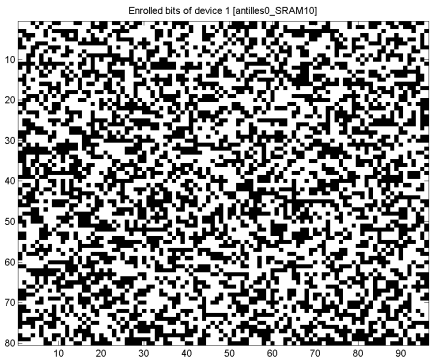

Figure 2.3.29: Example of SRAM PUF response from GTX 295 measurement.

2.3.8 Pandaboard

Information

Number of devices measured:	5
Number of measurements per device:	1000
PUF type:	SRAM PUF
PUF size:	16KB

Repeated Start-up Test

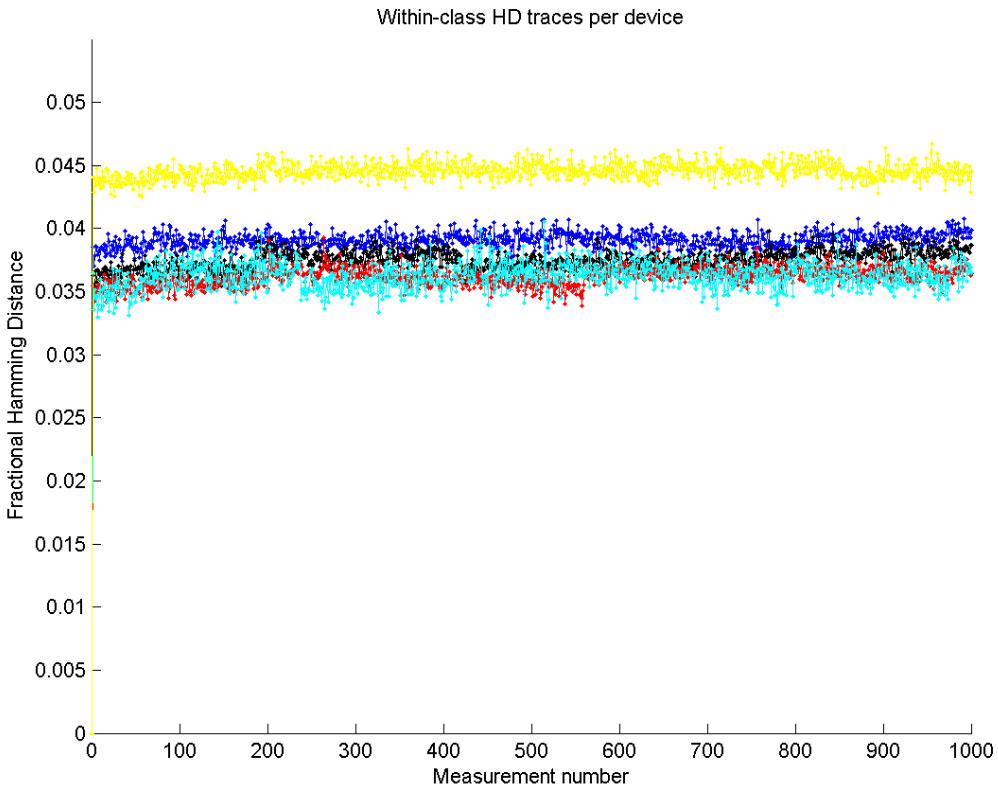

Figure 2.3.30: Within-class Hamming distance of SRAM in Pandaboard measurements.

Figure 2.3.30 shows the Hamming distances from the 1000 measurements of the Repeated Start-up Test for the 5 Pandaboards (each individual line representing one of the devices). The first measurement of each device has been used for enrolment (so it has 0% noise); all other measurements are compared to this enrolment measurement. The maximum within-class Hamming distance for these devices (at room temperature) is less than 5%. This amount of noise can easily be corrected using commonly known Fuzzy Extractors. Therefore, **these devices pass the Repeated Start-up Test**.

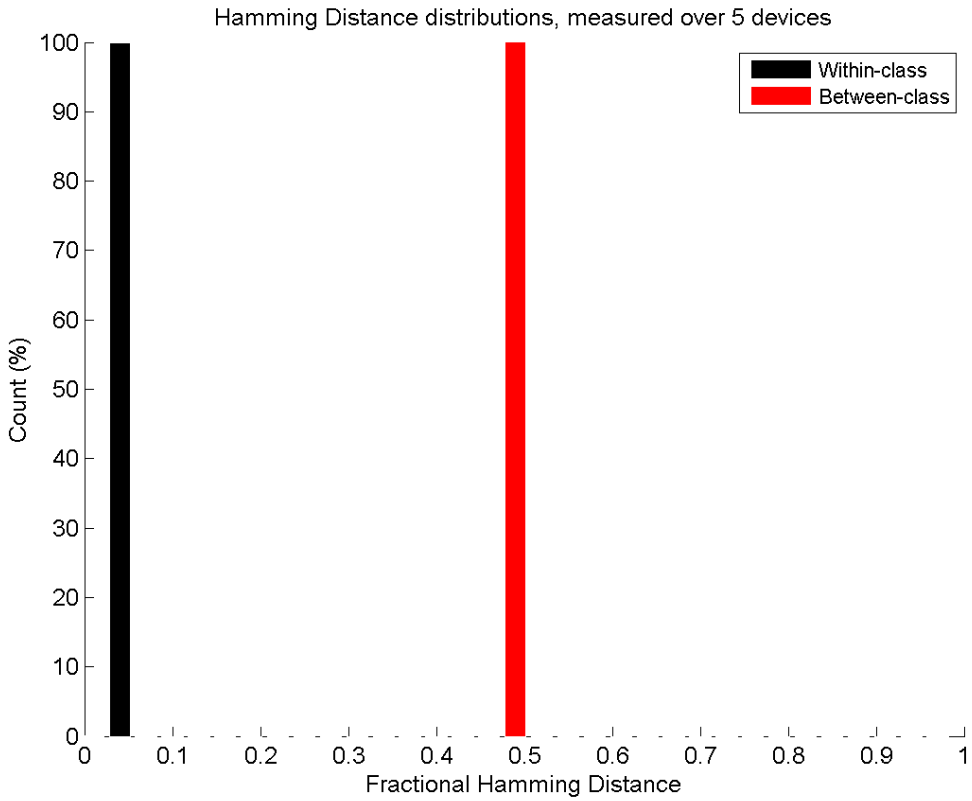

Figure 2.3.31: Between-class versus within-class Hamming distance of SRAM in Pandaboard measurements.

Between-Class Hamming Distance Test

Figure 2.3.31 compares the results from the Repeated Start-up Test (in black) to the results of the Between-Class Hamming Distance Test (in red) for the Pandaboards. The Hamming distance between different devices is much higher than the noise measured for each individual device. This indicates that the devices can all be uniquely identified based on their PUF responses.

In more detail: We calculated $5 \times 4 : 2 = 10$ Hamming distances between the 5 devices. These 10 distances cannot be fitted by a Gaussian distribution, because the number of values is not big enough for a proper Gaussian fit. The distances between the devices vary from 49.7% to 50.1% which shows that there is almost no correlation between the PUF responses from different devices. This indicates that they are suitable inputs for commonly known Fuzzy Extractors. Therefore, **these devices pass the Between-Class Hamming Distance Test**.

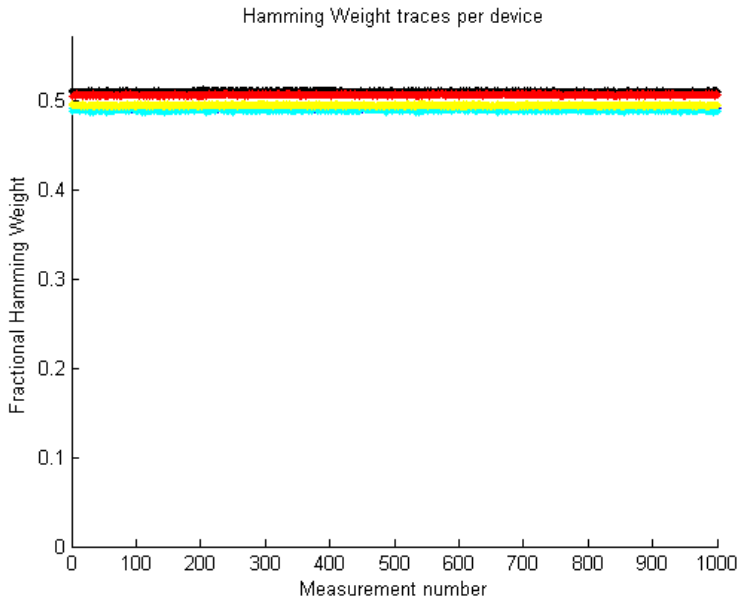

Figure 2.3.32: Hamming weight of SRAM in Pandaboard measurements.

Hamming Weight Test

Figure 2.3.32 shows the Hamming weight of the measurements from the Repeated Start-up Test for the 5 Pandaboards (each individual line representing one of the devices). The fractional Hamming weight of the measurements from most of the devices is close to 50% (although it varies slightly between devices), i.e., there is an equal number of 0's and 1's in the PUF responses (Figure 2.3.33 shows a plot of an example PUF measurement). This indicates that these PUF responses are suitable inputs for commonly known Fuzzy Extractors. Therefore, **these devices pass the Hamming Weight Test**.

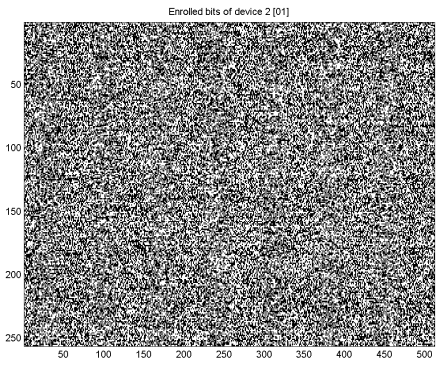

Figure 2.3.33: Example of SRAM PUF response from Pandaboard measurement.

2.4 Conclusions

Platform	Quantity	RST	TCT	BCHDT	HWT
Ainol Novo 7	7	Pass	n.a.	Pass	Pass
MSP430F5308	15	Pass	Pass	(Weak) Pass	(Weak) Pass
PIC16F1825	16	Pass	n.a.	**Fail**	**Fail**
STM32F100R8	11	Pass	n.a.	(Weak) Pass	Pass
STM32F100RB	11	Pass	n.a.	(Weak) Pass	Pass
ATMega328p	16	Pass	n.a.	(Weak) Pass	(Weak) Pass
GTX 295	4	Pass	n.a.	(Weak) Pass	(Weak) Pass
Pandaboard	5	Pass	n.a.	Pass	Pass

Table 2.4.1: Test results for the different devices

Table 2.4.1 gives an overview of the test results from the previous section. The most important conclusion that can be drawn from these results is that PUF behaviour can be found in the SRAMs of many different commercially available platforms. Most of the SRAMs that have been measured show promising results and therefore are suitable for use in PUF implementations; however, the amount of pre-processing required on the data will vary between the platforms.

From the platforms that we have investigated up to now, the PIC16F1825 microcontroller appears to be the only one where the SRAM measurements are not usable for PUF applications. Due to severe (bytewise) biasing of the PUF responses, these SRAMs do not provide enough entropy/uniqueness to be the basis for a proper PUF implementation. Based on our practical experience, the most common reason for biased start-up patterns is due to the supply voltage ramp-up curve on the SRAM cells. Unfortunately, we can only measure the curve on the external pins of the devices and we do not know what happens internally with the supply voltage before it reaches the SRAM. It is possible that there are (analog) components connected to the power supply, which distort the ramp-up on the SRAM. Microchip does not provide information about their silicon implementation, which makes it impossible for us to investigate what is happening inside the devices.

Out of the platforms that do pass the described tests, the Pandaboard might be the most interesting candidate for implementing prototypes during a later phase of the PUFFIN project. This computer development platform offers a completely open-source Linux/Android development environment, including the possibility to modify its bootloader. This enables us to obtain PUF data from the SRAM for use even during the boot sequence of the board. At this moment we do not have very much statistics (yet) on the PUF behaviour of the SRAM of this board, because we only have access to 5 boards. However, the results from these 5 boards do seem promising. During the next phase of the PUFFIN project the possibilities of this board will be studied more thoroughly.

All results from this WP2 work have been communicated to WP3 such that the obtained results can be used as input for the use-case implementations that will be developed in WP3.

Chapter 3

New methods for PUF analysis

Besides analysing the PUF measurements from WP1, work in WP2 has also been focussed on developing new methods for analysing PUF data. For this purpose two new analysis methodologies have been developed during the first phase of the PUFFIN project.

The first new method focusses on analysing PUF reliability. The work introduces a new reliability model taking an observed heterogeneous nature of PUF cells into account. A substantial experimental validation has demonstrated that the new predicted distributions from this model describe the empirically observed data statistics almost perfectly, even considering sensitivity to operational temperature. This will allow to study PUF failure behaviour in full detail, including the average and the worst case probabilities.

Besides for reliability, also a new methodology for evaluating the uniqueness of PUFs has been developed within PUFFIN. The aim of this work was to develop and implement a new methodology for accurately estimating the entropy of PUFs. This novel method estimates the extractable entropy by calculating the mutual information between enrolment and reconstruction measurements.

Two papers have been written about these two new methodologies for publication:

- "An Accurate Probabilistic Reliability Model for Silicon PUFs", Roel Maes (Intrinsic-ID). Published at Workshop on Cryptographic Hardware and Embedded Systems 2013 (CHES 2013).
- "Bias-based modeling and entropy analysis of PUFs", Robbert van den Berg (Eindhoven University of Technology), Boris Škorić (Eindhoven University of Technology), and Vincent van der Leest (Intrinsic-ID). Accepted for publication at International Workshop on Trustworthy Embedded Devices 2013 (TrustED 2013).

Both of these papers have used PUF data from the FP7 project UNIQUE (contract number: 238811) for evaluating the performance of their proposed methodologies. Reason for this is the fact that the UNIQUE database contains the biggest set of PUF data currently available, which makes the evaluation statistically relevant. The dataset as gathered by PUFFIN is currently simply not sufficient for this purpose yet.

To provide a complete overview of the work that has been performed on (as well as the results from) developing the new analysis methodologies, the above mentioned papers have been attached to this deliverable as appendices.

Appendix A

Paper: "An Accurate Probabilistic Reliability Model for Silicon PUFs"

Author: Roel Maes (Intrinsic-ID)
Venue: Workshop on Cryptographic Hardware and Embedded Systems (CHES) 2013
Date: August 18th - August 22nd, 2013

Status: Published

An Accurate Probabilistic Reliability Model for Silicon PUFs *

Roel Maes

Intrinsic-ID, Eindhoven, the Netherlands
roel.maes@intrinsic-id.com

Abstract. The power of an accurate model for describing a physical process or designing a physical system is beyond doubt. The currently used reliability model for physically unclonable functions (PUFs) assumes an equally likely error for every evaluation of every PUF response bit. This limits an accurate description since experiments show that certain responses are more error-prone than others, but this *fixed error rate model* only captures *average case* behavior. We introduce a new PUF reliability model taking this observed heterogeneous nature of PUF cells into account. A substantial experimental validation demonstrates that the new predicted distributions describe the empirically observed data statistics almost perfectly, even considering sensitivity to operational temperature. This allows to study PUF failure behavior in full detail, including the average and the *worst case* probabilities. This is an invaluable tool for the future design of more efficient and better adapted PUFs and PUF-based systems.

1 Introduction

After a decade of ongoing scientific research and sustained technical development, silicon PUF technology [1,2] is steadily finding its way into electronic products [3,4]. To meet the high reliability and security constraints imposed by such applications, bare silicon PUFs don't operate on their own but are embedded in a system. The fundamental physical security of such a system originates from the PUF implementation, but considerable post-processing is involved to meet the overall requirements and facilitate the intended application, e.g. key storage. Constructing a PUF system is an intricate design exercise since it requires balancing typically opposing goals between reliability, security and efficiency.

The starting point of a PUF system design is evidently the probabilistic behavior of the PUF itself, both regarding reliability (*error behavior*) and security (*unpredictability behavior*). The more insight one has in these

*This work has been supported by the European Commission through the ICT program under contract INFSO-ICT-284833 (PUFFIN) and by the EUREKA Network through the CATRENE program under contract CA-403 (RELY).

details, the better one is able to fine tune design choices, and the more confidence one has in the obtained results. To consistently deal with a PUF's probabilistic behavior, an accurate model which closely fits empirical statistics is of great importance. Such a model should be sufficiently generic to confidently extrapolate predictions to unobserved points and to work with a variety of PUF constructions. When available, it is an indispensable tool for analyzing the design space of a PUF system and converging on an optimized solution. The main focus of this work is the development and analysis of a more accurate and generic *reliability model* for silicon PUFs than the one in use today, and a demonstration of its advantages.

Related Work. The commonly used PUF reliability model, e.g. in [2,5,6,7,8,9] and many others, is that of a *fixed error rate*, i.e. each evaluation of each response bit is assumed equally likely to be wrong. Many cell-individual details are lost by reducing the reliability behavior to a single average-case parameter. A first extension of this model, e.g. as used in [10,11,12], is the binary differentiation between *stable* and *unstable* PUF response bits. This idea is generalized in [13] which demonstrates that PUF cell reliabilities are continuously distributed, from very unreliable to almost perfectly stable.

Contributions. In this work, we start from and greatly expand on the model as proposed in [13], to describe PUF reliability behavior in a much more accurate and detailed manner as has been done up to now. The basic model from [13] is modified to more realistically describe error-behavior, and extended to take environmental dependencies like temperature into account. This new model is extensively validated on reliability data from measurements of PUFs implemented in 65nm CMOS. The fit between predicted distributions and empirical statistics is strikingly accurate at all measured temperatures from $-40°$C to $+85°$C. Moreover, the model proves to be very generic by being extremely accurate for different types of memory-based PUF types, like the SRAM PUF [2], the buskeeper PUF [14] and the D Flip-flop PUF [15], as well as for the delay-based arbiter PUF [16]. We also demonstrate the gained insight offered by such an accurate model, by analyzing the implications for key generation. This clearly shows the limitations of the old fixed error rate model, and the added value of designing a PUF system using the new model.

Overview. Sect. 2 introduces the newly proposed model, motivates the assumed relations, and derives the hypothesized distribution functions. The model's accuracy is consequently validated in Sect. 3 by fitting it on empirical data from actual silicon PUF measurements. The gained

insights of the new model and their consequences for PUFs and PUF-based applications are discussed in Sect. 4. Finally, we identify the potential for future work based on these findings and conclude in Sect. 5.

2 Model Description

2.1 Notation and Preliminaries

Without loss of generality we consider silicon PUFs with single-bit responses. For the sake of clarity, the presented model is introduced in terms of memory-based PUFs, where each bit is produced by an individual (memory) *PUF cell*.[1] However, as demonstrated, the applicability of the model is certainly not limited to memory-based PUFs, but is also particularly accurate in describing the reliability behavior of delay-based silicon PUFs.

Variable Notation. Most of the model's variables are random in nature. We distinguish between random values sampled *once* for a particular PUF cell i (upon creation) and remain fixed for the cell's entire lifetime, which are denoted with subscript indexing (m_i), and others which are resampled every time the cell is evaluated, which are denoted with superscript indexing ($n_i^{(j)}$ for evalutation j of cell i). Random variables in general are denoted as capital literals, e.g. M is the random variable which is sampled to a value m_i for cell i, according to the distribution of M.

Distribution Functions. The distribution of a random variable X is characterized by its probability density function ($\mathbf{pdf}_X(x)$) and/or its cumulative distribution function ($\mathbf{cdf}_X(x)$). For discrete random variables, the probability density function degenerates to a probability mass function ($\mathbf{pmf}_X(x)$). Two basic distributions used in this work are the (standard) normal distribution ($\mathbf{pdf}_X(x) = \varphi(x)$ and $\mathbf{cdf}_X(x) = \Phi(x)$) and the binomial distribution ($\mathbf{pmf}_X(x) = f_{\text{bino}}(x; n, p)$ and $\mathbf{cdf}_X(x) = F_{\text{bino}}(x; n, p)$). We refer to App. A for details on these distributions.

2.2 The "Old" Model: PUF Response with Fixed Error Rate

We first briefly discuss the probabilistic model which is thus far used in the majority of related literature (e.g. in [2,5,6,7,8,9]) for assessing the reliability of PUFs and their applications.

[1] We refer to the literature on memory-based PUFs and silicon PUFs in general for more details on their operation and implementation. See e.g. [17] for an overview.

Rationale. The foundation of the old model is the assumption that all cells of a PUF are *homogeneous*, i.e. every cell in the PUF is equally likely to produce an error at any time. This means the reliability behavior of the PUF as a whole is described by a single fixed parameter: the *(bit) error rate* (p_e). This is the probability that *any* evaluation of *any* cell differs from its enrolled response, and is assumed equal to the average-case behavior averaged over many cells.

Limitations. Though convenient to use, this model's limitations are evident when looking at experimental PUF results. A typical PUF instantiation exhibits *unstable* and *stable* cells, i.e. some cells are more likely to produce an error while other cells are hardly ever wrong. This behavior is not captured by the old model which treats every cell in the same way. However, as shown in Sect. 4, it is wise to take this observation into account when designing PUF-based applications. The main motivation behind the newly introduced model is to accurately capture this cell-specific behavior.

2.3 The "New" Model: Cell-Specific Error-Probabilities

In line with the experimental observation that some PUF cells are more error-prone than others, the foundation of the new model lies in the assumed cell *heterogenity*, i.c. every cell in a PUF has an individual error-probability. An early form of this basic idea was introduced in [13] and serves as a starting point for the new model presented here.

Hidden Variable Model. The implied approach of [13], which we make explicit, is that of a *hidden variable model*. Basically, it is assumed that the *observable variables* of a PUF cell which describe its observable behavior, are governed by underlying *hidden variables*. By assuming plausible distributions for the hidden variables, the resulting distributions of the observable variables are derived and validated against experimental data.

The Observable Variables describe the probabilistic behavior of an evaluation (j) of a PUF cell i to a response bit value $r_i^{(j)} \in \{0, 1\}$ (a random sampling of R_i):

- *The One-Probability* (p_i) of a cell i is the probability that it returns '1' upon a random evaluation: $p_i \stackrel{\text{def}}{=} \mathbf{Pr}(R_i = 1)$. The one-probability is itself a random variable P randomly sampled to a value $p_i \in (0, 1)$ for a cell i.

– *The Error-Probability* ($p_{e,i}$) of a cell i is the probability that a random evalutation differs from an earlier recorded evaluation of that cell during an *enrollment phase*[2]: $p_{e,i} \stackrel{\text{def}}{=} \mathbf{Pr}\left(R_i \neq r_i^{\text{enroll}}\right)$. The error-probability is itself a random variable P_e randomly sampled to a value $p_{e,i} \in (0,1)$.

The Hidden Variables are abstractions of underlying physical (electrical) processes in a silicon PUF cell circuit. We do not consider low-level physical details explicitly to avoid complex simulations and to maintain a generic model. The used hidden variables are regarded as generic and approximated *lumped* versions of underlying measurable physical quantities:

– *The Process Variable* (m_i) quantifies the accumulated effect of process variations on a cell's internals, introduced during manufacturing. This is a random variable (M), sampled at a cell's creation time, according to a distribution determined by the manufacturing process.
– *The Noise Variable* ($n_i^{(j)}$) quantifies the accumulated effect of random noise on a cell's internals during evaluation. This is a random variable (N_i), resampled for every evaluation of the cell, according to a distribution determined by the cell's susceptibility to noise.

The Model Relation is the fundamental connection between hidden and observable variables, from which all further conclusions are derived:

$$\boxed{r_i^{(j)} = \begin{cases} 0 \text{ , if } m_i + n_i^{(j)} \leq t \,, \\ 1 \text{ , if } m_i + n_i^{(j)} > t \,. \end{cases}} \tag{1}$$

The implied assumptions of this relation are: *i)* that the hidden variables are *additive*,[3] and *ii)* that the evaluation outcome is the result of a comparison with a constant *threshold parameter* t. The relation for the one-probability is directly derived from (1) as: $p_i = \mathbf{Pr}\left(m_i + N_i > t\right) = 1 - \mathbf{cdf}_{N_i}\left(t - m_i\right)$.

Distributions of the New Model. Since both hidden variables are considered lumped physical quantities, a normal distribution is a motivated assumption for both: $M \sim \mathcal{N}(\mu_M, \sigma_M^2)$, and $N_i \sim \mathcal{N}(0, \sigma_N^2)$. For ease of notation, the parameters $\lambda_1 = {}^{\sigma_N}\!/_{\sigma_M}$, and $\lambda_2 = {}^{(t\,-\,\mu_M)}\!/_{\sigma_M}$ are used.

[2] In [13], error-probability is defined with respect to a cell's *most-likely* outcome which is not representative for the realistic use of a PUF. Therefore, we consider a random enrollment instead: r_i^{enroll} is randomly sampled according to the one-probability p_i, and can (coincidentally) be an unlikely outcome for the considered cell

[3] This is intuitively justified by considering that the hidden variables are of an *electrical* nature, i.e. voltages or currents. Additivity then follows from Kirchoff's laws.

Based on these assumed distributions, the resulting observable variable distributions are derived by employing the model relation as expressed in (1). The one-probability distribution was already derived in [13]:[4]

$$\mathbf{cdf}_P(x) = \Phi\left(\lambda_1 \Phi^{-1}(x) + \lambda_2\right). \tag{2}$$

The detailed derivation of the new error-probability distribution is presented in App. B.1 and results in:[5]

$$\mathbf{cdf}_{P_e}(x) = \lambda_1 \cdot \int_{-\infty}^{\Phi^{-1}(x)} \Phi(-u) \cdot \left(\varphi(\lambda_1 u + \lambda_2) + \varphi(\lambda_1 u - \lambda_2)\right) \mathrm{d}u\,. \tag{3}$$

2.4 Modeling Temperature Dependence

From many PUF experiments (e.g. in [18]) it is clear that the operating conditions of a silicon PUF, such as temperature and voltage, have a noticeable impact on response behavior. At increasingly different conditions this even becomes the primary source of unreliability, much more so than instantaneous random noise. To realistically describe a PUF cell's error-behavior we incorporate these effects in the new model. This is done for *temperature*, which typically has the largest impact on PUF reliability [18].[6]

Hidden Variable Model: Temperature Extension. The basic hidden variable model from Sect. 2.3 is extended with a new hidden variable quantifying a cell's sensitivity to temperature: the *temperature dependence* (d_i). Since different cells react differently to temperature changes, this is a cell-specific value randomly sampled at manufacturing time. The observable variables are straightforwardly extended to express temperature dependence: $p_i(T) = \mathbf{Pr}\left(R_i(T) = 1\right)$ and $p_{e,i}(T; T_{ref}) = \mathbf{Pr}\left(R_i(T) \neq r_i^{\text{enroll}}(T_{ref})\right)$. Note that error-probability depends on two temperatures, at enrollment (T_{ref}) and at reconstruction (T).

The Temperature Model Relation extends the additive threshold relation of the new model as given by (1) with a temperature dependent term. This relation assumes a linear dependence on the (absolute) temperature with a cell-dependent sensitivity quantified by d_i:

$$\boxed{r_i^{(j)}(T) = \begin{cases} 0\,, \text{ if } m_i + n_i^{(j)} + d_i \cdot T \leq t\,, \\ 1\,, \text{ if } m_i + n_i^{(j)} + d_i \cdot T > t\,. \end{cases}} \tag{4}$$

[4] Since P and P_e are probabilities, $\mathbf{cdf}_P(x)$ and $\mathbf{cdf}_{P_e}(x)$ are only defined for $x \in (0, 1)$.

[5] This and following integral expressions are evaluated using numerical methods.

[6] Other conditions can be equivalently modelled but are omitted due to lack of space.

Distribution of the Temperature Model. For the temperature dependence variable we also assume a normal distribution: $D \sim \mathcal{N}(0, \sigma_D^2)$. A third model parameter is introduced as $\theta = {}^{\sigma_N}/_{\sigma_D}$. Following the temperature model relation expressed by (4), the distribution of the temperature-dependent error-probabilities becomes:

$$\mathbf{cdf}_{P_e(T;T_{ref})}(x) = \frac{\lambda_1 \theta}{|\Delta T|} \cdot \int_{-\infty}^{\Phi^{-1}(x)} \int_{-\infty}^{+\infty} \Big[\Phi(-u)\, \varphi\left(\theta \tfrac{v-u}{|\Delta T|}\right) + \Phi(u)\, \varphi\left(\theta \tfrac{v+u}{|\Delta T|}\right) \Big] \cdot \varphi(\lambda_1 u + \lambda_2)\, \mathrm{d}u\, \mathrm{d}v\,. \tag{5}$$

The complete derivation is given in App. B.2. We introduced $\Delta T = T - T_{ref}$, and (5) is only defined for $\Delta T \neq 0$. In case $T = T_{ref}$, the limiting case of (5) for $\Delta T \to 0$ reverts to (3).

3 Experimental Validation

We assess the validity of the assumptions made in Sect. 2 by fitting the predicted error-probability distribution to empirically observed statistics. For this purpose we use the extensive experimental PUF data set originating from the UNIQUE project [19], of which the initial analysis was presented in [18,20]. This data set was acquired from 192 ASICs manufactured in 65nm CMOS, each implementing six silicon PUF types. We applied our model in particular to the SRAM, D flip-flop, buskeeper and arbiter PUFs.

3.1 From Error-Probability to Error-Count

The error-probability of a particular PUF cell can be estimated by counting the number of errors in a number of cell evaluations and dividing it by that number. However, since the majority of cells typically has an error-probability very close to 0, this estimate is rather inaccurate when the number of evaluations is limited. E.g., based on 100 measurements of cell i which are all error-free, it is impossible to differentiate between $p_{e,i} = 10^{-3}$ or $p_{e,i} = 10^{-6}$ or even smaller. This inaccuracy hampers an accurate fit of the model, especially in the distribution tails (close to 0 and 1) which happen to be the most interesting parts. To overcome this problem we introduce a variable closely related to the error-probability but directly observable in experimental data without estimation accuracy problems: the *error-count* $s_{e,i}^{(n)}$ is the number of evaluations in n measurements of cell i which differ from an enrollment response bit for that cell. By consequence,

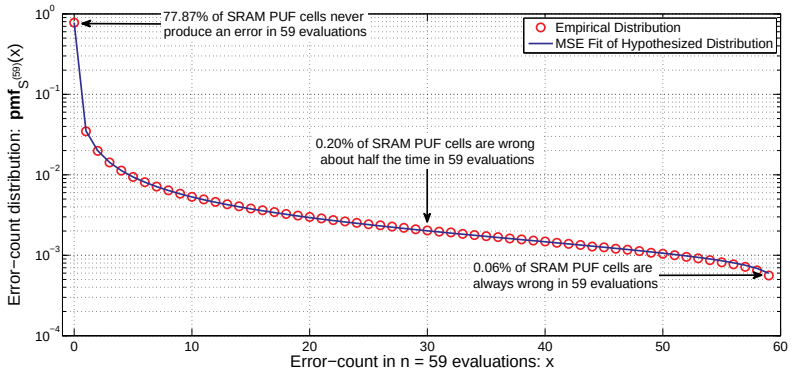

Fig. 1. Fit of $\mathbf{pmf}_{S_e^{(59)}}(x)$ on empirical SRAM PUF data at 25°C.

the value of $s_{e,i}^{(n)}$ is also a random value sampled (at a given temperature T), according to the discrete distribution characterized by:

$$\mathbf{pmf}_{S_e^{(n)}(T;T_{ref})}(x) = \int_0^1 f_{\text{bino}}(x;n,u) \cdot \mathbf{pdf}_{P_e(T;T_{ref})}(u)\, \mathrm{d}u\,. \tag{6}$$

In this section, we focus on fitting this distribution to the error statistics of the experimental PUF data. The expression for $\mathbf{pdf}_{P_e(T;T_{ref})}(u)$ is obtained by differentiating (3) (if $T = T_{ref}$) or (5) (if $T \neq T_{ref}$) and is listed for completeness in App. B.2.

3.2 Fitting the Error-Count Distrution

Fitting (λ_1, λ_2) at $T_{ref} = 25°C$. The first experimental data set we use for fitting the parameters (λ_1, λ_2) consists of 60 evaluations of 65,536 cells from 768 identical but distinct SRAM PUF instantiations[7] at a fixed temperature of $T_{ref} = 25°C$. This totals to $768 \times 65{,}536 = 50{,}331{,}648$ distinct but assumably identically implemented SRAM PUF cells all evaluated 60 times. We randomly pick one enrollment response and 59 reconstruction evaluations from which we calculate the error-count $s_{e,i}^{(59)}$ for each PUF cell i with respect to its enrollment value. From these 50,331,648 randomly sampled error-count values the empirical distribution of $S_e^{(59)}$ is calculated. If the model from Sect. 2.3 is accurate, then the hypothesized distribution of $S_e^{(59)}$ as characterized by (6) should closely fit the empirical histogram. We perform a non-linear optimization over (λ_1, λ_2) using the Levenberg-Marquardt algorithm to minimize the mean

[7] The 768 SRAM PUFs are implemented on 192 ASICs, with 4 instances per chip.

Table 1. Fit results of $\mathbf{pmf}_{S_e^{(n)}}(x)$ on empirical data of different PUF types at 25°C.

PUF Type	Silicon PUF	MSE of fit	λ_1	λ_2
Memory-based	SRAM PUF	$4.467 \cdot 10^{-9}$	0.1213	0.0210
Memory-based	Buskeeper PUF	$5.760 \cdot 10^{-10}$	0.0929	0.0340
Memory-based	D Flip-flop PUF	$1.150 \cdot 10^{-9}$	0.0812	0.0381
Delay-based	Arbiter PUF	$1.843 \cdot 10^{-9}$	0.0676	0.0461

squared error (MSE) between the empirical and hypothesized probability mass functions. The result is shown in Fig. 1 and shows that the function from (6) yields a strikingly accurate fit. The closest fit was found for $\boldsymbol{(\lambda_1 = 0.1213\ ,\ \lambda_2 = 0.0210)}$ with an MSE of merely $4.467 \cdot 10^{-9}$.

To demonstrate the generic nature of the proposed model we also apply it to other silicon PUF types. We considered the experimental data of 60 evaluations of 8,192 cells from 384 instantiations, for each of the buskeeper, the D flip-flop and the arbiter PUF[8]. All fitting results are summarized in Table 1 and show that the best fit for each of these alternative PUF types is at least as accurate as that for the SRAM PUF. Remarkably, the model succeeds in accurately predicting the reliability distributions for both memory-based as well as delay-based PUFs.

Fitting θ for the SRAM PUF at $T = [-40°\text{C}, \ldots, +85°\text{C}]$. To validate the temperature dependence of the model as presented in Sect. 2.4, we use an experimental data set obtained from 65,536 cells from a limited set of 20 identical but distinct SRAM PUF instantiations, evaluated 100 times at thirteen temperatures between −40°C and 85°C. This gives a total set of $20 \times 65{,}536 = 1{,}310{,}720$ cells, for each of which we calculate the error count $s_{e,i}^{(100)}(T; T_{ref})$ at every measured temperature with respect to a randomly selected enrollment response at $T_{ref} = 25°\text{C}$. The accuracy of the temperature model is tested by fitting the hypothesized distribution of $S_e^{(100)}(T; 25°\text{C})$, as characterized by (6), to the empirical distribution of these 1,310,720 samples at every measured $T \neq T_{ref}$. We use the estimated parameter values for (λ_1, λ_2) from the previous experiment, and perform an optimization over the remaining parameter θ to minimize the average MSE between the empirical and hypothesized probability mass functions over all T. The results are shown in Fig. 2 and demonstrate an accurate fit at every considered temperature. A minimal average MSE of $1.643 \cdot 10^{-6}$ over all temperatures is obtained for $\boldsymbol{\theta = 45.0}$, with the largest deviation

[8]For the arbiter PUF, a "cell" refers to an evaluation with a random challenge.

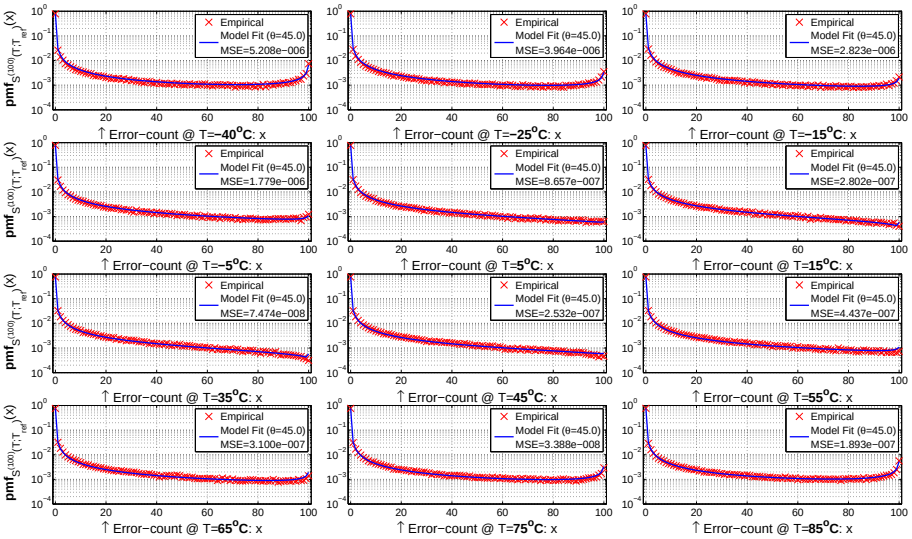

Fig. 2. Fit of $\mathbf{pmf}_{S_e^{(100)}(T;T_{ref}=25°\mathrm{C})}(x)$ on empirical SRAM PUF data for different T.

at the extreme temperature of $-40°\mathrm{C}$ (MSE of $5.208 \cdot 10^{-6}$). Given the single parameter linear temperature dependence assumed by the model, as given by (4), the fitted distributions are remarkably accurate.

4 Interpretation and Discussion

We are now able to quantify the consequences of the heterogenity of individual PUF cells. We first interpret the reliability distribution directly in Sect. 4.1 and study the effect on PUF-based key generation in Sect. 4.2.

4.1 Interpretation of the New Model Distributions

We consider the experimentally studied SRAM PUF from Sect. 3, with fitted model parameters: $(\lambda_1 = 0.1213, \lambda_2 = 0.0210, \theta = 45.0)$. The error-probability distribution is analysed at the worst-case temperature $T = -40°\mathrm{C}$ with respect to enrollment at $T_{ref} = 25°\mathrm{C}$. The cumulative distribution function is plotted in Fig. 3. From this graph the heterogene nature of the individual PUF cells is immediately clear. A remarkable observation is that about 34% of the SRAM PUF cells have an error-probability $\leq 10^{-15}$, i.e. in any practical setting they are always correct. On the other hand, about 7% of the cells produce an error in more than

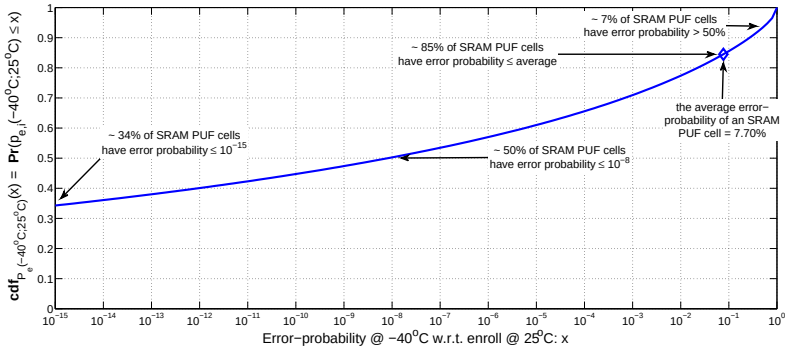

Fig. 3. Plot of $\mathbf{cdf}_{P_e\,(T=-40^\circ\mathrm{C};T_{ref}=25^\circ\mathrm{C})}\,(x)$ (Eq.(5)) with interpretation.

50% of their evaluations, and about 1% of the cells in more than 99%.[9] Another remarkable observation is the discrepancy between the *mean* error-probability, which is 7.70%, and the *median*, which is only in the order of 10^{-8}. The large majority of errors in a PUF response is hence caused by a small minority of cells which are wrong very often. This is exactly the kind of behavior which is oblivious in the fixed-error rate model (Sect. 2.2) and motivated us to develop a more accurate model (Sect. 2.3).

4.2 Implications for PUF-based Key Generators

Due to their appealing security properties like intrinisic uniqueness and physical unclonability, PUFs provide a strong physical foundation for secure key storage. To turn a PUF response into a secure key, post-processing is required by a *key generator* to boost the reliability and unpredictability to the cryptographically required level. For this purpose, a typical PUF-based key generator deploys a *fuzzy extractor* as introduced by [21], e.g. as implemented by [6,22,9,8]. For the analysis presented here, it suffices to consider a fuzzy extractor as a black box algorithm FE(n, t) which is able to correct up to t bit errors in an n-bit PUF response. We refer to the cited literature for in-depth details about a fuzzy extractor's operation.

From PUF Cell Error-Probabilities to Key Failure Rate. A key generation fails when the fuzzy extractor is unable to correct all the PUF response bit errors that simultaneously occur in a single evaluation. The *key*

[9] Cells with very high ($> 50\%$) error-probabilities are caused by a cell coincidentally assuming an unlikely value during enrollment, or alternatively because a cell's prefered value changes over the temperature shift between enrollment and reconstruction.

failure rate (p_{fail}) is the probability of this happening and hence becomes: $p_{\text{fail}} = \mathbf{Pr}\,(\#\text{ errors in } n \text{ response bits} > t)$, and should be very small for practical applications (typically 10^{-6} or 10^{-9}). With the fixed error-rate model (Sect. 2.2), as used in all literature on PUF-based key generators up to date, the number of errors in an n-bit response is binomially distributed. This results in a fixed failure rate for every key generator instantiation:

$$\text{(fixed error-rate)} \qquad p_{\text{fail}}(p_e) = 1 - F_{\text{bino}}\,(t; n, p_e)\ . \quad (7)$$

In the more accurate new model with random error-probabilities (Sect. 2.3), the number of errors in an n-bit PUF response is no longer binomially distributed, but *Poisson-binomially* distributed [23].[10] The Poisson-binomial cumulative distribution function $F_{\text{PB}}\,(t; \mathbf{p_e^n})$ is evaluated from the list of error-probabilities of n PUF cells: $\mathbf{p_e^n} = (p_{e,1}, p_{e,2}, \ldots, p_{e,n})$. The key failure rate for FE(n, t) then becomes:

$$\text{(random error-probabilities)} \qquad p_{\text{fail}}(\mathbf{p_e^n}) = 1 - F_{\text{PB}}\,(t; \mathbf{p_e^n})\ . \quad (8)$$

Since each of the elements of $\mathbf{p_e^n}$ is a randomly sampled variable, the resulting key failure rate will not be a fixed value for every generator, as in the old model, but also a randomly sampled value for each PUF instance.

The Key Failure Rate Distribution. We consider a key generator based on the SRAM PUF analysed in Sect. 4.1 (with worst-case reliability at -40°C) and a concatenated fuzzy extractor FE$(212, 11) \circ$ FE$(5, 2)$,[11] which extracts a 128-bit key from 1,060 cells, with $p_{\text{fail}} \leq 10^{-9}$ (on average).

Under the old fixed error-rate model of Sect. 2.2, the constant error rate is set equal to the mean error-probability over all cells: $p_e = 7.70\%$. The achieved average key failure rate is calculated by applying (7) twice:

$$p_{\text{fail}} = 1 - F_{\text{bino}}\,(11; 212, 1 - F_{\text{bino}}\,(2; 5, 0.0770)) = 1.15 \cdot 10^{-10}\,.$$

This key generator hence produces a 128-bit key with $p_{\text{fail}} = 1.15 \cdot 10^{-10} \leq 10^{-9}$. However, due the considered fixed-error model, this only holds for the *average case* key generator. No statements can be made about the distribution of failure rates, e.g. it is unclear which fraction of key

[10] Some details on this lesser known distribution are given in App. A.

[11] Concatenated fuzzy extractors are typically more efficient than single large fuzzy extractors [6]. The second fuzzy extractor sees the failure rate of the output of the first one as the error probability of its input symbols. For completeness, we mention the error-correcting codes on which the used fuzzy extractors are based: FE$(5, 2)$ uses the $(5, 1)$-repetition code and FE$(212, 11)$ the $(212, 128)$-BCH code.

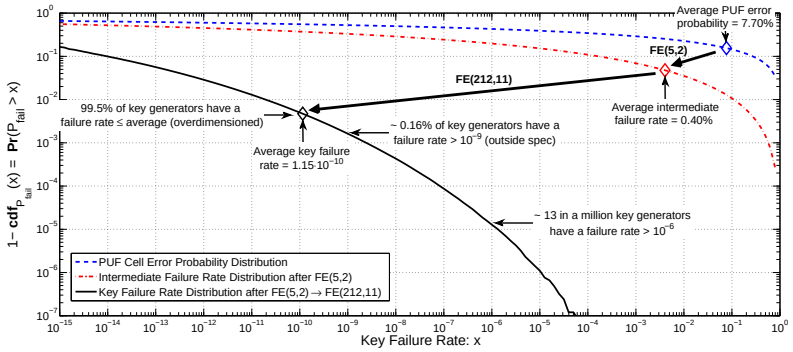

Fig. 4. Plot of the failure rate distribution of a PUF-based key generator.

generators actually reaches this average, or the required goal of 10^{-9}. This is a serious limitation which is solved by using the new reliability model.

The random distribution of key failure rates under the new model of Sect. 2.3 is hard to treat analytically since it involves an n-dimensional integration over the distribution of $\mathbf{p_e^n}$. However, we are able to efficiently simulate a key generator by randomly picking n error-probabilities according to (5) (using inverse transform sampling) and calculating $p_{\text{fail}}(\mathbf{p_e^n})$ with (8). By repeating this, we get many random samples of p_{fail} from which its distribution is estimated. We performed a simulation over 50,000,000 key generators, sampling 1,060 random error probabilities for each one, and calculating the resulting p_{fail} by applying (8) twice. The resulting simulated distribution is shown in Fig. 4, together with the initial error-probability distribution and the distribution of intermediate failure rates after FE$(5, 2)$ but before FE$(212, 11)$.

Interpretation of the Key Failure Rate Distribution. It is clear that the expected value of the derived key failure rate distribution under the new model is equivalent to the fixed key failure rate predicted under the old fixed error rate model. However, the failure rate distribution as plotted in Fig. 4 provides much more insight, e.g. it indicates not only the average failure rate but also the fraction of key generators actually attaining this average. For the studied example, we see that 99.5% of the generators operate above average, and even up to 99.84% have a failure rate within the specied goal of $p_{\text{fail}} \leq 10^{-9}$. On the other hand, this means that a very small but non-negligible fraction of 0.16% of the generators does not meet the specification. This is potentially important information for the application which is oblivious in the old fixed error rate model!

The small fraction of generators outside spec is not necessarily problematic. A large portion of that 0.16% still has a very small failure rate, only not as small as 10^{-9}. Only 13 in a million generators have $p_{\text{fail}} > 10^{-6}$, and less than 1 in 10 million generators have $p_{\text{fail}} > 10^{-4}$. Whether this is a problem depends on the envisioned application, such as the number of devices in the field and the acceptability of a potential failure. In fact, by taking these considerations into account the system specifications might even be relaxed, which will result in a more efficient design. E.g., a PUF-based key generator for a public transport ticketing system, with a huge number of deployed devices but a low criticality of failure, should be approached very differently than that for a life-supporting medical implant, with a relatively small number of devices in the field but an extremely high criticality of failure. The main advantage of the new model proposed in this work is exactly that it allows to study this tradeoff, whereas in the old model one is not aware of it.

5 Conclusion and Future Work

We introduced a more realistic new reliability model for silicon PUFs, which no longer assumes a single fixed error rate as before but considers randomly distributed cell error-probalities. An hypothetical error-probability distribution was derived based on plausible assumptions, including the effects of environmental factors like temperature. Experimental validations based on a substantial set of silicon PUF measurement data demonstrate a strikingly accurate fit of the predicted distributions on empirical statistics. This is a strong indication of the correctness and generic nature of the newly proposed model. An important implication of the use of this model is the ability to study the full failure distribution of a PUF-based application, whereas the old fixed error rate model only displays *average case* behavior. This introduces a new dimension in the design of PUF systems, allowing more focused specifications and better adapted solutions.

The ability to accurately describe the probabilistic reliability behavior of a silicon PUF spawns various seeds for future research. An obvious continuation of this work is the inclusion of more external parameters and conditions, besides temperature, in the model and the distributions; e.g, supply voltage variation, silicon device aging effects and technology node dependence. A further experimental validation on alternative silicon PUF technologies and under varying conditions will strengthen the applicability of the presented model. The offered possibility to realistically simulate PUF reliability behavior, as demonstrated in Sect. 4.2, could be of great

interest in the development of PUF-based applications, e.g. when no real PUF measurements are available. Finally, an interesting parallel research track is the analysis of unpredictability (entropy) of PUF responses using the same methods as presented in this work.

References

1. Gassend, B., Clarke, D., van Dijk, M., Devadas, S.: Silicon Physical Random Functions. In: ACM Conference on Computer and Communications Security (ACM CCS). (2002) 148–160
2. Guajardo, J., Kumar, S.S., Schrijen, G.J., Tuyls, P.: FPGA intrinsic PUFs and their use for IP protection. In: Workshop on Cryptographic Hardware and Embedded Systems (CHES). (2007) 63–80
3. NXP: PUF - Physical Unclonable Functions: Protecting next-generation Smart Card ICs with SRAM-based PUFs. `http://www.nxp.com/documents/other/75017366.pdf` (February 2013)
4. Microsemi: SmartFusion2 System-on-Chip FPGAs Product Brief. `http://www.actel.com/documents/SmartFusion2_PB.pdf` (February 2013)
5. Suh, G.E., Devadas, S.: Physical unclonable functions for device authentication and secret key generation. In: Design Automation Conference (DAC). (2007) 9–14
6. Bösch, C., Guajardo, J., Sadeghi, A.R., Shokrollahi, J., Tuyls, P.: Efficient Helper Data Key Extractor on FPGAs. In: Workshop on Cryptographic Hardware and Embedded Systems (CHES). (2008) 181–197
7. Maiti, A., Schaumont, P.: Improved Ring Oscillator PUF: An FPGA-friendly Secure Primitive. IACR Journal of Cryptology **24** (2011) 375–397
8. van der Leest, V., Preneel, B., van der Sluis, E.: Soft Decision Error Correction for Compact Memory-Based PUFs Using a Single Enrollment. In: Workshop on Cryptographic Hardware and Embedded Systems (CHES). (2012) 268–282
9. Maes, R., Herrewege, A.V., Verbauwhede, I.: PUFKY: A Fully Functional PUF-Based Cryptographic Key Generator. In: Workshop on Cryptographic Hardware and Embedded Systems (CHES). (2012) 302–319
10. Suzuki, D., Shimizu, K.: The Glitch PUF: A New Delay-PUF Architecture Exploiting Glitch Shapes. In: Workshop on Cryptographic Hardware and Embedded Systems (CHES). (2010) 366–382
11. Bhargava, M., Cakir, C., Mai, K.: Attack resistant sense amplifier based PUFs (SA-PUF) with deterministic and controllable reliability of PUF responses. In: IEEE International Symposium on Hardware-Oriented Security and Trust (HOST). (2010) 106–111
12. Chen, Q., Csaba, G., Lugli, P., Schlichtmann, U., Ruhrmair, U.: The Bistable Ring PUF: A new architecture for strong Physical Unclonable Functions. In: IEEE International Symposium on Hardware-Oriented Security and Trust (HOST). (2011) 134–141
13. Maes, R., Tuyls, P., Verbauwhede, I.: Soft Decision Helper Data Algorithm for SRAM PUFs. In: IEEE Symposium on Information Theory (ISIT). (2009) 2101–2105
14. Simons, P., van der Sluis, E., van der Leest, V.: Buskeeper PUFs, a promising alternative to D Flip-Flop PUFs. In: IEEE International Symposium on Hardware-Oriented Security and Trust (HOST). (2012) 7–12

15. van der Leest, V., Schrijen, G.J., Handschuh, H., Tuyls, P.: Hardware intrinsic security from D flip-flops. In: ACM Workshop on Scalable Trusted Computing (ACM STC). (2010) 53–62
16. Lee, J.W., Lim, D., Gassend, B., Suh, G.E., van Dijk, M., Devadas, S.: A technique to build a secret key in integrated circuits for identification and authentication application. In: Symposium on VLSI Circuits. (2004) 176–159
17. Maes, R., Verbauwhede, I.: Physically Unclonable Functions: A Study on the State of the Art and Future Research Directions. In Sadeghi, A.R., Naccache, D., eds.: Towards Hardware-Intrinsic Security. Information Security and Cryptography. Springer (2010) 3–37
18. Katzenbeisser, S., Kocabas, U., Rozic, V., Sadeghi, A.R., Verbauwhede, I., Wachsmann, C.: PUFs: Myth, Fact or Busted? A Security Evaluation of Physically Unclonable Functions (PUFs) Cast in Silicon. In: Workshop on Cryptographic Hardware and Embedded Systems (CHES). (2012) 283–301
19. (EU FP7-ICT 238811): UNIQUE Project - Foundations for Forgery-Resistant Security Hardware. `https://www.unique-project.eu/`
20. Maes, R., Rozic, V., Verbauwhede, I., Koeberl, P., van der Sluis, E., van der Leest, V.: Experimental evaluation of Physically Unclonable Functions in 65 nm CMOS. In: European Solid-State Circuits Conference (ESSCIRC). (2012) 486–489
21. Dodis, Y., Ostrovsky, R., Reyzin, L., Smith, A.: Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data. SIAM Journal on Computing **38**(1) (March 2008) 97–139
22. Maes, R., Tuyls, P., Verbauwhede, I.: Low-Overhead Implementation of a Soft Decision Helper Data Algorithm for SRAM PUFs. In: Workshop on Cryptographic Hardware and Embedded Systems (CHES). (2009) 332–347
23. Fernandez, M., Williams, S.: Closed-Form Expression for the Poisson-Binomial Probability Density Function. IEEE Transactions on Aerospace and Electronic Systems **46**(2) (april 2010) 803–817

A Basic Probability Distributions

The Binomial Distribution is the discrete distribution of the number of successes in n Bernoulli trials with constant success probability p. Its distribution functions are given by:

$$f_{\text{bino}}(x;n,p) = \binom{n}{x}p^x(1-p)^{n-x}\,, \text{ and } F_{\text{bino}}(x;n,p) = \sum_{i=0}^{\lfloor x\rfloor}\binom{n}{i}p^i(1-p)^{n-i}\,.$$

The Standard Normal Distribution is the normal distribution with zero mean and unit variance, denoted as: $\mathcal{N}(0,1)$. Any normal distribution can be expressed as a function of the standard normal: if $X \sim \mathcal{N}(\mu,\sigma^2)$, then $\frac{X-\mu}{\sigma} \sim \mathcal{N}(0,1)$. Its distribution functions are given by:

$$\varphi(x) = (2\pi)^{-\frac{1}{2}}e^{-\frac{x^2}{2}}\,, \text{ and } \Phi(x) = \frac{1}{2}\left(1+\operatorname{erf}\left(\frac{x}{\sqrt{2}}\right)\right)\,.$$

The Poisson-Binomial Distribution is the discrete distribution of the number of successes in n Bernoulli trials when the success probability is no longer constant, but different for every trial. The probability mass function and cumulative distribution function of the Poisson-binomial distribution can be efficiently calculated as shown in [23]:

$$f_{\mathrm{PB}}\left(x; \mathbf{p_e^n}\right) = \tfrac{1}{n+1} \sum_{i=0}^{n} C^{-i \cdot x} \prod_{k=1}^{n} \left(p_{e,k} C^i + (1 - p_{e,k})\right), \text{ with } C = e^{\frac{j 2\pi}{n+1}},$$

$$F_{\mathrm{PB}}\left(x; \mathbf{p_e^n}\right) = \tfrac{x+1}{n+1} + \tfrac{1}{n+1} \sum_{i=1}^{n} \tfrac{1 - C^{-i \cdot (x+1)}}{1 - C^{-i}} \prod_{k=1}^{n} \left(p_{e,k} C^i + (1 - p_{e,k})\right).$$

B Derivation of New Model Distributions[12]

All derived distributions concern random variables representing probabilities. This entails that all derived distribution functions are only defined on $(0, 1)$ and make no sense outside this interval. Most of the derived distributions approach infinity for $x \to 0^+$ and $x \to 1^-$, therefore, we only consider the *open* interval $(0, 1)$. This implies that, e.g. an error-probability cannot be a hard 0 (absolutely never wrong) or a hard 1 (absolutely always wrong), though it can be arbitrarily close to 0 or 1.

B.1 Fixed Temperature Model

The One-Probability Distribution is derived by considering the definition of its cumulative distribution function:

$$\mathbf{cdf}_P\left(x\right) \stackrel{\text{def}}{=} \mathbf{Pr}\left(P \leq x\right) = \Phi\left(\lambda_1 \Phi^{-1}\left(x\right) + \lambda_2\right),$$

$$\mathbf{pdf}_P\left(x\right) \stackrel{\text{def}}{=} \tfrac{d\,\mathbf{cdf}_P(x)}{dx} = \tfrac{\lambda_1 \varphi\left(\lambda_1 \Phi^{-1}(x) + \lambda_2\right)}{\varphi(\Phi^{-1}(x))},$$

by substituting the assumed normal distributions for M and N_i and using the short-hand parameters $\lambda_1 = {}^{\sigma_N}\!/_{\sigma_M}$, and $\lambda_2 = {}^{(t\,-\,\mu_M)}\!/_{\sigma_M}$.

The Error-Probability Distribution is derived by first considering the conditional probability density function of the error-probability with respect to the one-probability. Note that the error-probability of a cell i is

[12] In order to adhere to the page limit, the substeps in the following derivations are very limited. For a more detailed version of these derivations we refer to the full version of this work to appear on the Cryptology ePrint Archive (`http://eprint.iacr.org/`).

only completely determined at enrollment time, i.e. $p_{e,i} = p_i$ if $r_i^{\text{enroll}} = 0$ and $p_{e,i} = 1 - p_i$ if $r_i^{\text{enroll}} = 1$. The conditional distribution is derived as:

$$\mathbf{pdf}_{P_e|P=p_i}(x) = \begin{cases} p_i & \text{, for } x = 1 - p_i\,, \\ 1 - p_i & \text{, for } x = p_i\,, \\ 0 & \text{, for all other } x\,. \end{cases} = \begin{cases} 1 - x & \text{, for } p_i = 1 - x\,, \\ 1 - x & \text{, for } p_i = x\,, \\ 0 & \text{, for all other } p_i\,. \end{cases}$$

The unconditional probability functions of P_e then follow as:

$$\mathbf{pdf}_{P_e}(x) = \lambda_1(1-x)\frac{\varphi\left(\lambda_1\Phi^{-1}(x)+\lambda_2\right)+\varphi\left(\lambda_1\Phi^{-1}(x)-\lambda_2\right)}{\varphi(\Phi^{-1}(x))}\,,$$

$$\mathbf{cdf}_{P_e}(x) = \lambda_1 \cdot \int_{-\infty}^{\Phi^{-1}(x)} \Phi(-u) \cdot \left(\varphi(\lambda_1 u + \lambda_2) + \varphi(\lambda_1 u - \lambda_2)\right) \mathrm{d}u\,.$$

B.2 Model with Temperature Sensitivity

Conditional One-Probability Distribution. The main goal of the temperature extension of the basic model is to describe the evolution of a PUF cell's behavior over changing temperature, i.e. given a reference behavior what will be its behavior when the temperature changes. We first introduce a conditional variant of the one-probability to describe this, and derive the relation of this conditional one-probability to the hidden variables following from the temperature model relation given by (4).

$$p_i(T|T_{ref}) \stackrel{\text{def}}{=} \mathbf{Pr}\left(R_i(T) = 1 | p_i(T_{ref})\right) = \Phi\left(\Phi^{-1}\left(p_i(T_{ref})\right) + \frac{d_i \cdot \Delta T}{\sigma_N}\right)\,,$$

with $\Delta T = T - T_{ref}$ and using the normal distribution assumption for N_i. The distribution of the conditional one-probabilities follows from considering the definition of their cumulative distribution function:

$$\mathbf{cdf}_{P(T|T_{ref})}(x) \stackrel{\text{def}}{=} \mathbf{Pr}\left(P(T|T_{ref}) \leq x\right) = \Phi\left(\theta \cdot \frac{\Delta\Phi^{-1}(x)}{|\Delta T|}\right)\,,$$

$$\mathbf{pdf}_{P(T|T_{ref})}(x) = \frac{d\,\mathbf{cdf}_{P(T|T_{ref})}(x)}{dx} = \frac{\theta}{|\Delta T|} \cdot \frac{\varphi\left(\theta \cdot \frac{\Delta\Phi^{-1}(x)}{|\Delta T|}\right)}{\varphi(\Phi^{-1}(x))}.$$

with $\Delta\Phi^{-1}(x) = \Phi^{-1}(x) - \Phi^{-1}\left(p_i(T_{ref})\right)$ and after filling in the normal distribution assumption for D and using the short-hand notation $\theta = \frac{\sigma_N}{\sigma_D}$.

Error-Probability Distribution. We first express the conditional distribution of the error-probability conditioned on a known value for the one-probability at T_{ref}: $p_i(T_{ref})$, and a known value for the conditional one-probability $p_i(T|T_{ref})$:

$$\mathbf{Pr}\left(P_e(T;T_{ref}) = x | P(T|T_{ref}) = y, P(T_{ref}) = p_{i,ref}\right) = \begin{cases} p_{i,ref} & \text{, for } x = 1 - y\,, \\ 1 - p_{i,ref} & \text{, for } x = y\,, \\ 0 & \text{, for all other } x\,. \end{cases}$$

We begin with removing the conditioning on $p_i(T|T_{ref})$:

$$\mathbf{Pr}\left(P_e(T;T_{ref}) = x | P(T_{ref}) = p_{i,ref}\right) = (1-p_{i,ref}) \cdot \mathbf{pdf}_{P(T|T_{ref})}(x) + p_{i,ref} \cdot \mathbf{pdf}_{P(T|T_{ref})}(1-x)\ .$$

The unconditional distribution of $P_e(T;T_{ref})$ then follows as:

$$\begin{aligned}
\mathbf{pdf}_{P_e(T;T_{ref})}(x) &= \int_0^1 \Big((1-p_{i,ref}) \cdot \mathbf{pdf}_{P(T|T_{ref})}(x) + p_{i,ref} \cdot \mathbf{pdf}_{P(T|T_{ref})}(1-x)\Big) \mathbf{pdf}_P(p_{i,ref})\, dp_{i,ref}\,, \\
&= \tfrac{\lambda_1 \theta}{|\Delta T| \varphi(\Phi^{-1}(x))} \cdot \int_{-\infty}^{+\infty} \left[\Phi(-u)\,\varphi\left(\theta \tfrac{\Phi^{-1}(x)-u}{|\Delta T|}\right) + \Phi(u)\,\varphi\left(\theta \tfrac{\Phi^{-1}(x)+u}{|\Delta T|}\right)\right] \cdot \varphi(\lambda_1 u + \lambda_2)\ \mathrm{d}u\,. \\
\mathbf{cdf}_{P_e(T;T_{ref})}(x) &= \tfrac{\lambda_1 \theta}{|\Delta T|} \cdot \int_{-\infty}^{\Phi^{-1}(x)} \int_{-\infty}^{+\infty} \left[\Phi(-u)\,\varphi\left(\theta \tfrac{v-u}{|\Delta T|}\right) + \Phi(u)\,\varphi\left(\theta \tfrac{v+u}{|\Delta T|}\right)\right] \cdot \varphi(\lambda_1 u + \lambda_2)\ \mathrm{d}u\,\mathrm{d}v\,.
\end{aligned}$$

For $\Delta T \to 0^+$ this reverts to the distribution functions for the basic fixed temperature model as derived in App. B.1.

Appendix B

Paper: "Bias-based modeling and entropy analysis of PUFs"

Authors: Robbert van den Berg (Eindhoven University of Technology), Boris Škorić (Eindhoven University of Technology), and Vincent van der Leest (Intrinsic-ID)
Venue: International Workshop on Trustworthy Embedded Devices (TrustED) 2013
Date: November 4th, 2013

Status: Accepted for publication

Bias-based modeling and entropy analysis of PUFs

Robbert van den Berg
Eindhoven University of Technology
Eindhoven, The Netherlands

Boris Škorić
Eindhoven University of Technology
Eindhoven, The Netherlands

Vincent van der Leest
Intrinsic-ID
Eindhoven, The Netherlands

ABSTRACT

Physical Unclonable Functions (PUFs) are increasingly becoming a well-known security primitive for secure key storage and anti-counterfeiting. For both applications it is imperative that PUFs provide enough entropy. The aim of this paper is to propose a new model for binary-output PUFs such as SRAM, DFF, Latch and Buskeeper PUFs, and a method to accurately estimate their entropy. In our model the measurable property of a PUF is its set of *cell biases*. We determine an upper bound on the 'extractable entropy', i.e. the number of key bits that can be robustly extracted, by calculating the mutual information between the bias measurements done at enrollment and reconstruction.

In previously known methods only uniqueness was studied using information-theoretic measures, while robustness was typically expressed in terms of error probabilities or distances. It is not always straightforward to use a combination of these two metrics in order to make an informed decision about the performance of different PUF types. Our new approach has the advantage that it simultaneously captures both of properties that are vital for key storage: uniqueness and robustness. Therefore it will be possible to fairly compare performance of PUF implementations using our new method.

Statistical validation of the new methodology shows that it clearly captures both of these properties of PUFs. In other words: if one of these aspects (either uniqueness or robustness) is less than optimal, the extractable entropy decreases. Analysis on a large database of PUF measurement data shows very high entropy for SRAM PUFs, but rather poor results for all other memory-based PUFs in this database.

Categories and Subject Descriptors

B.8.2 [**Hardware**]: Performance and reliability—*Performance Analysis and Design Aids*

Keywords

PUF, SRAM, entropy

TrustED'13, November 4, 2013, Berlin, Germany.

http://dx.doi.org/10.1145/2517300.2517301.

1. INTRODUCTION

Due to deep-submicron manufacturing process variations every transistor in an integrated circuit (IC) has slightly different physical properties that lead to measurable differences. Examples of such physical properties are threshold voltages and gain factors of the IC's transistors. The submicron variations are uncontrollable during manufacturing, which ensures that these physical properties cannot be copied or cloned. Therefore these properties can be used to derive a unique fingerprint of an electronic circuit, similar to human biometrics. It is very hard, expensive and economically not viable to create a device with a specifically chosen fingerprint.

The functions used to derive unique fingerprints for ICs are known as Physical Unclonable Functions (PUFs). Implementing a PUF requires an electronic circuit that measures the responses of the hardware to certain given inputs or challenges, which depend on the unique physical properties of the device. In order for a PUF implementation to be practically useful the PUF should be easy to challenge and the response easy to measure, but very hard to reproduce by construction[1]. Common applications for PUFs are to use them as identification or authentication primitives [6, 11], storing secret keys "without actually storing them" [5, 10] (for IP protection or as a "root of trust" in secure environments), and random number generation [6, 26].

1.1 Physical Unclonable Functions

Pappu [16] introduced the concept of PUFs in 2001 under the name Physical One-Way Functions. The proposed technology was based on obtaining a response (scattering pattern) when shining a laser on a bubble-filled transparent epoxy wafer. In 2002 the first physical random function for silicon devices was introduced by Gassend et al. [4]. This function makes use of the manufacturing process variations in ICs, with identical masks, to uniquely characterize each IC. For this purpose the frequencies of ring oscillators were measured. Using this method (now known as a Ring Oscillator PUF), they were able to characterize ICs. In 2004 Lee et al. [11] proposed another PUF that is based on delay measurements, the Arbiter PUF. In 2010 Suzuki et al. [21] introduced the Glitch PUF, which exploits glitch waveforms from delay variation between gates.

Besides intrinsic PUFs based on delay measurements, a second type of PUF in ICs is known: the memory-based

[1]Note that the way a PUF is implemented is vital to the security of this PUF. E.g. in case of a memory-based PUF there should be no interface on which the start-up pattern can be read by an attacker (PUF response is kept secret).

PUF. These PUFs are based on the measurement of start-up values of memory cells. This memory-based PUF type includes SRAM PUFs, which were introduced by Guajardo et al. in 2007 [5]. Furthermore, so-called Butterfly PUFs were described in 2008 by Kumar et al. [10]. In the same year Maes et al. [14] introduced D Flip-Flop PUFs and Su et al. [19] published about Latch PUFs. Recently, Buskeeper PUFs were demonstrated by Simons et al. [18] in 2012.

1.2 PUF properties

In order for the IC to be uniquely identifiable, the PUF must be reliable and unique. In this case reliable means that one is able to reproduce the same behaviour of the function when challenged with the same input over and over again. The characteristics of electronic components depend on the environment they are exposed to (ambient temperature, voltage ramp-up curves, etc.), but also on the ageing process of CMOS. It is of crucial importance that the function has a stable behaviour across a range of environmental conditions during the lifetime of the IC. Typically it is observed that PUFs exhibit a noisy behaviour. Therefore the PUF implementation must include an error correction process to stabilize the PUF responses both over environmental conditions and over time.

The second important parameter for PUFs is entropy. At the time of PUF manufacture, there is an uncontrollable process that leads to the creation of stably measurable challenge-response properties. The uncontrollability of the manufacturing process ensures the physical unclonability of the PUF, provided that there is enough entropy. We require that the entropy of the uncontrollable stable PUF properties[2] is sufficiently high. When this requirement is met, the following properties hold:

- *Uniqueness.* The probability that two PUFs have closely resembling properties is exponentially small.
- *Unpredictability.* The probability of correctly predicting an unknown PUF's set of responses is exponentially small. Furthermore, knowledge of one PUF's responses does not help in the prediction of another PUF's responses and knowledge of part of PUF response does not help predicting the other bits from this particular response.

1.3 Contribution

The focus of this paper is on demonstrating a novel method for quantifying the usable ('extractable') entropy of PUF responses. *Mutual information* provides a fundamental upper bound on the amount of key material that can be reliably extracted from a PUF using a helper data scheme (a.k.a. Fuzzy Extractor) [3, 8, 15, 22].

We calculate the mutual information between the enrollment measurements and later reconstruction measurements. Here the *bias* of a memory cell / flip-flop / latch serves as the measurable PUF property; multiple enrollment measurements (k) and multiple reconstruction measurements (ℓ) are performed on each cell in order to estimate the bias. The mutual information between the k enrollment measurements and the ℓ reconstruction measurements is an upper bound on the usable entropy.

[2] Entropy of controllable part is irrelevant, since this part can be cloned. Entropy of unstable part is also irrelevant here since we cannot exploit it for reproducible key extraction.

In order to validate our approach and to quantify the results of our approach in a real-life setting we used a large data set from the European project UNIQUE. This statistical validation of the new methodology shows that it clearly captures both the uniqueness and robustness of PUFs. In other words: if one of these aspects is less than optimal, the extractable entropy calculated with this method will decrease. The analysis using the UNIQUE PUF measurement data shows very high entropy for SRAM PUFs, but rather poor results for all other memory-based PUFs of this database.

2. RELATED WORK

This paper has been derived from the work in the M.Sc. thesis of Robbert van den Berg [24] in 2012. In his work a new method is proposed for calculating (extractable) entropy for memory-based PUFs. In this section we briefly list known methods.

Extensive entropy analyses of optical PUFs by Tuyls et al. [23] and of coating PUFs by Škorić et al. [27] exist, but these analyses are not applicable to memory-based PUFs.

A simple first indication of uniqueness involves the calculation of Hamming Weights of PUFs. The Hamming weight of a PUF, the number of cells that return non-zero upon start-up, can be used to determine if a PUF is biased [9, 17, 25]. When sampling multiple PUFs, the minimum or maximum Hamming weight can be used to estimate an upper bound on the bias.

The inter-device (or between-class) Hamming distance is a measure of the uniqueness of PUFs; it indicates how easy it is to distinguish or identify different devices [20]. For uniqueness, it is desirable to have a fractional[3] inter-device distance close to 0.5 which means that on average half the cells prefer a different start-up state [2, 9, 10, 17, 25]. It indicates a low correlation between responses of different devices.

A method to derive a conservative lower bound on the entropy is calculating min-entropy based on the enrollment measurements of a set of PUFs. This is a very conservative entropy estimation, but a good one for measuring uncertainty about a cryptographic key [1, 2, 9, 18, 26]. However, it does not take into account how much entropy is lost due to noise.

An optimal compression algorithm can compress a PUF response to a description with length at least equal to the entropy of the PUF data. By reversing this principle, an optimal compression algorithm can be used to provide an estimate for the PUF entropy. In PUF entropy analysis, the Context-Tree Weighting algorithm (CTW) [28] is regularly used to estimate an upper bound on the entropy of PUFs [1, 2, 7, 17].

Furthermore, in [12] a model was developed for Silicon PUFs, but no entropies were computed. We work with a somewhat similar model and use it to estimate entropies.

3. MODELING BINARY-OUTPUT PUFS

Random variables are written with capitals, and their realizations in lower case. Vectors are in boldface. The number of components (memory bits / flip-flops / latches / ...) in the PUF is denoted as n. The components will be referred to as cells. We define the set $[n] = \{1, \ldots, n\}$. At enrollment, the PUF is fully characterized by a vector of biases: $\boldsymbol{b} = (b_i)_{i=1}^{n}$. When an enrollment measurement is done on cell i, the result

[3] A fractional Hamming distance is the Hamming distance between two strings divided by the length of the strings.

is '1' with probability b_i. For every cell, k enrollment measurements are done (with $k \geq 1$). The number of '1' results in cell i is denoted as x_i. We define $\boldsymbol{x} = (x_i)_{i=1}^n$. The random variable X_i is binomial-distributed with parameters k and b_i: $\Pr[X_i = x] = p_{x|b_i} := \binom{k}{x} b_i^x (1-b_i)^{k-x}$. We denote the joint probability as $p_{\boldsymbol{x}|\boldsymbol{b}} = \prod_{i\in[n]} p_{x_i|b_i}$.

In the reconstruction phase the environmental circumstances are in general different than during enrollment, which leads to modified cell biases b'_i. A number ℓ of measurements is done on each cell; the number of '1' results in cell i is denoted as y_i. The variable Y_i is binomial-distributed with parameters ℓ and b'_i. We define $q_{y|b'_i} = \binom{\ell}{y} (b'_i)^y (1-b'_i)^{\ell-y}$ and $q_{\boldsymbol{y}|\boldsymbol{b}'} = \prod_{i\in[n]} q_{y_i|b'_i}$. Note that $\boldsymbol{x}/k$ is an estimate of $\boldsymbol{b}$, and $\boldsymbol{y}/\ell$ is an estimate of $\boldsymbol{b}'$. The estimates become more accurate by increasing k and ℓ, respectively.

Biases $\boldsymbol{b}$ and $\boldsymbol{b}'$ are themselves the result of probabilistic processes: (i) Random variable $\boldsymbol{B}$ has a distribution ρ dictated by the randomness in PUF manufacturing. (ii) After enrollment there are random influences that alter $\boldsymbol{B}$ to $\boldsymbol{B}'$. This is modeled as a set of transition probabilities $\tau(\boldsymbol{b}'|\boldsymbol{b})$.

The amount of common key material that can be *reliably* extracted from the enrollment and reconstruction measurements is upper bounded by the mutual information $\mathrm{I}(\boldsymbol{X};\boldsymbol{Y}) = \mathsf{H}(\boldsymbol{X}) + \mathsf{H}(\boldsymbol{Y}) - \mathsf{H}(\boldsymbol{X},\boldsymbol{Y})$. Note that $\mathrm{I}(\boldsymbol{X};\boldsymbol{Y})$ depends on k and ℓ. We have

$$\Pr[\boldsymbol{X}=\boldsymbol{x}] = \int_0^1 \mathrm{d}^n b\ \rho(\boldsymbol{b})\ p_{\boldsymbol{x}|\boldsymbol{b}} \quad (1)$$

$$\Pr[\boldsymbol{Y}=\boldsymbol{y}] = \int_0^1 \mathrm{d}^n b' [\int_0^1 \mathrm{d}^n b\ \rho(\boldsymbol{b})\tau(\boldsymbol{b}'|\boldsymbol{b})]\ q_{\boldsymbol{y}|\boldsymbol{b}'} \quad (2)$$

$$\Pr[\boldsymbol{X}=\boldsymbol{x},\boldsymbol{Y}=\boldsymbol{y}] = \int_0^1 \mathrm{d}^n b\ \rho(\boldsymbol{b}) p_{\boldsymbol{x}|\boldsymbol{b}} \int_0^1 \mathrm{d}^n b'\ \tau(\boldsymbol{b}'|\boldsymbol{b}) q_{\boldsymbol{y}|\boldsymbol{b}'} \quad (3)$$

(In our notation the an integral is an operator acting on everything to the right.) Our aim is to estimate ρ and τ from our set of measurements on the UNIQUE PUFs, and then use Eqs. (1–3) to compute $\mathrm{I}(\boldsymbol{X};\boldsymbol{Y})$. However, the space in which the biases live is very large due to the large number of cells ($\boldsymbol{b}, \boldsymbol{b}' \in \mathcal{B} = [0,1]^n$), no matter how we discretize the interval $[0,1]$. This makes estimation of probability distributions difficult, since any histogram we construct is based on only N points in the whole space $\mathcal{B}$, where N is the number of PUFs we have at our disposal; the density of points is so low that typically each bin will contain at most one point.

We introduce the following, rather crude, approximation,

$$\rho(\boldsymbol{b}) \approx \prod_{i\in[n]} \rho_i(b_i) \quad ; \quad \tau(\boldsymbol{b}'|\boldsymbol{b}) \approx \prod_{i\in[n]} \tau_0(b'_i|b_i). \quad (4)$$

In words: (i) At manufacture, each cell has its own probability distribution (ρ_i) for the bias, independent of the other cells. (ii) We use *global* transition probabilities $\tau_0(\ \cdot\ |\ \cdot\ )$, independent of the cell index, to model the effect of environmental influences on the biases.

The functions ρ_i and τ_0 are defined on small domains: $[0,1]$ and $[0,1]^2$ respectively. Hence they can be estimated fairly accurately. Note that our approximation for ρ is not capable of modeling correlations between cells. Our approach (4) is motivated by (a) the lack of correlation we observe between cells in most of the PUF types (see Section 4.3), and (b) a feeling that the physics of the transitions $b_i \mapsto b'_i$ should not depend on the cell index i.

Substitution of (4) into (1–3) gives factorized equations,

$$\Pr[\boldsymbol{X}=\boldsymbol{x}] \approx \prod_{i\in[n]} \int_0^1 \mathrm{d}b_i\ \rho_i(b_i) p_{x_i|b_i} \quad (5)$$

$$\Pr[\boldsymbol{Y}=\boldsymbol{y}] \approx \prod_{i\in[n]} \int_0^1 \mathrm{d}b'_i [\int_0^1 \mathrm{d}b_i\ \rho_i(b_i)\tau_0(b'_i|b_i)] q_{y_i|b'_i} \quad (6)$$

$$\Pr[\boldsymbol{X}=\boldsymbol{x},\boldsymbol{Y}=\boldsymbol{y}] \approx \prod_{i\in[n]} \int_0^1 \mathrm{d}b_i \rho_i(b_i) p_{x_i|b_i} \int_0^1 \mathrm{d}b'_i \tau_0(b'_i|b_i) q_{y_i|b'_i} \quad (7)$$

The mutual information then consists of independent parts, $\mathrm{I}(\boldsymbol{X};\boldsymbol{Y}) \approx \sum_{i\in[n]} \mathrm{I}(X_i;Y_i) = \sum_{i\in[n]} \mathsf{H}(X_i) + \mathsf{H}(Y_i) - \mathsf{H}(X_i,Y_i)$.

4. RESULTS

4.1 Data set

To test the results of our proposed method, a large data set of PUF measurements has been used. This data set was created in the EU funded FP7 programme project UNIQUE (contract number 238811). The UNIQUE project yielded 192 ASICs featuring six different PUF types: SRAM, D Flip-Flop (DFF), Latch, Buskeeper, Arbiter and Ring Oscillator.

We analyze the four memory-based PUF types. Each ASIC has four instantiations of the Latch, DFF and SRAM PUF and two instantiation of the Buskeeper PUF. Unfortunately two Latch PUFs per ASIC are unusable due to faults in the addressing logic. Furthermore, during preliminary testing, one DFF instance was found to be very unreliable when compared to the other instances (also noted in [9, 13]). This instance we also excluded from the test data. This leaves a total of $2 \cdot 192 = 384$ Latch and Buskeeper PUFs, $3 \cdot 192 = 576$ DFF and $4 \cdot 192 = 768$ SRAM PUFs for analysis.

All these PUF types provide 8192 bits of output, except the SRAM PUF which has 65536 bits of output. However, during the entropy analysis we used only 8192 out of these 65536, in order to reduce the required memory for processing.

In the UNIQUE project, several different test (such as temperature and voltage variation) were done to determine reliability (e.g. in [9] and [12]). In this paper, we use the data from the temperature variation test for the uniqueness analysis, since it provides PUF responses obtained both at room temperature and at the standard operational temperature limits of electronics. The room temperature measurements are ideal candidates for enrollment, while the measurements at other temperatures provide reconstruction conditions. The data set contains a total of 60 measurements per PUF instantiation at +25°C (used as enrollment measurements) and 40 measurements at -40°C and +85°C respectively (used as reconstruction measurements). The two temperatures used for reconstruction have been chosen because the industrial standard for temperature testing of ICs ranges from -40°C to +85°C. Therefore, these two temperatures are the corner cases for using PUFs in industrial grade devices.

The analysis of the data has been performed on a 32-bit 3GHz dual core PC with 2GB RAM, using Matlab. To process the PUF data with Matlab, data matrices were created with cells as columns and measurements as rows. This was repeated for each PUF, creating a #Measurements × #Cells × #PUFs three-dimensional matrix per PUF instance. The memory size required for these matrices can become rather large as the number of elements of these matrices increases. For example, if all cells of the SRAM PUF would be used, a

$60 \times 65536 \times 192 = 754{,}974{,}720$ element matrix would be required to store enrollment data. However, as some Matlab functionality only works with (64-bit) doubles, these matrices cannot be stored and processed efficiently.

4.2 Applying the proposed model

In order to apply the model as proposed in Section 3 we need to investigate whether individual PUF cells are correlated with each other. Since the proposed method requires PUF cells to be independent, it should be made sure that this is indeed the case for the PUFs from the UNIQUE database. This verification is described in Section 4.3.

In order to make contact with the approaches in the literature, we first separately investigate PUF uniqueness and reliability, before presenting the mutual information results. A measure of device uniqueness is calculated in Section 4.4. For this purpose we use the inter-device distance. As stated before, the extractable entropy derived by the proposed model is based, besides uniqueness, also on the reliability of the PUFs. The robustness of the biases is calculated in Section 4.5.

We compute the mutual information in Section 4.6. This mutual information contains aspects of both the uniqueness and the reliability. The mutual information computed according to our model provides an estimated upper bound on the extractable entropy per cell. Finally we calculate the amount of extractable entropy per mm^2 for each PUF type.

Note that all results in this paper are taken from the M.Sc. thesis of Robbert van den Berg [24]. For more details on the results and for comparisons of our method with results from methods in literature, we refer the reader to this thesis.

4.3 Correlations

Pearson's product-moment coefficient is calculated for every PUF to determine if there is any correlation among cells. Although 0 correlation does not directly imply independence, any correlation found during testing would indicate that there exists linear dependencies between cells. In literature, PUF cells are generally assumed independent (e.g. [2, 18]).

For this test, the first 1024 cells of each PUF are used. Pairwise, the covariance of two cells is divided by the product of their standard deviations as shown in Eq. (8). The result is a value between -1 and 1, where 1 denotes a very strong positive relation and -1 denotes a very strong negative relation, which means that when the bias of cell i increases, the bias of cell j decreases. The closer this value lies to zero the weaker the relationship between the two cells.

$$\mathrm{Corr}(x_i, x_j) = \frac{\mathrm{Cov}(x_i, x_j)}{\sigma_{x_i}\sigma_{x_j}} \tag{8}$$

Furthermore, we calculated the probabilities of getting a correlation as large as observed under the hypothesis that there is no correlation. When this probability is less than 0.01, a correlation is considered significant.

For all PUF instances, the percentage of cells failing the hypotheses of no correlation lies around 0.010 with a maximum of 0.014 for the Latch PUF. This amount of significant correlations is exactly what can be expected by chance. Furthermore, from the significant correlations, the strength of the correlation is approximately 0.2. When the same correlation test is applied to synthetically generated PUF data (known to be independent), similar values are observed. These results indicate that linear dependence among cells is very low or non-existent. We cannot exclude nonlinear dependences.

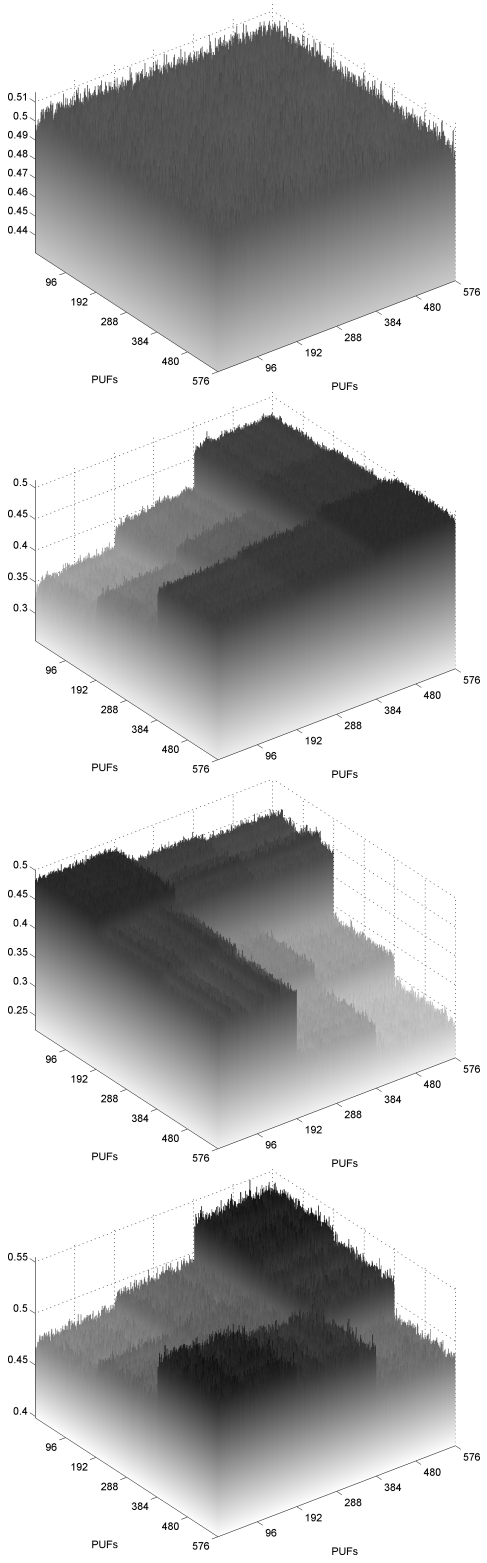

Figure 1: *Inter-device distances $\Delta_{p,p'}$. Device numbers 1–192 refer to the set of devices at $-40^\circ C$, 193–384 denote the same devices at $+25^\circ C$, and 385–576 at $+85^\circ C$.* **From top to bottom:** *SRAM, DFF, Latch and Buskeeper.*

4.4 Inter-device distance

Let $x_i^{(p)}$ be the x-count in cell i of PUF p. We define the inter-device distance $\Delta_{p,p'}$ between PUFs p and p' as the cell-average of the absolute bias difference,

$$\Delta_{p,p'} := \frac{1}{n}\sum_{i=1}^{n}\left|\frac{x_i^{(p)}}{k} - \frac{x_i^{(p')}}{k}\right|. \tag{9}$$

Fig. 1 shows inter-device distances. Here the device numbers 1–192 refer to the set of devices at $-40°$C, while 193–384 denote the same set of devices at $+25°$C, and 385–576 at $+85°$C. For the SRAM PUFs the temperature seems to have no effect on the inter-device distances, which are all close to 50%. This indicates a close to optimal inter-device distance (around 50% is optimal), which is also very stable over different environmental conditions.

For DFF PUFs the distances become smaller with decreasing temperature. This happens because the average Hamming Weight of the DFF PUFs rises with decreasing temperature. At $-40°$C this value gets close to 100%, which leaves little room for differences between devices.

In Latch PUFs the opposite happens: distances become smaller with increasing temperature. In this case Hamming Weight rises with temperature (close to 100% at $+85°$C).

The Buskeeper behaves differently. There is a marked difference between $+85°$C and the other temperatures. This is because the Buskeeper memories are slightly biased towards 0 at $-40°$ and $+25°$, while there is a significant bias towards 1 at $+85°$. The result is a lower inter-device distance when comparing devices at an equal temperature and comparing $-40°$ to $+25°$. Comparing devices at $+85°$ to any other temperature results in a higher inter-device distance, since the Hamming Weight is very different in these cases.

4.5 Robustness of the biases

We denote the vector $\boldsymbol{x}$ associated with the a'th PUF as $\boldsymbol{x}^{(a)}$, and similarly $\boldsymbol{y}^{(a)}$. The robustness of a cell's bias can be characterized using the following distance measure,

$$D_i := \frac{1}{N}\sum_{a=1}^{N}\left|\frac{x_i^{(a)}}{k} - \frac{y_i^{(a)}}{\ell}\right|. \tag{10}$$

Here $i \in [n]$ is the cell index. Values for large k and ℓ are listed in Table I.

Table I. *Bias robustness of UNIQUE PUFs. Listed values are average and maximum D_i values, with k and ℓ very large.*

	Av. distance		*Max. distance*	
Instance	-40°C	+85°C	-40°C	+85°C
SRAM 1	0.054	0.050	0.059	0.056
SRAM 2	0.053	0.050	0.060	0.057
SRAM 3	0.053	0.050	0.059	0.057
SRAM 4	0.053	0.050	0.061	0.058
DFF 1	0.125	0.176	0.158	0.194
DFF 2	0.153	0.174	0.318	0.217
DFF 3	0.122	0.178	0.157	0.196
DFF 4	0.120	0.177	0.166	0.197
Latch 1	0.231	0.103	0.274	0.171
Latch 2	0.233	0.117	0.277	0.182
Buskeeper 1	0.09	0.172	0.099	0.196
Buskeeper 2	0.092	0.171	0.101	0.20

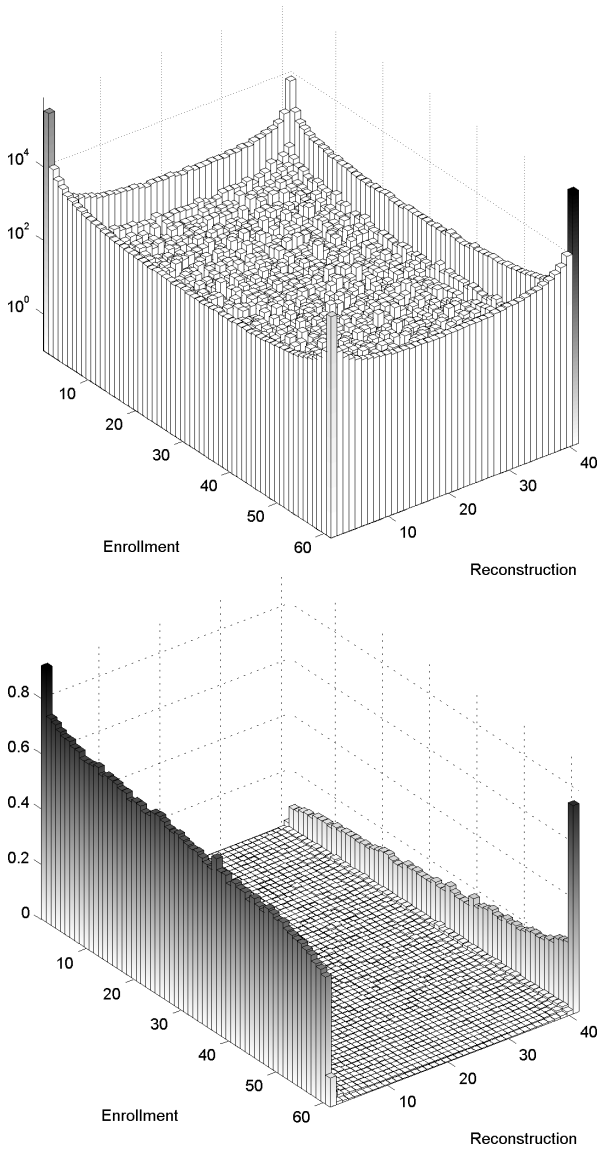

Figure 2: *Bias changes in the DFF PUFs at $+85°$C.* **Top:** *Histogram of (x_i, y_i) pairs, for $k = 60$, $\ell = 40$, on a logarithmic scale. The vertical axis counts the number of cells in which a combination (x, y) occurs.* **Bottom:** *The transition probabilities $\tau_0(\frac{y}{\ell}|\frac{x}{k})$ derived from the histogram, plotted as a function of x and y.*

In Fig. 2 we show observed bias transition counts and the transition model derived from them (transition probabilities τ_0). The figure shows the result for DFF PUFs; the other PUF types behave similarly. We see that biases far away from 0 and 1 practically never occur (not even when the enrollment bias lies around 0.5). Note that the top figure is logarithmically scaled in order to make the low parts of the histogram visible. Hence, the typical bias changes that occur are jumps to 0 or 1.

Furthermore, as expected, in the bottom figure we see that the probability mass of y given x is concentrated at small y when x is small, and at large y when x is large. In DFF PUFs at +85°C, bias jumps to 0 are more likely than jumps to 1.

4.6 Mutual information

For each of the four PUF types we have estimated the mutual information $I(\boldsymbol{X};\boldsymbol{Y})$ using the independent-cell approximation (4), with empirical ρ_i and τ_0. The results are shown in Fig. 3, as an average per cell, as a function of k and ℓ. Unsurprisingly, (i) the mutual information grows with increasing k and ℓ; and (ii) saturation occurs at large k,ℓ.

The rate of growth is not the same for all PUF types. SRAM PUFs benefit most from increasing k and ℓ. Note that SRAM PUFs can achieve a mutual information of more than one bit per cell. This is entirely natural, since this mutual information is calculated based on the values of the cell biases (and not on the binary start-up values of these cells). These cell biases are continuum variables which in theory can have infinite entropy. Note also that even at $k = 1$ (a single enrollment measurement) it is advantageous to take $\ell > 1$.

Finally, based on the mutual entropy results and known size of the PUF instances on the UNIQUE ASIC (based on [13]) the minimum number of extractable bits per mm^2 of each PUF type can be calculated. The results of the calculation can be found in Table III.

From these results it becomes very clear that the SRAM PUF by far has the highest extractable entropy out of all these PUF types. This is no surprise, since SRAM PUFs were found to be the most reliable and unique PUFs in [9, 13]. Furthermore, the number of PUF cells per mm^2 is also highest for the SRAM PUF. Hence there are multiple reasons why none of the other PUFs even comes close to the performance of the SRAM PUF.

Out of the other (memory-based) PUF types, the Buskeeper PUF can be ranked in second place (fairly good uniqueness, but much less robust over temperature variations). Both the DFF and Latch PUFs (ranked third and fourth respectively) perform much worse, because for these PUFs both the uniqueness and robustness are poor. All of these results are comparable to the conclusions drawn in [9, 13] about the UNIQUE data set.

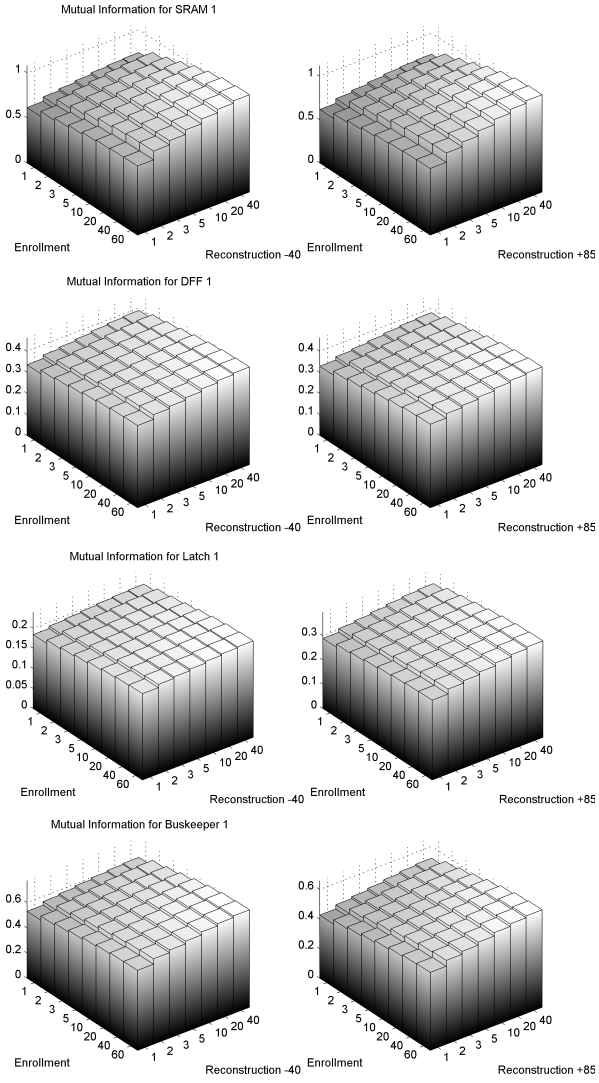

Figure 3: *Mutual information between X_i and Y_i as a function of k and ℓ, for the four PUF types, at reconstruction temperatures $-40°C$ and $+85°C$.* **From top to bottom:** *SRAM, DFF, Latch and Buskeeper.*

Table II. *Mutual information for different k and ℓ. Instance and condition giving the lowest mutual information per PUF type is marked with *.*

Cond.	Instance	Mutual Information (per cell) k=1, ℓ=1	k=60, ℓ=1	k=60, ℓ=40
−40°C	SRAM 1*	0.61	0.77	1.08
	SRAM 2	0.61	0.77	1.08
	SRAM 3	0.61	0.77	1.08
	SRAM 4	0.62	0.77	1.08
	DFF 1	0.33	0.39	0.46
	DFF 3	0.33	0.39	0.45
	DFF 4*	0.33	0.38	0.44
	Latch 1*	0.18	0.21	0.24
	Latch 2	0.20	0.23	0.26
	Busk. 1	0.53	0.63	0.77
	Busk. 2	0.52	0.62	0.75
+85°C	SRAM 1	0.62	0.77	1.12
	SRAM 2	0.62	0.77	1.12
	SRAM 3	0.62	0.77	1.12
	SRAM 4	0.62	0.77	1.12
	DFF 1	0.33	0.40	0.46
	DFF 3	0.32	0.39	0.46
	DFF 4	0.33	0.40	0.46
	Latch 1	0.29	0.33	0.40
	Latch 2	0.28	0.32	0.39
	Busk. 1	0.43	0.53	0.66
	Busk. 2*	0.42	0.52	0.65

Table III. *Extractable bits per mm^2 on the UNIQUE chip, broken down to PUF type and depending on k and ℓ. The lowest numbers were taken from Table II.*

PUF type	Area (mm^2)	Cells/ mm^2	Minimum #bits/mm^2 k=1, ℓ=1	k=60, ℓ=1	k=60, ℓ=40
SRAM	0.213	≈ 1.2M	0.75M	0.95M	1.3M
DFF	0.392	≈ 84k	28k	32k	37k
Latch	0.272	≈ 0.12M	22k	25k	29k
Busk.	0.076	≈ 0.22M	91k	0.11M	0.14M

5. CONCLUSIONS AND FUTURE WORK

We have developed a model for memory-based PUFs that treats the *cell biases as the identifying property* of the PUF. The enrollment procedure, consisting of k measurements, gives an estimate $\boldsymbol{X}/k$ of the cell biases $\boldsymbol{b}$ under enrollment conditions; similarly the ℓ reconstruction measurements give an estimate $\boldsymbol{Y}/\ell$ of the biases $\boldsymbol{b}'$ at reconstruction conditions. The mutual information $I(\boldsymbol{X};\boldsymbol{Y})$ is an upper bound on the amount of key material that can be reliably extracted from the PUF. The mutual information depends on the probability distribution $\rho(\boldsymbol{b})$, which models the uncontrollable manufacturing process, and on the transition probabilities $\tau(\boldsymbol{b}'|\boldsymbol{b})$ which model the various sources of noise.

This approach has the advantage that it simultaneously captures two issues of vital importance for key storage: uniqueness and robustness. (Usually only uniqueness is studied using information-theoretic measures; robustness is typically expressed in terms of error probabilities or distances.)

We have applied our model to the UNIQUE date set, assuming that all cells are independent. Furthermore, we have adopted a specific noise model in which the transition probabilities $\tau_0(b'|b)$ do not depend on the cell index. Our analysis shows a very high entropy for the SRAM PUFs in the UNIQUE database (especially when the number of enrollment and reconstruction measurements increases, the entropy per cell becomes more than 1). However, all other PUFs contain significantly less entropy. The Latch PUFs perform poorly, with values between 0.18 and 0.40 bits of entropy per cell.

Based on the results from this paper, we foresee as future work:

- Mutual information estimates including correlations between cells. This requires dealing with an $n \times n$ correlation matrix, which is cumbersome for large n.
- We have not addressed the question of Fuzzy Extractor design. The mutual information $I(\boldsymbol{X};\boldsymbol{Y})$ is an upper bound on the amount of extractable key material, but knowing this number does not tell you *how* to achieve this bound. Efficient Fuzzy Extractors have to be found.

Acknowledgements

This work has been supported by the European Commission through the ICT program under contract INFSO-ICT-284833 (PUFFIN).

References

[1] Frederik Armknecht, Roel Maes, Ahmad-Reza Sadeghi, Franccois-Xavier Standaert, and Christian Wachsmann. 2011. A Formal Foundation for the Security Features of Physical Functions. *IEEE Security and Privacy 2011* 2011, 1 (2011), 16.

[2] Mathias Claes, Vincent van der Leest, and An Braeken. 2012. Comparison of SRAM and FF PUF in 65nm technology. In *Proceedings of the 16th Nordic conference on Information Security Technology for Applications (NordSec'11)*. Springer-Verlag, Berlin, Heidelberg, 47–64. DOI: http://dx.doi.org/10.1007/978-3-642-29615-4_5

[3] Y. Dodis, M. Reyzin, and A. Smith. 2004. Fuzzy Extractors: How to generate strong keys from biometrics and other noisy data. In *Eurocrypt 2004 (LNCS)*, Vol. 3027. 523–540.

[4] Blaise Gassend, Dwaine Clarke, Marten van Dijk, and Srinivas Devadas. 2002. Silicon physical random functions. In *ACM Conference on Computer and Communications Security (CCS'02)*. ACM, New York, NY, USA, 148–160.

[5] Jorge Guajardo, Sandeep S. Kumar, Geert-Jan Schrijen, and Pim Tuyls. 2007. FPGA Intrinsic PUFs and Their Use for IP Protection. In *Workshop on Cryptographic Hardware and Embedded Systems (CHES '07) (LNCS)*, Pascal Paillier and Ingrid Verbauwhede (Eds.), Vol. 4727. Springer-Verlag, Berlin, Heidelberg, 63–80. DOI:http://dx.doi.org/10.1007/978-3-540-74735-2_5

[6] Daniel E. Holcomb, Wayne P. Burleson, and Kevin Fu. 2009. Power-Up SRAM State as an Identifying Fingerprint and Source of True Random Numbers. *IEEE Trans. Computers* 58, 9 (2009), 1198–1210.

[7] Tanya Ignatenko, Geert-Jan Schrijen, Boris Škorić, Pim Tuyls, and Frans M. J. Willems. 2006. Estimating the secrecy rate of Physical Uncloneable Functions with the Context-Tree Weighting method. In *Proc. IEEE International Symposium on Information Theory 2006*. Seattle, USA, 499–503.

[8] A. Juels and M. Wattenberg. 1999. A fuzzy commitment scheme. In *ACM Conference on Computer and Communications Security (CCS) 1999*. 28–36.

[9] Stefan Katzenbeisser, Ünal KocabaÅ§, Vladimir Rožić, Ahmad-Reza Sadeghi, Ingrid Verbauwhede, and Christian Wachsmann. 2012. PUFs: Myth, Fact or Busted? A Security Evaluation of Physically Unclonable Functions (PUFs) Cast in Silicon. In *Cryptographic Hardware and Embedded Systems (CHES) 2012*, Emmanuel Prouff and Patrick Schaumont (Eds.). Lecture Notes in Computer Science, Vol. 7428. Springer Berlin Heidelberg, 283–301. DOI:http://dx.doi.org/10.1007/978-3-642-33027-8_17

[10] S.S. Kumar, J. Guajardo, R. Maes, G.-J. Schrijen, and P. Tuyls. 2008. The butterfly PUF protecting IP on every FPGA. In *IEEE International Workshop on Hardware-Oriented Security and Trust (HOST'08)*, Mohammad Tehranipoor and Jim Plusquellic (Eds.). IEEE Computer Society, 67–70. DOI:http://dx.doi.org/10.1109/HST.2008.4559053

[11] J.W. Lee, Daihyun Lim, B. Gassend, G.E. Suh, M. van Dijk, and S. Devadas. 2004. A technique to build a secret key in integrated circuits for identification and authentication applications. In *IEEE Symposium on VLSI Circuits 2004*. IEEE, 176–179. DOI:http://dx.doi.org/10.1109/VLSIC.2004.1346548

[12] R. Maes. 2013. An Accurate Probabilistic Reliability Model for Silicon PUFs. In *Workshop on Cryptographic Hardware and Embedded Systems (CHES) 2013*.

[13] R. Maes, V. Rozic, I. Verbauwhede, P. Koeberl, E. van der Sluis, and V. van der Leest. 2012. Experimental evaluation of Physically Unclonable Functions in 65 nm CMOS. In *ESSCIRC (ESSCIRC), 2012 Proceedings of the*. 486 –489. DOI:http://dx.doi.org/10.1109/ESSCIRC.2012.6341361

[14] Roel Maes, Pim Tuyls, and Ingrid Verbauwhede. 2008. Intrinsic PUFs from Flip-flops on Reconfigurable Devices. In *Workshop on Information and System Security (WISSec 2008)*. Eindhoven, NL, 17.

[15] J.-P. Linnartz P. and Tuyls. 2003. New Shielding Functions to Enhance Privacy and Prevent Misuse of Biometric Templates. In *Audio- and Video-Based Biometric Person Authentication*. Springer.

[16] Ravikanth Srinivasa Pappu. 2001. *Physical one-way functions*. Ph.D. Dissertation. Massachusetts Institute of Technology. AAI0803255.

[17] Geert-Jan Schrijen and Vincent van der Leest. 2012. Comparative analysis of SRAM memories used as PUF primitives. In *Design, Automation Test in Europe Conference Exhibition (DATE) 2012*. 1319 –1324.

[18] Peter Simons, Erik van der Sluis, and Vincent van der Leest. 2012. Buskeeper PUFs, a Promising Alternative to D Flip-Flop PUFs. In *IEEE International Workshop on Hardware-Oriented Security and Trust (HOST'12), in print*. IEEE Computer Society.

[19] Ying Su, J. Holleman, and B.P. Otis. 2008. A Digital 1.6 pJ/bit Chip Identification Circuit Using Process Variations. *Solid-State Circuits, IEEE Journal of* 43, 1 (2008), 69–77. DOI:http://dx.doi.org/10.1109/JSSC.2007.910961

[20] G.E. Suh and S Devadas. 2007. Physical Unclonable Functions for Device Authentication and Secret Key Generation. In *Design Automation Conference, 2007. DAC '07. 44th ACM/IEEE*. 9–14.

[21] Daisuke Suzuki and Koichi Shimizu. 2010. The Glitch PUF: A New Delay-PUF Architecture Exploiting Glitch Shapes. In *Cryptographic Hardware and Embedded Systems, CHES 2010*, Stefan Mangard and Francois-Xavier Standaert (Eds.). Lecture Notes in Computer Science, Vol. 6225. Springer Berlin Heidelberg, 366–382. DOI:http://dx.doi.org/10.1007/978-3-642-15031-9_25

[22] P. Tuyls, B. Škorić, and T. Kevenaar. 2007. *Security with Noisy Data: Private Biometrics, Secure Key Storage and Anti-Counterfeiting*. Springer, London.

[23] P. Tuyls, B. Škorić, S. Stallinga, T. Akkermans, and W. Ophey. 2004. An information theoretic model for Physical Uncloneable Functions. In *Information Theory, 2004. ISIT 2004. Proceedings. International Symposium on*. 141–. DOI:http://dx.doi.org/10.1109/ISIT.2004.1365176

[24] R. van den Berg. 2012. Entropy analysis of Physical Unclonable Functions. MSc. thesis, Eindhoven University of Technology. (2012).

[25] Vincent van der Leest, Geert-Jan Schrijen, Helena Handschuh, and Pim Tuyls. 2010. Hardware intrinsic security from D flip-flops. In *Proceedings of the fifth ACM workshop on Scalable trusted computing (STC '10)*. ACM, New York, NY, USA, 53–62. DOI:http://dx.doi.org/10.1145/1867635.1867644

[26] Vincent van der Leest, Erik van der Sluis, Geert-Jan Schrijen, Pim Tuyls, and Helena Handschuh. 2012. Efficient Implementation of True Random Number Generator Based on SRAM PUFs. In *Cryptography and Security: From Theory to Applications*, David Naccache (Ed.). Lecture Notes in Computer Science, Vol. 6805. Springer Berlin Heidelberg, 300–318.

[27] B. Škorić, S. Maubach, T. Kevenaar, and P. Tuyls. 2006. Information-theoretic analysis of capacitive Physical Unclonable Functions. *Journal of Applied Physics* 100, 2 (2006), 024902–024902–11. DOI:http://dx.doi.org/10.1063/1.2209532

[28] F.M.J. Willems, Y.M. Shtarkov, and T.J. Tjalkens. 1995. The context-tree weighting method: basic properties. *Information Theory, IEEE Transactions on* 41, 3 (1995), 653–664. DOI:http://dx.doi.org/10.1109/18.382012