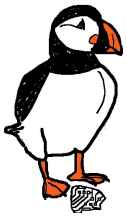


PUFFIN

Physically unclonable functions found in standard PC components

Project number: 284833
FP7-ICT-2011-C

D2.2

Scientific contribution of WP2, part 2 Analysis and Qualification

Due date of deliverable: 31. January 2015
Actual submission date: 31. January 2015

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 0.1

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 2
Analysis and Qualification

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

31. January 2015
Revision 0.1

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 second phase (months 19-36) 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 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 second phase (months 19-36) 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. 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. In the first period two methodologies were developed, which each focussed on one of the two basic properties of PUFs: reliability and uniqueness. These methodologies have already been reported in Deliverable D2.1. During the second phase of the PUFFIN project a new methodology has been developed for evaluating the trends for both reliability and uniqueness of SRAM PUFs during the lifetime of devices. Using this method it is possible to predict how a PUF will behave over time and what is required to make sure that the Fuzzy Extractor will be able to reconstruct the required enrollment pattern during the entire lifetime of an electronic device.

This new methodology is described in detail in a scientific publications that has been attached to this deliverable as an appendix. An additional second appendix has been added containing a paper that has been submitted by members of the PUFFIN project to the Journal of Cryptographic Engineering. This paper contains an overview of the work that has been performed in the project on analysing PUF data. Therefore this paper is the most important dissemination of the results from WP2 to the scientific community. The paper is currently still under review by the journal.

Chapter 2

Preliminary Analysis of PUFs

2.1 Introduction

During the PUFFIN project WP2 has received PUF measurements from WP1 for analysis. 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 in the PUFFIN project (both in the first and second project phase) 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,
- Pandaboards (computer development platform containing either Texas Instruments OMAP4430 or 4460, see `www.pandaboard.org` for more information),
- Texas Instruments Stellaris LM4F120H5QR microcontrollers,
- Texas Instruments AM3358 microcontrollers (on BeagleBone boards), and
- Atmel ATmega1280 microcontrollers (on Arduino Mega boards).

Deliverables D1.1 and D1.2 of the PUFFIN project explain which SRAM memories of these platforms have been used for these measurements and how data has been extracted.

Note: Part of the measurement analysis on these devices has already been reported in Deliverable D2.1. This document reports the analyses that have been performed in the second phase of the PUFFIN project. All results (both from project phase 1 and 2) will be summarized in the conclusions of this chapter.

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).
- Min-Entropy Test (MET)

The following sections provide a description for each of these tests. The Temperature Cycle Test has not been performed on all platforms, because some platforms (tablets, GPUs, and certain development boards) cannot be measured under extreme temperature conditions. Some components of these platforms will not survive extreme heat or cold.

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.

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 (fractional) Hamming distances between the enrolment measurements of the devices. Devices can be uniquely identified if there is some or no 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 in the order of 50%. The closer this value is to 50%, the smaller the correlation between devices is.

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.

2.2.5 Min-Entropy Test

This test is performed using either the measurements from the Repeated Start-up Test or from the Temperature Cycle Test. It is an additional test for evaluating the noise characteristics of the PUF candidates.

This test has been used to evaluate whether measured memories are suitable for creating a high quality seed from the noise present on the SRAM measurements for the Random Number Generator that has been developed in WP3. For this purpose, the amount of entropy present in the noise of SRAM start-up patterns should be determined. The method used here is based on the NIST specification SP800-90A (Recommendation for Random Number Generation Using Deterministic Random Bit Generators) that defines min-entropy as the worst-case (i.e., the greatest lower bound) measure of uncertainty for a random variable.

For a binary source, min-entropy is defined as:

$$H_{min} = -\log_2(\max(p_0, p_1)), \tag{2.1}$$

where p_0 and p_1 are the probabilities of 0 and 1 occurring. Assuming that all bits from the SRAM start-up pattern are independent, each bit i can be viewed as an individual binary source. For each of these sources the probabilities p_0^i and p_1^i of powering up in state 0 or 1 can be estimated, by repeatedly measuring the power-up values of the SRAM. In case m subsequent measurements are performed, p_0^i denotes the number of occurrences of a zero, divided by m and $p_1^i = 1 - p_0^i$. For n independent sources (where n is the length of the start-up pattern), we have:

$$H_{min} = \sum_{i=1}^{n} -\log_2(\max(p_0^i, p_1^i))). \tag{2.2}$$

Hence, under the assumption that all bits are independent, the min-entropy of each individual SRAM cell can be summed to derive the min-entropy of the entire SRAM. To represent this min-entropy of the entire SRAM, we choose to display it as a percentage of the total size of the SRAM by using the following formula:

$$H_{min} = \sum_{i=1}^{n} -\log_2(\max(p_0^i, p_1^i))) \times \frac{100\%}{n}. \tag{2.3}$$

A Matlab script is used to calculate p_0^i and p_1^i for each individual SRAM cell, based on all available measurements for a specific device (*within-class*) at a certain temperature. The more measurements of a single device are available at a temperature, the more accurate the estimate of the probabilities will be and therefore the more accurate the min-entropy estimate will be.

2.3 Test Results

This section contains the results for the PUF responses that have been acquired and analysed in the second phase of the PUFFIN project. This section contains four subsection with different kinds of data and analysis results. These subsections comprise of the following:

- The first subsection contains new Temperature Cycle Tests that have been performed on platforms that were already measured at room temperature during the first project phase. These platforms are the Microchip PIC16F1825, the ST STM32F100R8, and the Atmel ATmega328p.
- The second subsection also describes new measurements on a platform from the first project period, in this case the NVIDIA GeForce GTX 295 graphics card. For this device more data has been extracted during the second project phase. The size of the SRAMs measured has increased (to almost 16KB) and the number of SRAMs measured has increased to 510 individual SRAMs.
- In the third subsection analysis has been performed on three new platforms, which were measured during the second project period: the Texas Instruments Stellaris LM4F120H5QR, the BeagleBone board with a Texas Instruments AM3358 microcontroller, and the Arduino Mega board with an Atmel ATmega1280 microcontroller.
- Finally, the fourth subsection describes a new test which has been performed on all measured devices (both from period 1 and 2): the Min-Entropy Test, as described in the previous section. This test is used to evaluate whether a platform is suitable to create a strong seed for a Random Number Generator.

In the rest of this section, the results for the above described analyses can be found. Conclusions about these results are combined with those from the first project period in the next section.

2.3.1 New Temperature Cycle Tests on platforms from first project phase

Microchip PIC16F1825

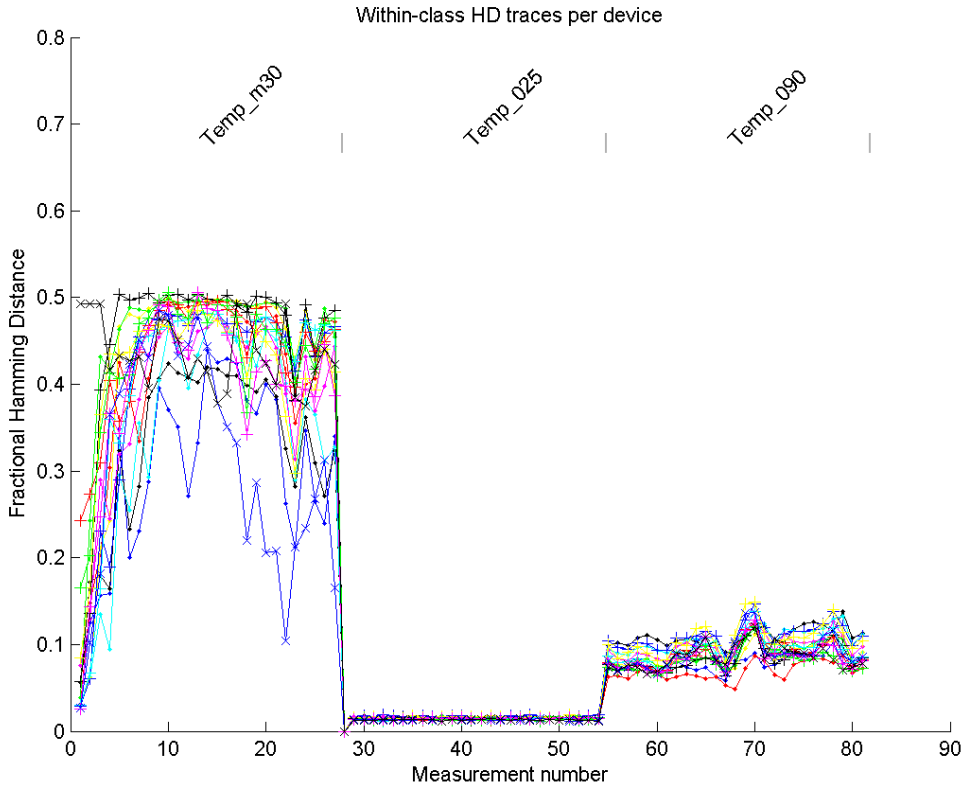


Figure 2.3.1: Within-class Hamming distance of SRAM in PIC16F1825 measured over different temperatures.

Figure 2.3.1 shows the results from the measurements of the Temperature Cycle Test for the 16 Microchip PIC16F1825 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 is extremely high, close to 50%. This amount of noise can certainly not be corrected using commonly known Fuzzy Extractors, because it is in the same order of magnitude as the Between-Class Hamming Distance should be. The explanation for these high noise values at low temperatures can be found in Figure 2.3.2, which displays the Hamming Weights of these measurements at different temperatures. What can be seen in this picture is that the SRAMs basically "freeze" at low temperatures, resulting in all SRAM cells going to '0'. Since the Hamming Weight is going from 50% at enrollment to almost 0% at low temperatures, it is clear that the Hamming Distances between measurements at these temperatures is approximately 50%. Therefore, **these devices fail the Temperature Cycle Test**.

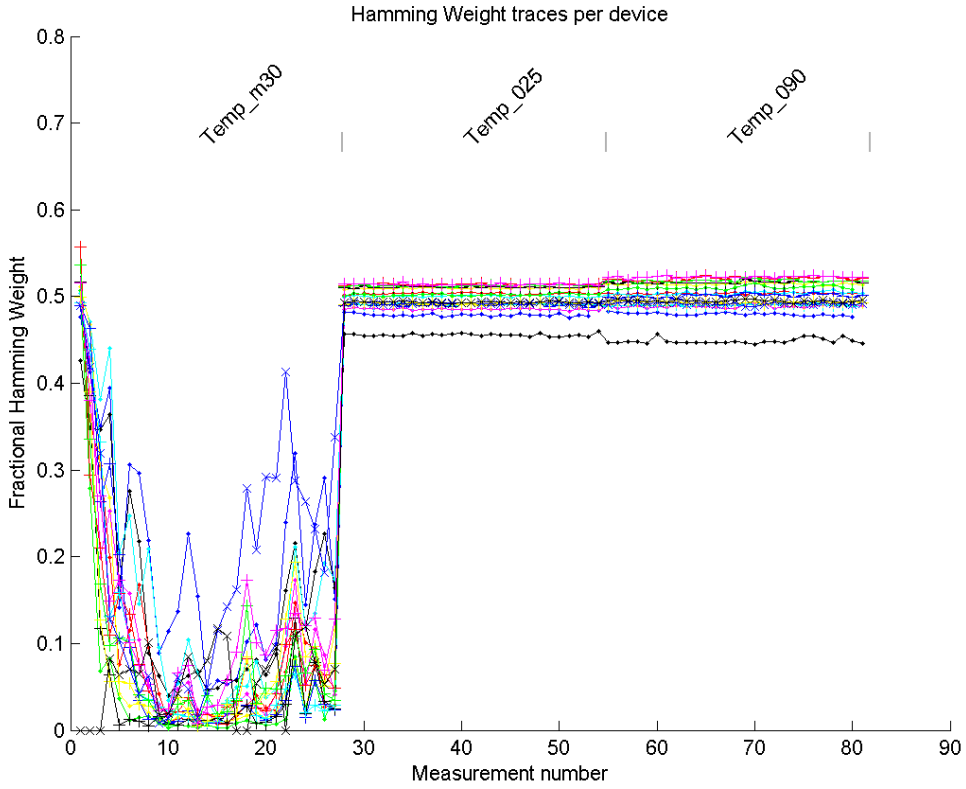


Figure 2.3.2: Hamming weight of SRAM in PIC16F1825 measurements over different temperatures.

ST STM32F100R8

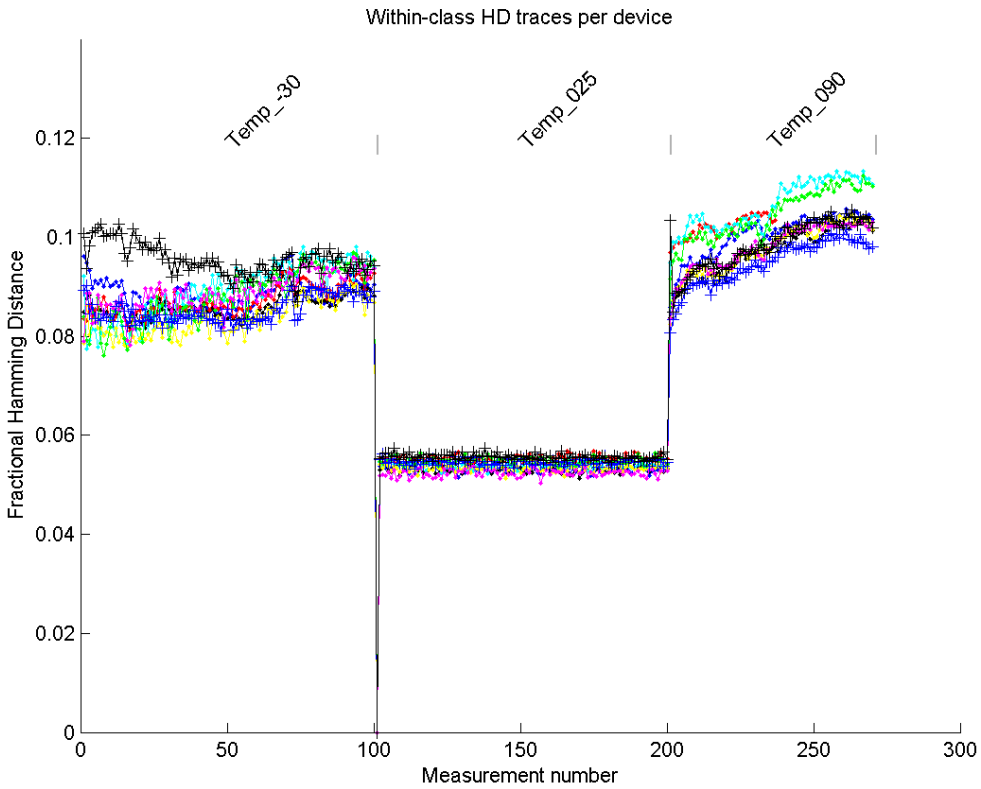


Figure 2.3.3: Within-class Hamming distance of SRAM in STM32F100R8 measured over different temperatures.

Figure 2.3.3 shows the results from the measurements of the Temperature Cycle Test for the 9 ST STM32F100R8 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 12%. This amount of noise can easily be corrected using commonly known Fuzzy Extractors. Therefore, **these devices pass the Temperature Cycle Test**.

Atmel ATmega328p

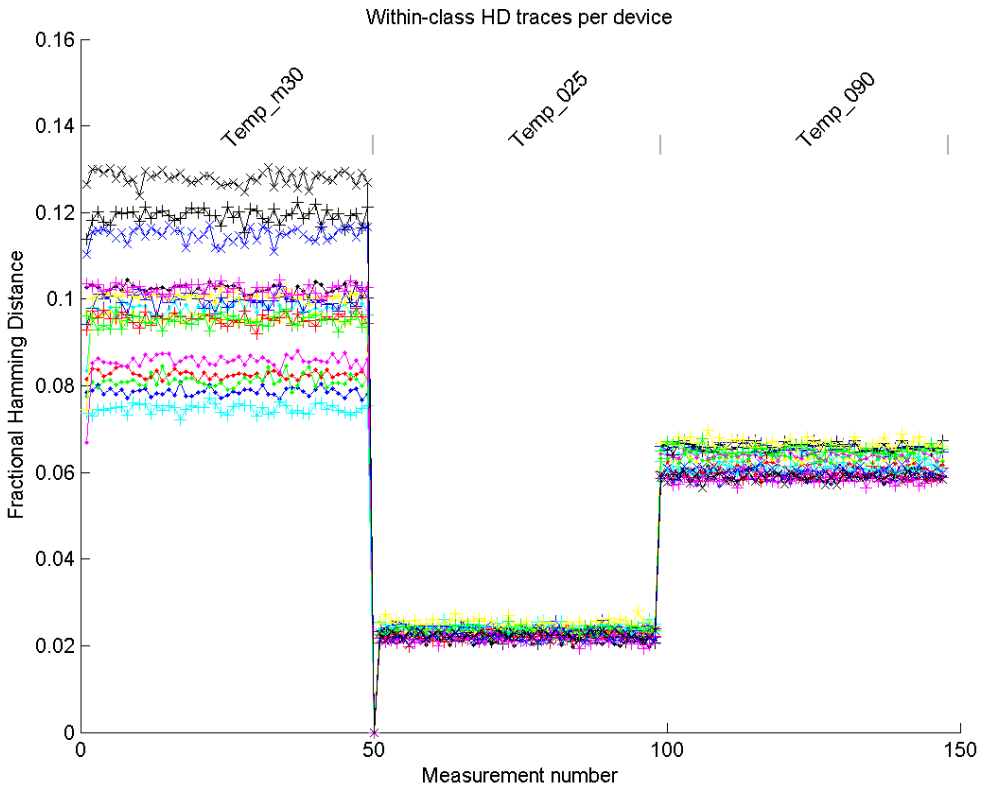


Figure 2.3.4: Within-class Hamming distance of SRAM in ATmega328p measured over different temperatures.

Figure 2.3.4 shows the results from the measurements of the Temperature Cycle Test for the 16 Atmel ATmega328p 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 14%. This amount of noise can easily be corrected using commonly known Fuzzy Extractors. Therefore, **these devices pass the Temperature Cycle Test**.

2.3.2 New data for platform from first project phase: NVIDIA GTX 295

Information

Number of devices measured:	510
Number of measurements per device:	19
PUF type:	SRAM PUF
PUF size:	16360 bytes

Repeated Start-up Test

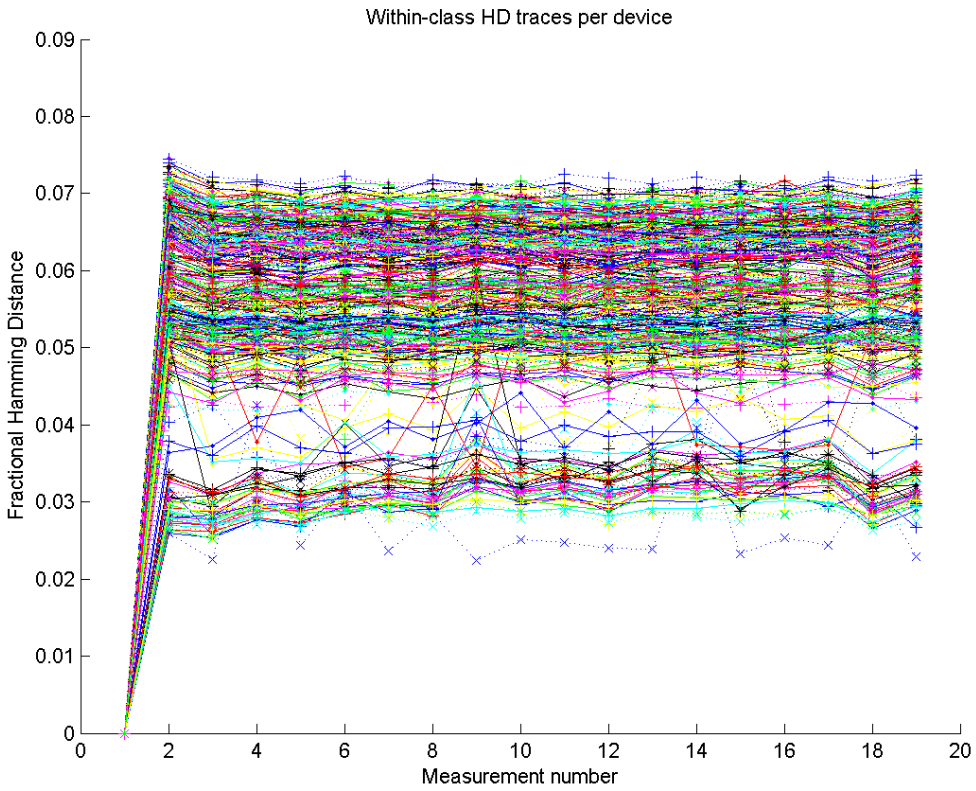


Figure 2.3.5: Within-class Hamming distance of SRAM in GTX 295 measurements.

Figure 2.3.5 shows the Hamming distances of 19 measurements of the Repeated Start-up Test on the 510 SRAMs from NVIDIA GeForce GTX 295 graphics cards (each individual line representing one of the SRAMs). 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 8%. This amount of noise can easily be corrected using commonly known Fuzzy Extractors. Therefore, **these devices pass the Repeated Start-up Test**.

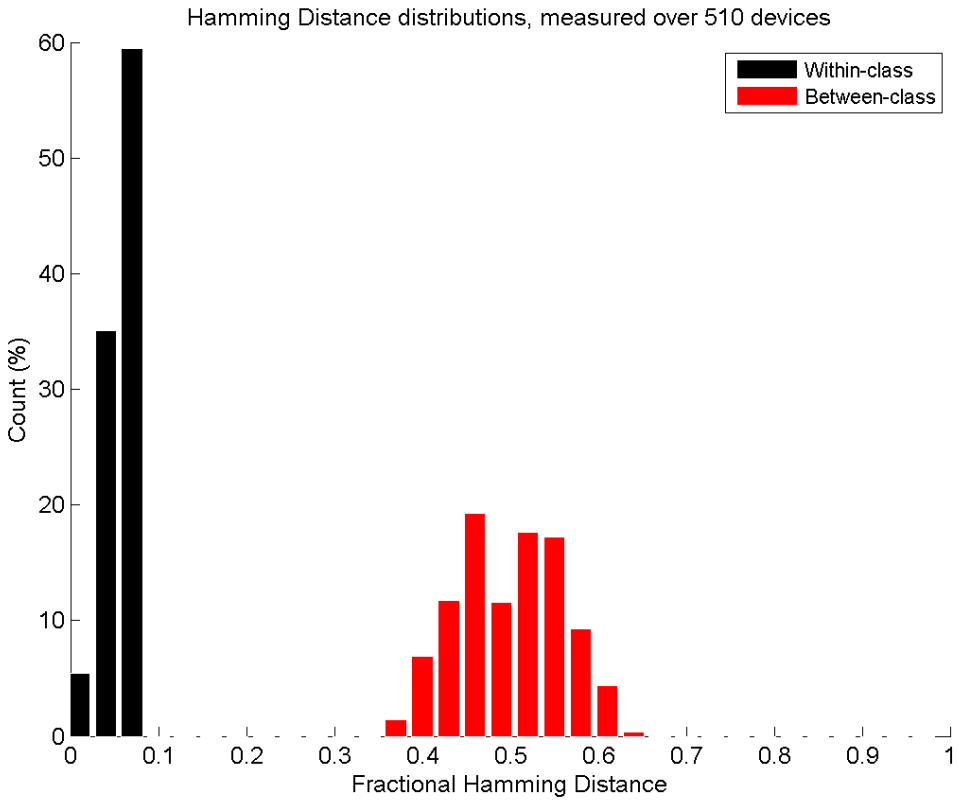


Figure 2.3.6: Between-class versus within-class Hamming distance of SRAM in GTX 295 measurements.

Between-Class Hamming Distance Test

Figure 2.3.6 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 510 SRAMs from NVIDIA GeForce GTX 295 graphics cards. 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.

Zooming in on the *between-class* distribution (see Figure 2.3.7), it becomes clear that this is not a typical distribution.This is because this distribution actually consists of two separate distributions, which are mirrored around 50%. Let's first analyse the left distribution, which is similar to a Gaussian distribution with a mean value around 45%. This indicates that the PUF responses of the different SRAMs are slightly correlated, since the mean value of the distribution is lower than 50%. Hence, there is some kind of pattern present in the SRAMs that slightly decreases the between-class Hamming distances. However, Figure 2.3.7 shows a second distribution with a mean value around 55%. Investigations have shown that this distribution occurs because half of the measured SRAMs show the inverse of the already mentioned pattern in their PUF responses. If two SRAMs are compared to each other, which have an identical pattern in their data, the between-class Hamming distance will be lower

than 50% and this value ends up in the left distribution. However, if two SRAMs with inverse patterns in their PUF responses are compared, their between-class Hamming distance will be higher than 50% (due to negative correlation) and this value will end up in the right distribution.

Hence, it is clear that there is some correlation (both positive and negative) between the PUF responses from different SRAMs of the GTX GPUs. However, these SRAMs are still suitable as an input for commonly known Fuzzy Extractors. This is because these algorithms are able to remove the correlated patterns from SRAM data in order to extract sufficient entropy from these SRAMs, under the assumption that they receive an amount of SRAM that is significantly higher than the length of the key that needs to be derived. Given that the SRAMs of the GTX CPUs are very large, sufficient SRAM is available to derive strong independent keys. Therefore, **these devices (weakly) pass the Between-Class Hamming Distance Test**.

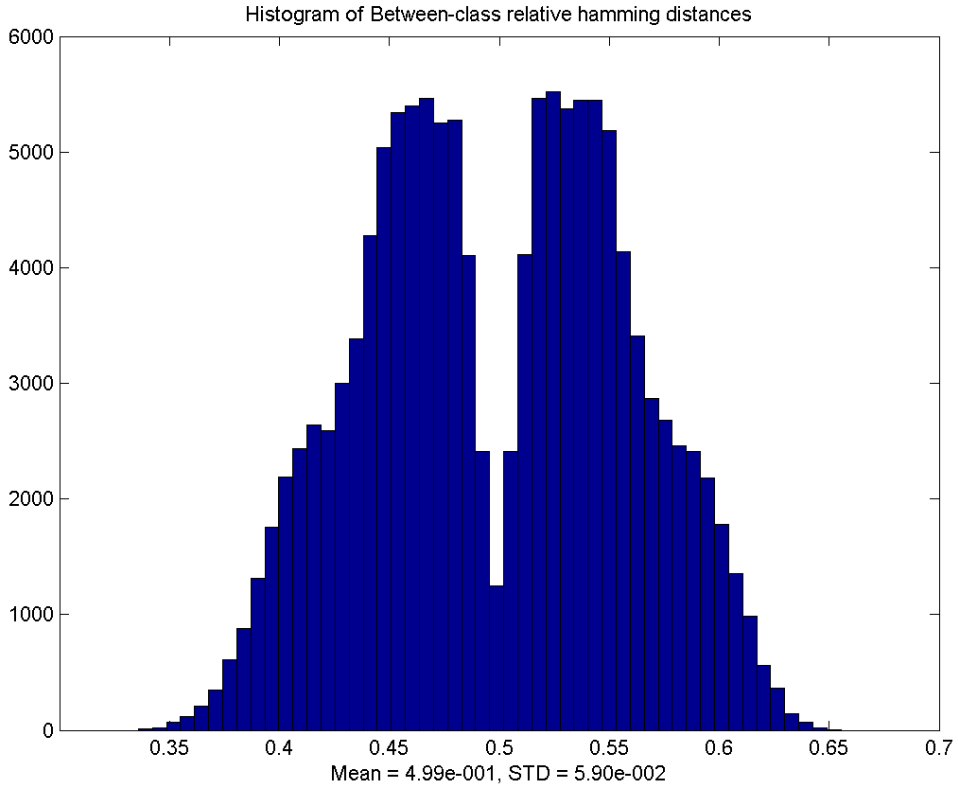


Figure 2.3.7: Between-class Hamming distance distribution of GTX 295 measurements.

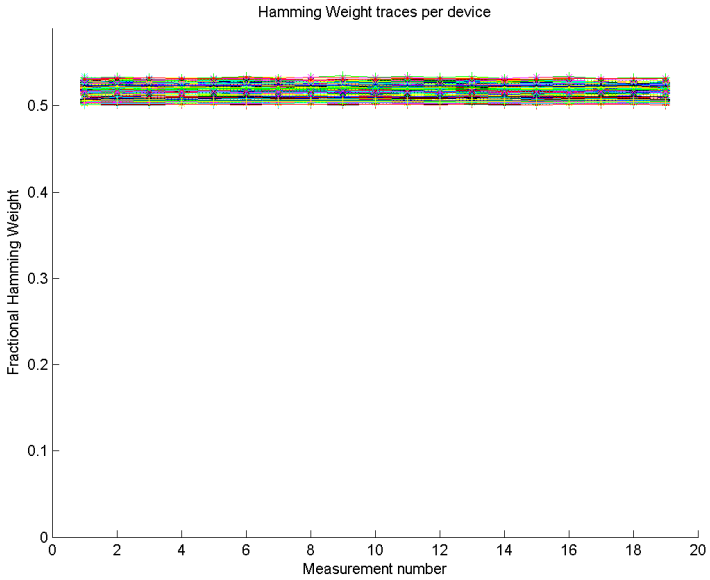


Figure 2.3.8: Hamming weight of SRAM in GTX 295 measurements.

Hamming Weight Test

Figure 2.3.8 shows the Hamming weight of the 19 measurements from the Repeated Start-up Test for the 510 SRAMs from NVIDIA GeForce GTX 295 graphics cards (each individual line representing one of the SRAMs). For all SRAMs, the fractional Hamming weight of the measurements is slightly higher than 50% (between 50% and 55%), i.e., there are more 1's than 0's in the PUF responses (an example PUF response is shown in Figure 2.3.9). 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, as stated before, these requirements can easily be fulfilled. Therefore, **these devices (weakly) pass the Hamming Weight Test**.

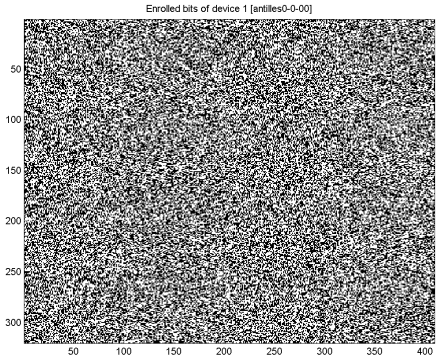


Figure 2.3.9: Example of SRAM PUF response from GTX 295 measurement.

2.3.3 New platforms: LM4F120H5QR and AM3358

Texas Instruments Stellaris LM4F120H5QR

Information

Number of devices measured:	15
Number of measurements per device:	1000
PUF type:	SRAM PUF
PUF size:	30KB

Repeated Start-up Test

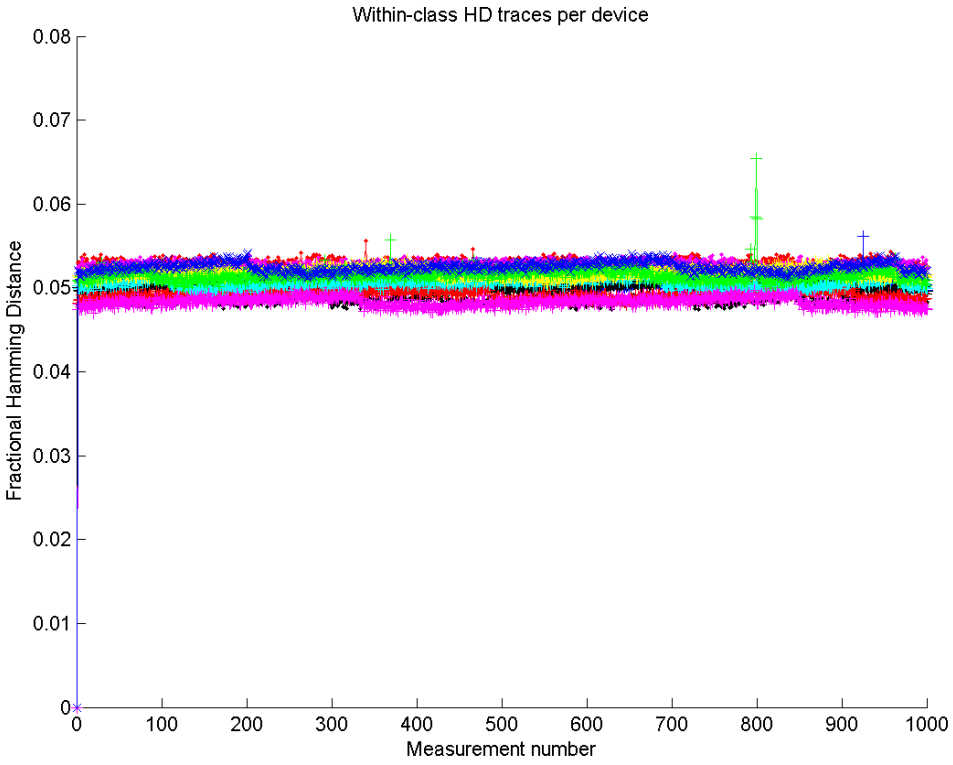


Figure 2.3.10: Within-class Hamming distance of SRAM in LM4F120H5QR measurements.

Figure 2.3.10 shows the results from the measurements of the Repeated Start-up Test for the 15 Texas Instruments Stellaris LM4F120H5QR 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 7%. This amount of noise can easily be corrected using commonly known Fuzzy Extractors. Therefore, **these devices pass the Repeated Start-up Test**.

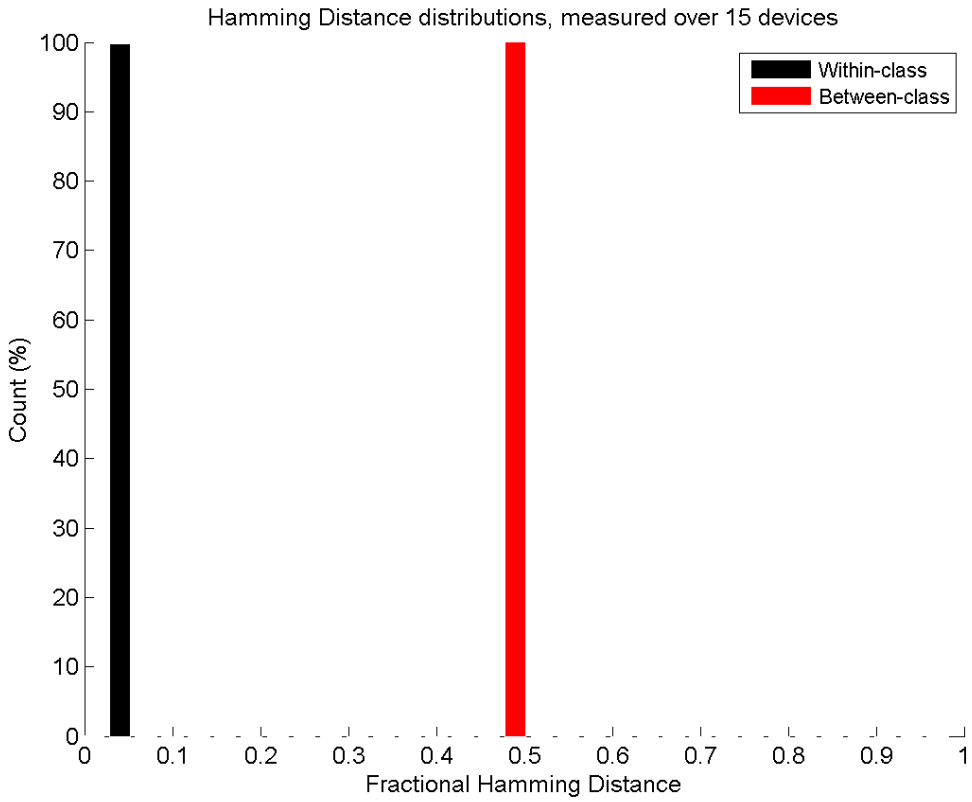


Figure 2.3.11: Between-class versus within-class Hamming distance of SRAM in LM4F120H5QR measurements.

Between-Class Hamming Distance Test

Figure 2.3.11 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 Stellaris LM4F120H5QR 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 49.9%. Since this value is approximately 50%, this is an indication that there is very little (to no) 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**.

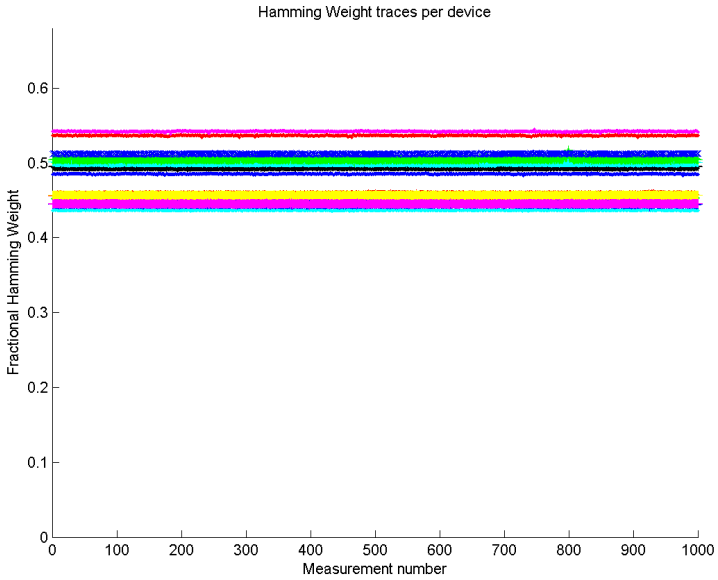


Figure 2.3.12: Hamming weight of SRAM in LM4F120H5QR measurements.

Hamming Weight Test

Figure 2.3.12 shows the Hamming weight of the measurements from the Repeated Start-up Test for the 15 Texas Instruments Stellaris LM4F120H5QR microcontrollers (each individual line representing 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 2.3.13). This indicates that these PUF responses are suitable inputs for commonly known Fuzzy Extractors. Therefore, **these devices pass the Hamming Weight Test**.

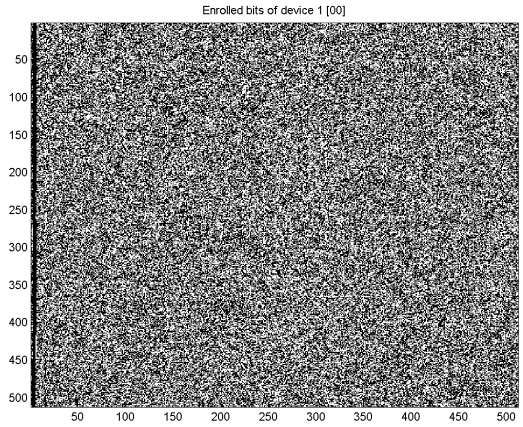


Figure 2.3.13: Example of SRAM PUF response from LM4F120H5QR measurement.

Texas Instruments AM3358 on BeagleBone board

Information

Number of devices measured:	3
Number of measurements per device:	1000
PUF type:	SRAM PUF
PUF size:	30KB

Repeated Start-up Test

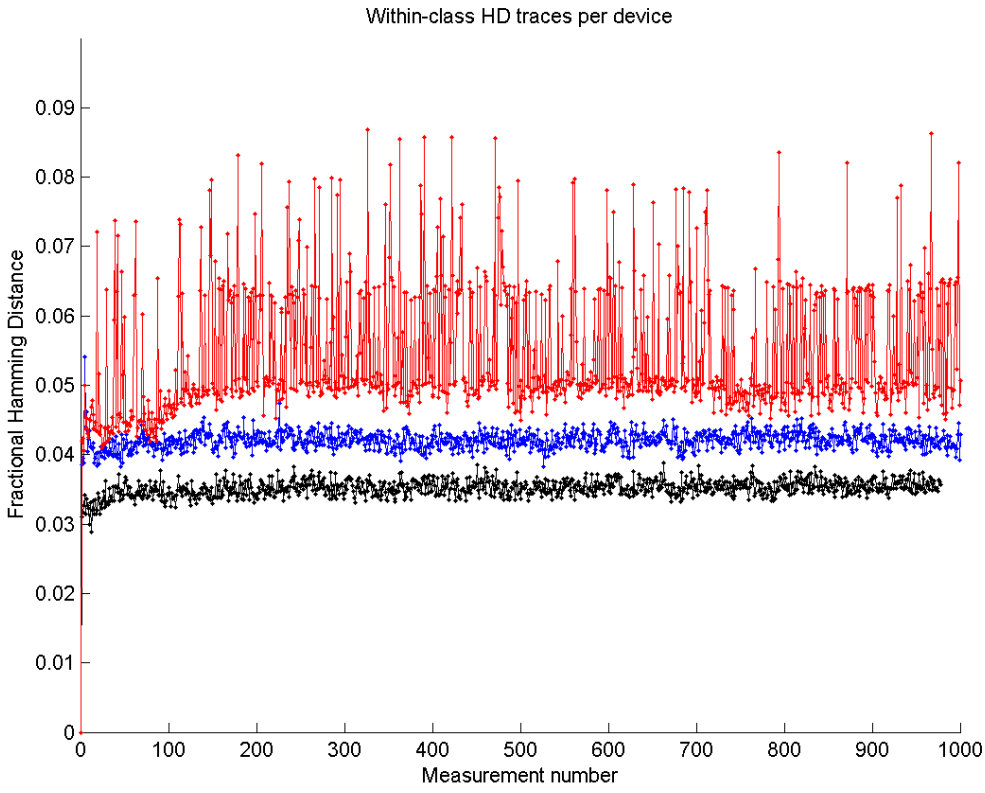


Figure 2.3.14: Within-class Hamming distance of SRAM in BeagleBone measurements.

Figure 2.3.14 shows the results from the measurements of the Repeated Start-up Test for the 3 Texas Instruments AM3358 microcontrollers on BeagleBone boards (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 9%. This amount of noise can easily be corrected using commonly known Fuzzy Extractors. Therefore, **these devices pass the Repeated Start-up Test**.

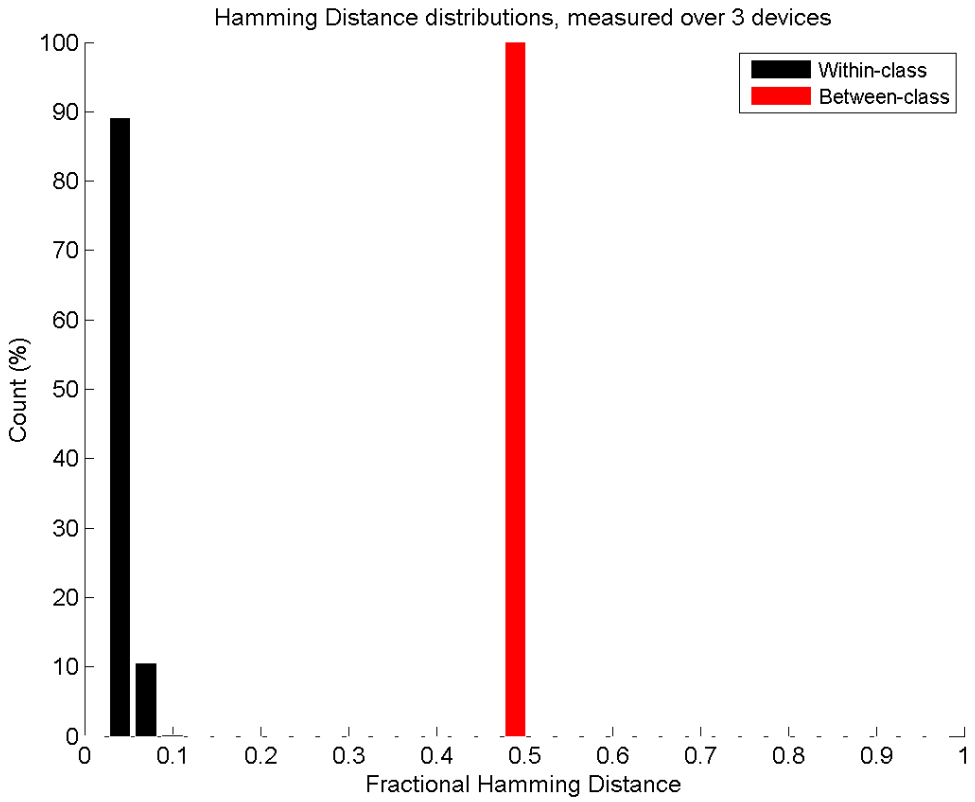


Figure 2.3.15: Between-class versus within-class Hamming distance of SRAM in BeagleBone measurements.

Between-Class Hamming Distance Test

Figure 2.3.15 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 3 Texas Instruments AM3358 microcontrollers on BeagleBone boards. 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 $3 \times 2 : 2 = 3$ Hamming distances between the 3 devices. These 3 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 are all around 49.4%. Since this value is quite close to 50%, this is an indication that there is very little correlation between the PUF responses from different devices (Figure 2.3.17 on the next page shows some slight patterning in the SRAM data, but this has very little influence on the Between-Class Hamming distances), which makes them suitable as an input for commonly known Fuzzy Extractors. Therefore, **these devices pass the Between-Class Hamming Distance Test**.

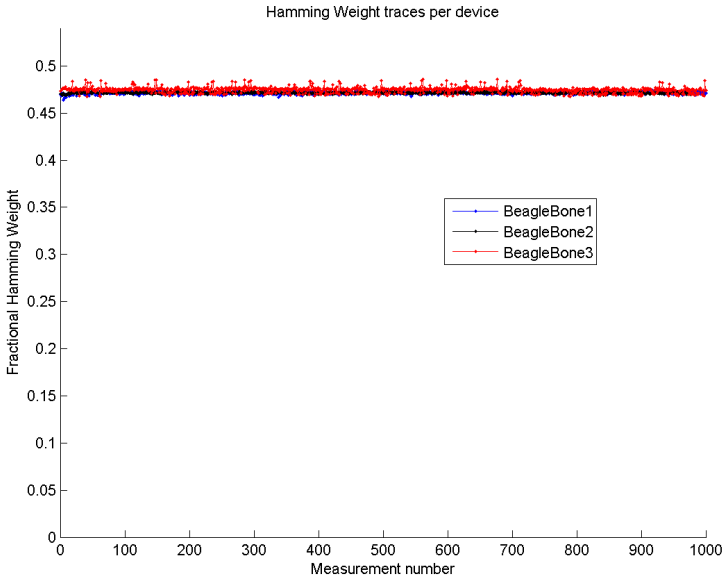


Figure 2.3.16: Hamming weight of SRAM in BeagleBone measurements.

Hamming Weight Test

Figure 2.3.16 shows the Hamming weight of the measurements from the Repeated Start-up Test for the 3 Texas Instruments AM3358 microcontrollers on BeagleBone boards (each individual line representing one of the devices). For all devices, the Hamming weight of the measurements is only slightly below 50%, indicating an almost equal number of 0's and 1's in the PUF responses (also visible in the plotted example PUF response 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**.

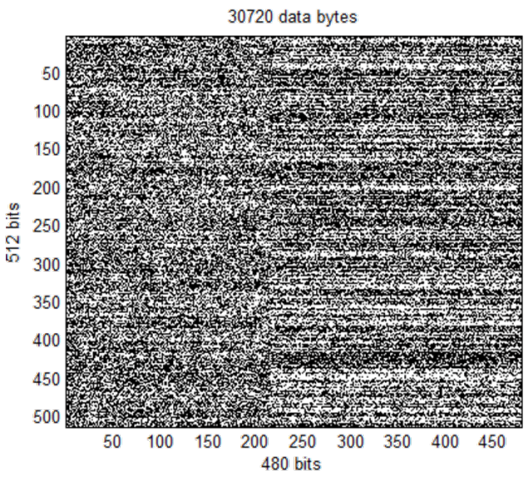


Figure 2.3.17: Example of SRAM PUF response from BeagleBone measurement.

Atmel ATmega1280 on Arduino Mega board

Information

Number of devices measured:	2
Number of measurements per device:	1000
PUF type:	SRAM PUF
PUF size:	8KB

Repeated Start-up Test

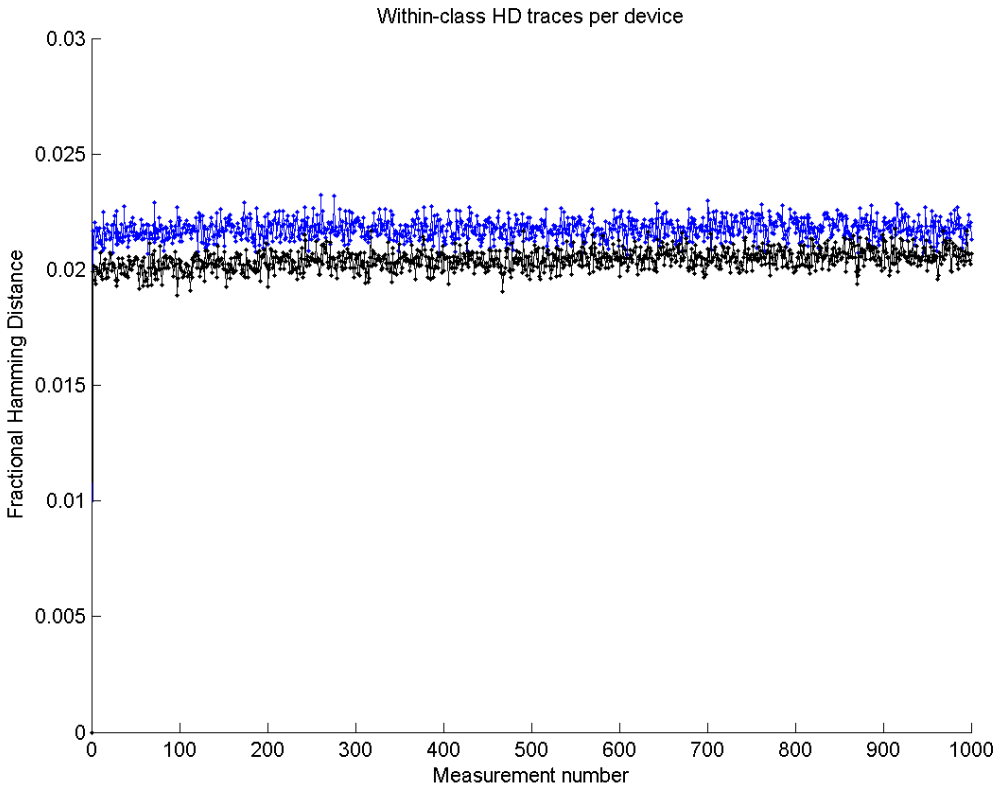


Figure 2.3.18: Within-class Hamming distance of SRAM in Arduino Mega measurements.

Figure 2.3.18 shows the results from the measurements of the Repeated Start-up Test for the 2 Atmel ATmega1280 microcontrollers on Arduino Mega boards (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 3%. This amount of noise can easily be corrected using commonly known Fuzzy Extractors. Therefore, **these devices pass the Repeated Start-up Test**.

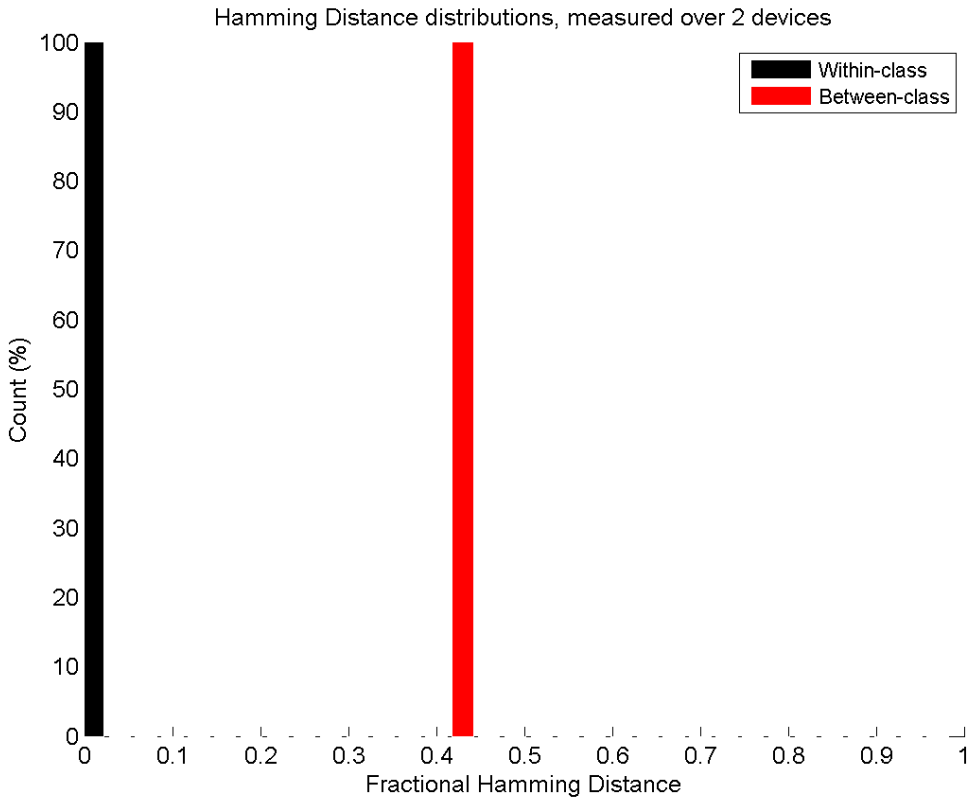


Figure 2.3.19: Between-class versus within-class Hamming distance of SRAM in Arduino Mega measurements.

Between-Class Hamming Distance Test

Figure 2.3.19 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 2 Atmel ATmega1280 microcontrollers on Arduino Mega boards. This figure clearly shows that the Hamming distance between the two 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 the Hamming distances between the two devices. It is clear that based on one distance, no distribution fitting can be performed. The distances between the two devices is 41.7%. 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. This is because these algorithms are able to remove the correlated patterns from SRAM data in order to extract sufficient entropy from these SRAMs, under the assumption that they receive an amount of SRAM that is significantly higher than the length of the key that needs to be derived. Given that the SRAMs of the ATmega1280 microcontrollers are large, sufficient SRAM is available to derive strong independent keys. Therefore, **these devices (weakly) pass the Between-Class Hamming Distance Test**.

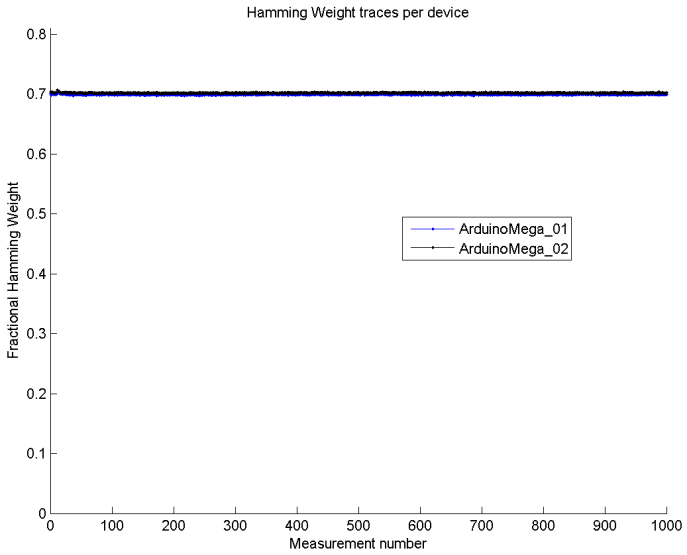


Figure 2.3.20: Hamming weight of SRAM in Arduino Mega measurements.

Hamming Weight Test

Figure 2.3.20 shows the Hamming weight of the measurements from the Repeated Start-up Test for the 2 Atmel ATmega1280 microcontrollers on Arduino Mega boards (each individual line representing one of the devices). For both devices, the Hamming weight of the measurements is significantly higher than 50% (around 70%), i.e., there are more 1's than 0's in the PUF responses (an example PUF response is shown in Figure 2.3.21, where the black vertical lines indicate initialized parts of the SRAM that have been excluded from the analysis). This indicates that these PUF responses require pre-processing in order to be suitable inputs for commonly known Fuzzy Extractors. This causes overhead in the size requirements for the PUF response; however, as stated before, these requirements can be fulfilled due to the large size of the SRAMs. Therefore, **these devices (weakly) pass the Hamming Weight Test**.

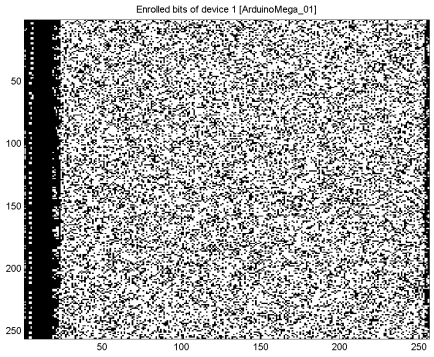


Figure 2.3.21: Example of SRAM PUF response from Arduino Mega measurement.

2.3.4 New test: Min-Entropy Tests on all measured devices

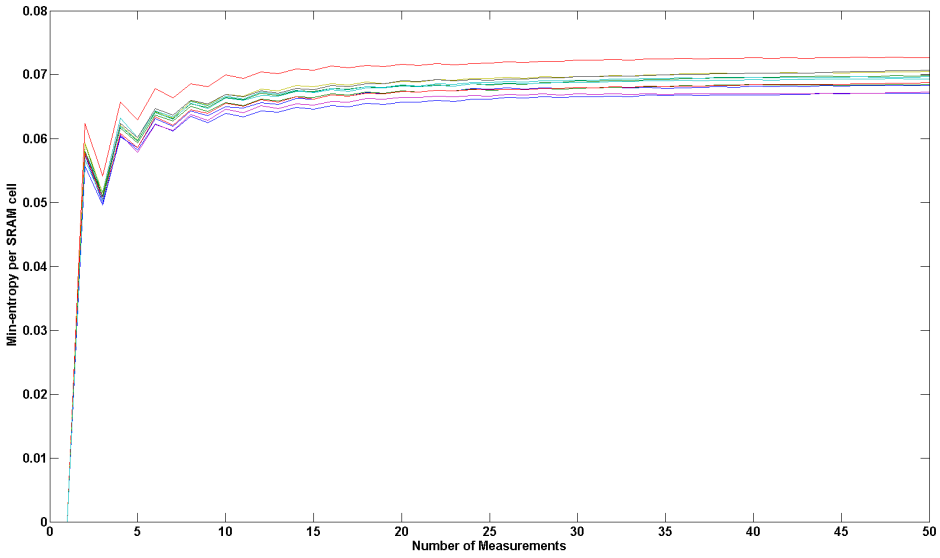


Figure 2.3.22: Min-entropy calculations for STM32F100RB based on 50 measurements.

As stated before, the Min-Entopy Test is used to evaluate whether measured memories are suitable for creating a high quality seed for the Random Number Generator that has been developed in WP3. For this purpose, the amount of entropy present in the noise of SRAM start-up patterns is determined, based on NIST specification SP800-90A.

Under the assumption that all bits are independent, the min-entropy of each individual SRAM cell can be summed to derive the min-entropy of the entire memory using Equations 2.1 and 2.2. The more measurements of a single device are available at a specific temperature, the more accurate the estimate of the probabilities will be and therefore the more accurate the min-entropy estimate will be. Plotting the min-entropy against the number of measurements used for the calculation will result in a function that increases with the number of measurements, up to a certain asymptotic value for the min-entropy. An example of such a plot can be seen in Figure 2.3.22 for 50 measurements of 11 devices of type ST STM32F100RB microcontrollers. In this example the minimal min-entropy after 50 measurements at room temperature over all devices is 6.7%.

Using this method, the minimal min-entropy for all devices measured during the PUFFIN project has been calculated. In Table 2.3.1 all results from the Min-Entropy Tests are listed. From these results it becomes clear that the amount of entropy that can be extracted from the noise on PUF responses varies between the different platforms and conditions.

The Microchip PIC16F1825 microcontrollers are not suitable for creating strong random seeds, due to the severe byte wise biasing (as already reported in Deliverable D2.1) and the observation earlier in this document that these SRAMs basically “freeze” at low temperatures resulting in all SRAM cells going to ‘0’. For the Texas Instruments MSP430F5308 microcontrollers it appears that the amount of min-entropy available also depends on the ambient

temperature. However, these memories do not freeze and therefore the consequences are less severe here than they are for the PIC16F1825.

For all other platforms it appears that the SRAMs are suitable for creating strong random seeds. However, the amount of SRAM required for such a seed varies significantly between the platforms. The higher the min-entropy of the noise is, the less SRAM is required for making the seed. Therefore, it is important to design the Random Number Generator with overhead on the amount of SRAM used in order to make sure that the amount of entropy will always be sufficient for creating a strong seed.

Example. In case a 128 bits random seed should be extracted from a GTX 295 SRAM, the minimal amount of SRAM required can be calculated as follows:

Min-entropy for GTX295 = 2.1% –> 128 / 0.021 = 6096 bits = 762 bytes.

Therefore, at least 762 bytes of GTX 295 SRAM should be used as input for a hash-function, which turns it into a 128 bits seed. However, in order to be on the safe side, it is recommendable to use some overhead for this purpose. For example use 1KB of SRAM, to make sure that under any (unpredictable) condition the amount of entropy available in the SRAM will be sufficient to create strong seeds.

Platform	$+25^oC$	-30^oC	$+90^oC$
Ainol Novo 7	5.4%	n.a.	n.a.
MSP430F5308	5.0%	2.3%	3.2%
PIC16F1825	1.7%	1.2%	2.1%
STM32F100R8	6.6%	5.3%	6.3%
STM32F100RB	6.7%	n.a.	n.a.
ATMega328p	2.7%	2.3%	3.2%
GTX 295	2.1%	n.a.	n.a.
Pandaboard	4.3%	n.a.	n.a.
LM4F120H5QR	5.9%	n.a.	n.a.
BeagleBone	4.0%	n.a.	n.a.
Arduino Mega	2.5%	n.a.	n.a.

Table 2.3.1: Minimum min-entropy of noise for all devices at different tested temperatures.

2.4 Conclusions

Platform	Quantity	RST	TCT	BCHDT	HWT	MET
Ainol Novo 7	7	Pass	n.a.	Pass	Pass	Pass
MSP430F5308	15	Pass	Pass	(Weak) Pass	(Weak) Pass	(Weak) Pass
PIC16F1825	16	Pass	**Fail**	**Fail**	**Fail**	**Fail**
STM32F100R8	11	Pass	Pass	(Weak) Pass	Pass	Pass
STM32F100RB	11	Pass	n.a.	(Weak) Pass	Pass	Pass
ATMega328p	16	Pass	Pass	(Weak) Pass	(Weak) Pass	Pass
GTX 295	510	Pass	n.a.	(Weak) Pass	(Weak) Pass	Pass
Pandaboard	5	Pass	n.a.	Pass	Pass	Pass
LM4F120H5QR	15	Pass	n.a.	Pass	Pass	Pass
BeagleBone	3	Pass	n.a.	Pass	Pass	Pass
Arduino Mega	2	Pass	Pass	(Weak) Pass	(Weak) Pass	Pass

Table 2.4.1: All test results for the different devices (phase 1 and 2)

Table 2.4.1 gives an overview of all test results from phase 1 and 2 of the PUFFIN project. 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.

As already discovered in the first phase of the project, the PIC16F1825 microcontroller SRAM measurements are not usable for PUF applications. Due to severe (byte wise) biasing of the PUF responses, these SRAMs do not provide enough entropy/uniqueness to be the basis for a proper PUF implementation. Also, the "freezing" of the SRAM, where at low temperatures all cells tend to start-up as '0', has extremely negative effects on both the reliability (during TCT) and uniqueness (during MET) of these SRAMs. After finishing the project, this remains the only device for which we could access uninitialized SRAM that was not suitable for PUF implementations.

During the second phase of the project, several devices from the first project phase have been tested at extreme temperatures. Besides additional negative results for the PIC16F1825 devices, the STM32F100R8 and ATMega328p did show that they are still usable as PUFs under extreme circumstances.

Furthermore, additional measurements on the NVIDIA GeForce GTX 295 graphics cards showed that these devices contain a lot of SRAM memories that are suitable for use as PUFs. Even though these SRAMs show slightly decreased entropy values, due to some pattern in the PUF responses, with sufficient SRAM and processing by the Fuzzy Extractor it is no problem to create PUF implementations on these devices.

New platforms that have been measured during the second project phase are the Texas Instruments Stellaris LM4F120H5QR microcontrollers, BeagleBone development boards, and Arduino Mega boards. The three microcontrollers have successfully passed all tests at room temperature, adding to the extensive portfolio of the PUFFIN project.

This Min-Entropy Test has also been performed on all platforms measured in the PUFFIN project. Conclusions here are that all platforms, except for the PIC16F1825, are suitable for

creating strong random seeds as long as the amount of SRAM used for such a seed is chosen based on the amount of min-entropy available.

All results from this WP2 work have been communicated to WP3 such that the obtained results could be used as input for the use-case implementations that have been developed in WP3. Recommendations that have been made to WP3 include:

- Results of data analysis in WP2 of the PUFFIN project has shown that it is possible to create good PUF implementations on a broad range of electronic hardware platforms. Devices in which PUFs have been found include microprocessors, tablets, smartphones, and GPUs. This shows that PUFs are actually much more commonly available than has been assumed up to now. The PUFFIN project has successfully shown that PUFs are already present on many COTS platforms and can be utilized with very limited overhead.

- Out of all studied platforms, the Pandaboard is the most interesting candidate for implementing prototypes in WP3. This platform passes all performed tests, but more importantly it is a computer development platform that offers a completely open-source Linux/Android development environment, including the possibility to modify its bootloader. This enables WP3 to obtain PUF data from the SRAM in a tablet/smartphone environment during the boot sequence of the operating system, which is a very interesting scenario for the PUFFIN project.

- Do not use the PIC16F1825 microcontroller for creating any PUF related demonstrators. The tests have shown that these devices are neither reliable nor unique enough to be considered decent PUFs.

- All platforms (except for PIC16F1825) that have been tested during the PUFFIN project show good prospects for creating strong seeds for Random Number Generators. This is a very encouraging result for adding a low-cost entropy source to resource constrained COTS devices. Given that this is serious problem in hardware security, this is an additional achievement of the PUFFIN project besides showing the possibility for PUF implementations in these devices.

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 were already developed during the first phase of the PUFFIN project. These two methods focussed on analysing PUF reliability and uniqueness respectively and have been described in Deliverable D2.1.

In the second phase of the PUFFIN project a new methodology has been developed (and applied) for analysing the aging behaviour of SRAM PUFs. This methodology has shown how SRAM will behave over time depending on different use case scenarios. From the results it has become clear that it is possible to influence the aging behaviour of SRAM PUF depending on how data is stored in these SRAMs when the device is powered. This way it is possible to make sure that SRAM PUFs will behave reliably over time, regardless of how long the lifetime of a device is.

A paper has been written and published about this new analysis methodology and the measures required for countering the effects of silicon aging for SRAM PUFs in "Countering the Effects of Silicon Aging on SRAM PUFs", by Roel Maes and Vincent van der Leest (both Intrinsic-ID). The paper has been published at the IEEE International Symposium on Hardware-Oriented Security and Trust (HOST) 2014.

For this paper measurements have been performed using devices designed and produced in the FP7 project UNIQUE (contract number: 238811). Reason for this is that the UNIQUE devices and test boards have been designed specifically for straining tests under extreme circumstances, like the described aging tests. In case we would have used COST devices from the PUFFIN project, it is highly likely that the devices or test boards would have broken down during the runtime of these aging tests. Therefore, it was much more appropriate to use the UNIQUE devices and boards for these tests. Even though tests have been performed on other devices than COTS, the results are still completely valid for the PUFFIN devices as well.

To provide a complete overview of the work that has been performed on (as well as the results from) developing the new analysis methodology, the above mentioned paper has been attached to this deliverable as an appendix.

Furthermore, a second appendix has been added to this document containing a paper that has been submitted by members of the PUFFIN project to the Journal of Cryptographic Engineering. This paper contains an overview of the work had been performed in the PUFFIN

project on analysing PUF data that have been gathered during the course of this project. Therefore this paper should not be excluded from the final overview of WP2, given that is the most important dissemination of the results from WP2 to the scientific community. It does not contain any new information compared to the Deliverables D2.1 and D2.2, but it provides a very nice overview of the results. The paper is currently still under review by the journal.

Appendix A

Paper: "Countering the Effects of Silicon Aging on SRAM PUFs"

Author: Roel Maes and Vincent van der Leest (Intrinsic-ID)
Venue: IEEE International Symposium on Hardware-Oriented Security and Trust (HOST) 2014
Date: May 6th - May 7th, 2014

Status: Published

Countering the Effects of Silicon Aging on SRAM PUFs

Roel Maes
Intrinsic-ID
Eindhoven, The Netherlands
roel.maes@intrinsic-id.com

Vincent van der Leest
Intrinsic-ID
Eindhoven, The Netherlands
vincent.van.der.leest@intrinsic-id.com

***Abstract*—Silicon aging, in particular NBTI, causes many PUFs to exhibit a natural tendency of growing less reliable over time. This is inconvenient or even unacceptable for in-the-field applications. In case of SRAM PUFs it is observed that the impact of NBTI aging depends on the data stored in the SRAM. In this work, we investigate the effects of data-dependent silicon aging on SRAM PUF reliability under a number of realistic scenarios. In an accelerated aging experiment on a 65nm CMOS SRAM PUF implementation it is observed that many scenarios cause a smaller reliability reduction than natural aging. Some scenarios even show *anti-aging* effects, i.e. they cause the SRAM PUF to grow more reliable over time. This is a significant improvement when using an SRAM PUF. Even more so because data-dependent (anti-)aging has a particularly low overhead, requiring neither any changes to the PUF circuit nor any pre-deployment effort.**

I. Introduction

A physically unclonable function or PUF implemented on a silicon integrated circuit (IC) can be used as a hardware root-of-trust for the digital system running on that IC, e.g. to generate and store the system's master encryption keys. Such PUF-based key generators are increasingly being deployed in digital security products [1], [2], [3] since they often outcompete traditional non-volatile memories (e.g. Flash, EEPROM, anti-fuses, etc.) as highly secure yet efficient key storage solutions.

For such applications, a high-quality PUF is needed which is both unpredictable as well as reliable, i.e. PUF responses are random per instantiation but repeatable with limited noise over time and under all circumstances. It is well known that a PUF's operating conditions such as environment temperature and supply voltage affect its reliability. PUF implementations are therefore tested under varying conditions to determine their *worst-case reliability*, typically occurring at high temperature and voltage [4]. An application like a PUF-based key generator needs to deal with (and hence be designed for) this worst-case reliability to ensure a failure-free operation in the field.

Another variable affecting the behavior and hence the reliability of a PUF implementation is the effective lifetime of the IC. Certain physical phenomena in a silicon IC cause a circuit's parameters to slowly change over time, mostly degrading its performance and even leading to failures. The accumulated effect of these phenomena is called *silicon aging*. Because PUF responses take their randomness from minute process variations in the circuit's parameters, it is evident that silicon aging affects and usually also degrades a PUF's reliability over time [5]. A particular consequence hereof is that the worst-case reliability of a typical PUF construction is not only to be found at high temperatures and voltages, but also at a point in the future at the end of the device's lifetime, after years of silicon aging. This makes it non-trivial to estimate the worst-case reliability. Moreover, the required effort to obtain such an estimate often results in silicon aging being (inappropriately) ignored as a reliability degradation factor in evaluations of PUF implementations.

Based on physical models we can to a certain extent predict or simulate the expected effect of silicon aging. Another option is to run a silicon device at elevated temperature and voltage which accelerates the aging phenomena, and allows us to measure the effect after a significantly shorter amount of time. Based on these methods, an estimated guess of the worst-case reliability at the end of the device's lifetime can be made.

As an alternative to designing a PUF-based application for predicted future worst-case reliability, a radically different approach can be taken. As we will demonstrate in this work, for certain PUF constructions (in particular SRAM PUFs) it is possible to slow down, halt, or even reverse the effects of silicon aging on the PUF response reliability. The latter has the tremendous advantage that the PUF's worst-case reliability is then to be found at the beginning of the device's lifetime and can be measured immediately after manufacturing. Over time the PUF's reliability will now stay constant or even improve which means that the reliability requirements for the PUF-based application can be significantly relaxed, resulting in a gain in efficiency (e.g. less complex error-correcting codes).

Related Work: An SRAM PUF, as proposed by Guajardo et al. [6], is a PUF construction based on the power-up state of an SRAM array. SRAM PUFs have been thoroughly studied and invariably show high-quality PUF behavior, see e.g. [4], [7]. However, it was also shown by Maes et al. [5] that without precautions, SRAM PUF reliability does suffer from silicon aging. Bhargava et al. [8], [9] were the first to study the potential beneficial effects of silicon aging phenomena on the reliability of PUF structures, including SRAM PUFs. In [8] they demonstrate that a post-manufacturing burn-in stress (high temperature and high voltage) applied for 120 hours reduces the initial amount of bit errors of an SRAM PUF by 40%. However, such a long pre-deployment burn-in time does represent a large overhead in the manufacturing flow

of a typical silicon IC. Moreover, once in the field their SRAM PUF is again subject to regular silicon aging and its reliability will deteriorate over time.

Our Contributions: In this work, we present:

- *Anti-aging* techniques for SRAM PUFs which are purely based on data-dependent silicon aging effects in regular SRAM cells during the regular lifetime of the IC. As a consequence, these techniques are directly usable on any standard SRAM (no circuit changes) without any pre-deployment overhead (no burn-in time).
- An overview and experimental validation (in 65nm CMOS) of a number of (anti-)aging scenarios differing only in the circumstances for generating the *anti-aging data*, and an assessment of their effect on the SRAM PUF's reliability over time.
- The identification of *ideal* anti-aging scenarios which effectively improve the SRAM PUF's reliability over the lifetime of the IC, with a reduction of 0.35% in the average amount of bit errors for the most optimal case.

Paper Outline: In Sect. II we provide some background on silicon aging, in particular the aging effects which enable data-dependent *anti-aging* for SRAM PUF cells. Next, in Sect. III we introduce a number of plausible (anti-)aging scenarios for SRAM PUFs and experimentally validate them on a 65nm CMOS test ASIC in an accelerated aging experiment. The most important findings of this experiment are discussed in Sect. IV and finally we conclude in Sect. V.

II. Background

A. PUF Quality Measures

The quality of a PUF is quantified by a number of experimentally verifiable measures. A PUF's *reliability* is measured by its average *intra-distance*, i.e. the (Hamming) distance between multiple evaluations of the same response on the same PUF instance. This is a good estimate of the number of bit errors one can expect in a response evaluation and is preferably very small. PUF response *uniqueness* on the other hand is measured by the average *inter-distance*, i.e. the distance between responses evaluated on different PUF instances. For binary response values, the average inter-distance is preferably very close to 50% of the response length. A PUF's *unpredictability* is ideally measured by evaluating the response *entropy*, but due to limitations on the amount of available experimental data this is often difficult to do in a meaningful manner. However, a necessary condition for a high response entropy is a low response bias which can be efficiently and accurately measured by calculating the average *(Hamming) weight* of the response vectors. Ideally the average response Hamming weight is also close to 50% of the response length.

B. Aging Effects in CMOS Silicon

The nominal operation of a silicon IC has a number of unintended but unavoidable side-effects which result in permanent physical alterations to the circuit's physical structure. Hence, an operational IC slowly but gradually changes over time, i.e. it *ages*. Eventually the induced physical changes affect the circuit's operation, typically in a degrading manner, and ultimately even lead to circuit failures after a long period. One dominant aging effect in modern ICs is negative-bias temperature instability (NBTI) causing a gradual increase in the threshold voltage, which is most evident in switched-on PMOS transistors. Other independent physical aging phenomena in CMOS devices are hot-carrier injection (HCI), time-dependent dielectric breakdown (TDDB) and electromigration (EM). For an overview of silicon aging effects we refer to [10].

C. SRAM PUFs and Data-Dependent (Anti-)Aging

An SRAM PUF [6] evaluates the power-up pattern of a standard 6T SRAM array. As shown in Fig. 1, at its core each SRAM cell in the array comprises two nominally matched CMOS inverters which are cross-coupled. Uncontrollable CMOS process variations introduce random parameter deviations which cause a mismatch between the inverter pairs affecting their power-up state. The predominant mismatch in an SRAM cell determining its power-up state is the difference between the threshold voltages (V_{th}) of both PMOS transistors P1 and P2. E.g. consider the case when random variations cause $V_{\mathrm{th,P1}}$ to be slightly smaller than $V_{\mathrm{th,P2}}$. As a result, at power-up (rising V_{dd}) P1 will start conducting before P2, causing A to go logically high and preventing P2 from switching on. The power-up state of the cell is hence $A = 1$. The larger the mismatch between $V_{\mathrm{th,P1}}$ and $V_{\mathrm{th,P2}}$, the stronger the power-up preference of a cell and hence the smaller the probability to power-up in its *wrong* state causing a PUF response bit error. Extensive experiments [4], [7] have demonstrated that due the independent random nature of process variations on each SRAM cell, the power-up pattern of an SRAM array demonstrates excellent PUF behavior, i.e. small intra-distances and both inter-distances and Hamming weights close to 50%.

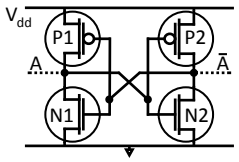


Fig. 1: Cross-coupled CMOS inverter circuit at the core of each SRAM cell. (Two SRAM access MOSFETs not shown.)

The effect of NBTI aging for SRAM cells depends on the bit value stored in the cell. When the cell stores a zero ($A = 0$), P1 is switched off and P2 is switched on. As a result, $V_{\mathrm{th,P2}}$ will increase over time due to NBTI while $V_{\mathrm{th,P1}}$ is unaffected. For $A = 1$ the opposite effect occurs. Combined with the power-up behavior, the situation is such that the PMOS with the smallest V_{th} tends to turn on at power-up and will subsequently experience a gradually increasing V_{th} due to NBTI. The natural tendency of an SRAM cell is hence to *age* such that $|V_{\mathrm{th,P1}} - V_{\mathrm{th,P2}}|$ grows smaller over time. From an SRAM PUF perspective, this is a disadvantage since a decreasing $|V_{\mathrm{th,P1}} - V_{\mathrm{th,P2}}|$ means a higher probability of a PUF response bit error. In other words, SRAM PUFs

tend to become less reliable over time due to silicon aging, as was experimentally observed, e.g. in [5]. Luckily, this disadvantageous tendency can be counteracted. An evident solution is to let each cell store the inverse of its power-up value, since this would in general increase $|V_{\mathrm{th,P1}} - V_{\mathrm{th,P2}}|$ and hence make the corresponding SRAM PUF response bit effectively more reliable over time. This effect is called *anti-aging* in the context of (SRAM) PUFs.

III. Experiment: (Anti-)Aging in SRAM PUFs

A. Motivation

From Sect. II it is clear that the effect of silicon aging (in particular NBTI) on SRAM PUF reliability depends heavily on the (long-term) data stored in the SRAM, ultimately even permitting anti-aging. This observation gives rise to a number of interesting issues and questions worth investigating, e.g.:

- SRAM power-up states partially change between power-ups (intra-distance). Which particular state needs to be inverted to serve as anti-aging data? What happens when this exact state can not or only partially be reconstructed?
- When used as PUF response, SRAM power-up data is typically security-sensitive information. Which anti-aging options are still possible when all SRAM power-up information (including its inverse) needs to be erased?
- What are the (anti-)aging effects when the SRAM is used for other means after having served its PUF purpose?
- What is the effect of data-dependent (anti-)aging on other quality measures of SRAM PUFs, besides reliability?

These questions each represent different possible (anti-)aging scenarios for SRAM PUFs, which will be experimentally studied in this section.

B. (Anti-)Aging Scenarios

We start by listing the different (anti-)aging scenarios which we will test on a 65nm CMOS implementation of an SRAM PUF. For each scenario we detail the method used to generate the anti-aging data (i.e. the data which is long-term stored in the SRAM) and motivate the scenario by explaining how it can arise in a realistic use case. We consider the typical usage of a PUF in which an initial response measurement, called *enrollment*, is compared to or reconstructed from a later in-the-field measurement, called *reconstruction*. In the case of a key generator, the reconstruction measurement needs to be error-corrected to obtain the original enrollment response from which the key is derived.

1) No Anti-Aging: **NO_AA**

Scenario: The long-term data stored in the SRAM are the (noisy) SRAM power-up states at reconstruction.

Motivation: This is the trivial scenario where no action is taken (no anti-aging, no clearing, nor any other use of the SRAM), and the reconstruction power-up data is maintained unmodified in the SRAM array for the whole time the SRAM is powered. This scenario is mostly considered as a reference for the other scenarios. As derived in Sect. II, this is assumed to be the pessimistic scenario which fully exhibits the natural tendency of an SRAM PUF to grow less reliable over time.

2) Full Anti-Aging: **FULL_AA**

Scenario: Using error-correction techniques, the initial enrollment power-up state is perfectly reconstructed from a (noisy) reconstruction measurement. The long-term anti-aging data is the inverse of the corrected enrollment state.

Motivation: As derived in Sect. II, this is assumed to be the optimal scenario where the enrollment power-up state is continuously reinforced and the reliability of the SRAM PUF (w.r.t. the enrollment response) improves over time. This scenario can be applied straightforwardly in a PUF-based key generator which perfectly regenerates the enrollment response.

3) Partial Correction Anti-Aging: **PART_AA**$(xx\%)$

Scenario: Using error-correction techniques, the initial enrollment power-up state is *partially* reconstructed from a (noisy) reconstruction measurement, i.e. a certain fraction ($xx\% = 0\% \ldots 100\%$) of the bit errors in the reconstruction measurement is corrected. The long-term anti-aging data is the inverse of this partially corrected enrollment state.

Motivation: This is a variant of **FULL_AA** which describes the scenarios where for particular reasons the enrollment response is not perfectly reconstructed. For certain PUF-based applications it is often not required, inconvenient or impossible to perfectly reconstruct the SRAM enrollment power-up state, e.g. when the PUF response is used as a noisy identifier.

These first three scenarios (**NO_AA**, **FULL_AA**, **PART_AA**) are conceptually visualized in Fig. 2.

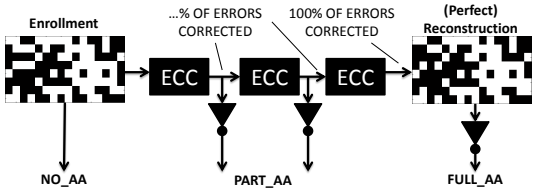


Fig. 2: Visualization of **NO_AA**, **FULL_AA**, and **PART_AA**.

4) Multiple Correction Anti-Aging: **MULT_AA**

Scenario: Using error-correction, one out of a set of initial enrollment SRAM power-up states is perfectly reconstructed from a reconstruction measurement. The long-term anti-aging data is the inverse of the selected enrollment state.

Motivation: This scenario arises when the same SRAM is enrolled as a PUF more than once, e.g. to derive a different PUF-based key for different users, applications, etc. Under normal circumstances, the different recorded enrollment states differ only due to noise on the PUF response bits.

5) Random Anti-Aging: **RAND_AA**

Scenario: The long-term anti-aging data is a random bit string. We differentiate between **RAND_AA(fix)** which uses a fixed random string after each reconstruction on each PUF instance, and **RAND_AA(true)** which uses a different randomly generated string after every reconstruction on every device.

Motivation: This is an (anti-)aging strategy that is independent of the power-up state. It avoids potential attack on security-sensitive data by erasing all PUF response information after it is used. Also it does not require error-correction.

6) Structured Anti-Aging: **STRUCT_AA**

Scenario: The long-term anti-aging data is a structured bit string, i.e. the anti-aging value of an SRAM cell is a deterministic function of the cell's memory address. The examples of specific data structures that we consider are:

1) **STRUCT_AA(zero)/STRUCT_AA(one)**: store zeros or ones in every bit location of the SRAM matrix.
2) **STRUCT_AA(row)**: alternately store zeros and ones in consecutive rows of the SRAM matrix.
3) **STRUCT_AA(checker)**: store a *checkerboard* pattern of zeros and ones in the SRAM matrix.

Motivation: Equal to **RAND_AA**, without the requirement for access to an embedded seeded PRNG or TRNG.

7) Dynamic Anti-Aging: **DYN_AA**

Scenario: After reconstruction, dynamically and constantly write data to the SRAM. We differentiate between **DYN_AA(rand)** which continuously writes randomly generated data, and **DYN_AA(fix)** which continuously cycles between a number of fixed patterns.

Motivation: This scenario arises when the SRAM used as a PUF, is also used for different purposes after key reconstruction, e.g. as an instruction or data memory. This scenario also captures a possible alternative (anti-)aging scenario which is independent of the power-up state.

C. Test Setup

As detailed in Sect. II, the major aging mechanism causing SRAM PUF reliability to change over time is NBTI. The effects of NBTI aging on a silicon device are considerably accelerated when the device is operated at increased temperature and/or supply voltage with respect to its nominal conditions [10]. The amount of acceleration is captured by an *acceleration factor* (AF) which depends on the accelerated aging conditions. For NBTI aging, AF is calculated as follows [10, Sect. 5.3]:

$$AF = \left(\frac{V_{stress}}{V_{nominal}}\right)^{\frac{\alpha}{n}} \cdot \exp\left(\frac{E_{aa}}{k} \cdot \left(\frac{1}{T_{stress}} - \frac{1}{T_{nominal}}\right) \cdot \frac{1}{n}\right).$$

The used model parameters for NBTI accelerated aging are the following: the gate voltage exponent $\alpha = 3.5$; the time exponent $n = 0.25$; the apparent activation energy $E_{aa} = -0.02eV$; and Boltzmann's constant $k = 8.62 \times 10^{-5} eV/K$. The nominal and stressed aging conditions we consider in our accelerated aging experiment are as follows: $(T_{nominal}, V_{nominal}) = (40°C, 1.2V)$ and $(T_{stress}, V_{stress}) = (85°C, 1.44V)$. The resulting NBTI acceleration factor under these conditions becomes $AF = 18.6$, i.e. one hour of accelerated aging amounts to 18.6 hours of effective NBTI device aging under nominal conditions (assuming that AF is constant over time).

We applied this accelerated aging experiment on five 65nm CMOS test ICs each implementing four SRAM PUFs of size 8.0KByte. Prior to starting the aging experiment, several measurements on each of these 20 PUFs have been performed at $25°C$ ambient temperature and $1.2V$ supply voltage. These measurements represent the PUFs' initial states after zero weeks of aging. After these measurements, the temperature and voltage were increased to their stress levels. Once every week during the experiment the PUFs are remeasured at $25°C$ and $1.2V$ respectively to acquire a data set at nominal conditions for every week of accelerated aging. This accelerated aging experiment has run for 2856 consecutive hours.With an (assumed constant) acceleration factor of 18.6 this amounts to an effective NBTI aging of more than 6 years.

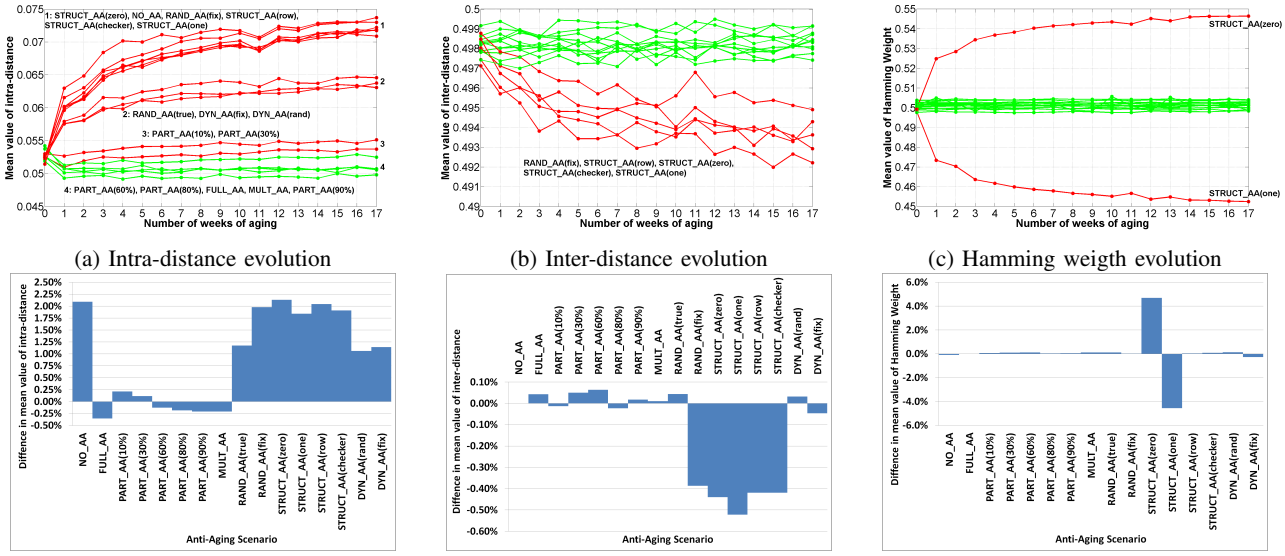
To evaluate the effect of the different (anti)-aging scenarios described in Sect. III-B, every SRAM PUF in the experiment is divided in 16 equal sections of size 0.5KByte each implementing one of the scenarios. Every SRAM PUF is repowered, and hence re-evaluated, every six hours during the experiment to emulate a realistic usage, followed by the data actions prescribed by the different scenarios for each SRAM section. For most scenarios this means that the SRAM section is written with the scenario's data which remains in the memory for the next six hours of aging. For the **NO_AA** section no action is taken and the power-up data after every re-power is untouched. Special scenarios are **MULT_AA** for which a new inverse enrollment pattern is written every 80 minutes, and the dynamic scenarios **DYN_AA(rand)** and **DYN_AA(fix)** which are continuously overwritten. In Table I (which also already summarizes the results of these tests) an overview of the 16 sections and their respective (anti-)aging scenarios is presented. By evaluating the aging behavior of each of these sections individually, it is possible to analyze which scenarios are suitable for dealing with specific circumstances.

TABLE I: Considered (anti-)aging scenarios in our accelerated aging experiment, and the corresponding summarized results. + : quality measure is stable or improves (++ is best scenario) – : quality measure degrades (−− is worst scenario).

#	Scenario	Written every...	Intra-distance	Inter-distance	Hamming weight
1	**NO_AA**	/	−−	+	+
2	**FULL_AA**	6 h	++	+	+
3	**PART_AA(10%)**	6 h	–	+	+
4	**PART_AA(30%)**	6 h	–	+	+
5	**PART_AA(60%)**	6 h	+	+	+
6	**PART_AA(80%)**	6 h	+	+	+
7	**PART_AA(90%)**	6 h	+	+	+
8	**MULT_AA**	80 min	+	+	+
9	**RAND_AA(true)**	6 h	–	+	+
10	**RAND_AA(fix)**	6 h	–	–	+
11	**STRUCT_AA(zero)**	6 h	–	–	−−
12	**STRUCT_AA(one)**	6 h	–	−−	−−
13	**STRUCT_AA(row)**	6 h	–	–	+
14	**STRUCT_AA(checker)**	6 h	–	–	+
15	**DYN_AA(rand)**	cont.	–	+	+
16	**DYN_AA(fix)**	cont.	–	+	+

D. Test Results

During the accelerated aging period the distributions of the PUF quality measures from Sect. II-A have been monitored: intra-distance (to evaluate reliability), inter-distance (uniqueness), and Hamming weight (unpredictability). The main focus is on intra-distance and the ultimate goal is finding *anti-aging*



(a) Intra-distance evolution (b) Inter-distance evolution (c) Hamming weigth evolution

(d) Intra-distance total change (e) Inter-distance total change (f) Hamming weight total change

Fig. 3: Impact on SRAM PUF quality measures during the accelerated aging test. The top graphs show the evolution of these measures during aging, while the bottom bar plots show their total relative change since the start of the experiment.

strategies which stabilize or even improve an SRAM PUF's reliability over time. However, the other two quality measures are also monitored to assure that any gain in reliability does not come at the expense of the PUF's uniqueness or unpredictability. A degradation of these parameters could cause a security risk since the PUF responses become less random.

An ideal anti-aging strategy prevents the mean intra-distance from increasing, but should also prevent the inter-distance and Hamming weight from moving away from 50%. For the tested SRAM PUFs, both the initial average inter-distance and Hamming weight are already very close to 50%, so they should preferably stay constant over time since any significant change (increase or decrease) makes them less ideal.

Using the output data of the experiment, a distribution of results from the 20 SRAMs has been compiled for each of the three PUF quality measures and for every week of aging elapsed in the experiment. The mean values (μ) of these distributions are calculated for each of the (anti-)aging scenarios. The evolution of these μ's for the different quality measures and aging scenarios for the duration of the experiment is presented in Figures 3a-3c [1], with colors indicating whether a quality measure is deteriorating over time (red) or not (green). The overall shift of the mean values for each of the quality measures during the total duration of the experiment is shown in Figures 3d-3f. A qualitative summary is to be found in Table I. The most important findings from these results and their consequences will be discussed in the next section.

[1]These graphs show a slowing down of most (anti-)aging effects on the SRAM PUF toward the end of the accelerated aging experiment which could be an indication that the accelaration factor AF as determined in Sect. III-C is not constant but slowly decreasing over time.

IV. Discussion

The experimental aging results presented in Table I and Figure 3 clearly show that for reliability, the **NO_AA** scenario exhibits the worst-case evolution displaying the largest increase in intra-distance over time. **FULL_AA** on the other hand shows the best-case evolution with an intra-distance which even decreases over time. These experimental observations are completely in line with the hypothesized physical effects of NBTI aging for SRAM cells as detailed in Sect. II-C. We consider these two cases as references for assessing the performance of the other tested (anti-)aging scenarios.

1) Ideal scenario: Besides improving reliability over time by up to 0.35%, **FULL_AA** also has no negative effect on inter-distance and Hamming weight since it keeps them stable[2] at their already ideal values very close to 50%. In this respect, **FULL_AA** is to be considered as *the ideal anti-aging strategy* for SRAM PUFs and it is highly recommended to apply it whenever possible since it significantly relaxes the error-correction requirements of the PUF-based system. However, depending on the application and implementation constraints, the **FULL_AA** scenario is not always possible, e.g. because:

- a perfect error-free reconstruction of the enrollment data is not available;
- it is required that all (security-sensitive) PUF response data is completely erased after use, this rules out using its inverse as anti-aging data;
- the SRAM is not solely dedicated to the SRAM PUF, but is used for other purposes afterward;

[2]We consider very small changes of less than 0.10% as being stable since they are not statistically significant.

- the SRAM PUF is enrolled multiple times and there is no single ideal enrollment data pattern.

These constraints have led us to investigate the (anti-)aging effects of other data scenarios, which we discuss next.

2) No/partial error-correction: When a perfect reconstruction of the enrollment data is not possible, the preferable scenario becomes **PART_AA**. Depending on the amount of achievable error-correction, the effect on reliability varies (the more error-correction, the better the effect on reliability). The results show that with a limited error-correction of 30% or lower, the reliability will slowly degrade over time. However, the deterioration is very small and significantly less than for **NO_AA**. With more error-correction the reliability will improve over time, as the results for 60% correction and higher show. Similar to **FULL_AA** the inter-distance and Hamming weight remain stable.

3) PUF zeroization: In case all security-sensitive PUF response information needs to be deleted from the SRAM, the best performing scenario is **RAND_AA(true)**. Although reliability degrades over time, it is still the most reliable scenario in which the data stored in SRAM is not based on the PUF response. The increase in intra-distance is also still considerably smaller than for **NO_AA**. The inter-distance and Hamming weight remain stable in this scenario. An important remark here is that the "obvious" option of really zeroizing the SRAM, i.e. overwriting it with all zeros (or ones), is a particularly bad choice since it not only degrades the reliability worse than all other options (besides **NO_AA**), it also has a significant negative impact on both inter-distance and Hamming weight. In fact, due to their negative effect on both intra- and inter-distance, none of the structured scenarios (**STRUCT_AA(...)** as well as **RAND_AA(fix)**) are recommendable, and one should even take care to avoid them in in-the-field situations.

4) Non-dedicated SRAM: When the SRAM is used for other purposes besides the SRAM PUF, the **DYN_AA** scenarios come into play. In these scenarios the reliability decreases over time, which is as expected because the data in the SRAM is not related to the PUF data. However, the reliability is still significantly better than for **NO_AA**, and when the dynamically written data to the SRAM is sufficiently random it will not deteriorate the inter-distance and Hamming weight.

5) Multiple-enrolled PUF: The **MULT_AA** scenario applies to SRAM PUFs which are enrolled more than once. The performance on all three quality measures is very good and comparable to that of **FULL_AA** and **PART_AA** with high error-correcting capabilities. **MULT_AA** exhibits an improvement in reliability and hence anti-ages the SRAM PUF.

V. Conclusion

We investigated the effects of silicon aging (NBTI) on SRAM PUF reliability. The physics behind NBTI predicts that SRAM PUFs have a natural tendency to become less reliable over time, but also reveals a potential solution which could even *anti-age* SRAM PUFs using the appropriate data-dependent actions. These cases, as well as a number of other relevant scenarios were tested in an accelerated aging experiment on a set of SRAM PUFs implemented on 65nm CMOS ICs. The test results confirm the predictions and identify an ideal anti-aging strategy which causes the tested PUFs to become 0.35% more reliable over an aging period of more than 6 years, without degrading the other PUF quality measures. This *full anti-aging* strategy even improves SRAM PUF reliability up to 2.45% compared to the natural reliability degradation over the same period when no actions are taken. Even when the full anti-aging scenario is not applicable, there are still strategies which are significantly better than doing nothing. A major practical advantage of the proposed anti-aging solutions is that they do not require any circuit changes or pre-deployment effort. They are hence usable for standard SRAM implementations in a regular development flow.

A noteworthy side observation of independent interest is that SRAM PUF zeroization (for security reasons) with a fixed pattern (e.g. all zeros) is particularly detrimental for each of the PUF's quality measures, and therefore highly discouraged. Overwriting the PUF response with on-the-fly randomly generated data is preferable instead.

Acknowledgment

We would like to thank Peter Simons and Dariusz Tesmer for their valuable help in performing the tests. This work has been supported in part by the European Commission through the FP7 programme under contract 284833 PUFFIN, and through the Catrene project RELY. The ASIC used in the described tests was developed in the FP7 project UNIQUE.

References

[1] NXP, "SmartMX2 unleashes secure multi-applications without compromise," http://www.nxp.com/documents/leaflet/75017276.pdf, Aug. 2012.

[2] Microsemi, "SmartFusion2 SoC FPGA Reliability and Security User Guide," http://www.microsemi.com/document-portal/doc_download/130926-smartfusion2-security-and-reliability-user-s-guide, Sep. 2013.

[3] Coherent Logix, "HyperX: Security Processing and Trusted Computing," http://www.coherentlogix.com/products/hyperx-processors/security/, Jan. 2014.

[4] S. Katzenbeisser, U. Kocabaş, V. Rožić, A.-R. Sadeghi, I. Verbauwhede, and C. Wachsmann, "PUFs: Myth, Fact or Busted? A Security Evaluation of Physically Unclonable Functions (PUFs) Cast in Silicon," in *Cryptographic Hardware and Embedded Systems (CHES) 2012*, ser. LNCS. Springer, 2012, vol. 7428, pp. 283–301.

[5] R. Maes, V. Rožić, I. Verbauwhede, P. Koeberl, E. van der Sluis, and V. van der Leest, "Experimental evaluation of Physically Unclonable Functions in 65 nm CMOS," in *European Solid-State Circuits Conference (ESSCIRC)*, Sep. 2012, pp. 486–489.

[6] J. Guajardo, S. S. Kumar, G.-J. Schrijen, and P. Tuyls, "FPGA intrinsic PUFs and their use for IP protection," in *Cryptographic Hardware and Embedded Systems (CHES)*, ser. LNCS, vol. 4727. Springer, 2007, pp. 63–80.

[7] G.-J. Schrijen and V. van der Leest, "Comparative analysis of SRAM memories used as PUF primitives," in *Design, Automation Test in Europe (DATE) 2012*, 2012, pp. 1319–1324.

[8] M. Bhargava, C. Cakir, and K. Mai, "Reliability enhancement of bistable PUFs in 65nm bulk CMOS," in *Hardware-Oriented Security and Trust (HOST)*, 2012, pp. 25–30.

[9] M. Bhargava and K. Mai, "A High Reliability PUF Using Hot Carrier Injection Based Response Reinforcement," in *Cryptographic Hardware and Embedded Systems (CHES)*, ser. LNCS. Springer, 2013, vol. 8086, pp. 90–106.

[10] JEDEC, "Failure Mechanisms and Models for Semiconductor Devices - JEP122G," http://www.jedec.org/standards-documents/docs/jep-122e, Oct. 2011.

Appendix B

Paper: "Empirical Analysis of Intrinsic Physically Unclonable Functions Found in Commodity Hardware"

Author: André Schaller (TU Darmstadt), Anthony Van Herrewege (KU Leuven), Vincent van der Leest (Intrinsic-ID), Tolga Arul (TU Darmstadt), Stefan Katzenbeisser (TU Darmstadt), and Ingrid Verbauwhede (KU Leuven)
Journal: Journal of Cryptographic Engineering

Status: Under Review

Noname manuscript No.
(will be inserted by the editor)

Empirical Analysis of Intrinsic Physically Unclonable Functions Found in Commodity Hardware

André Schaller · Anthony Van Herrewege · Vincent van der Leest · Tolga Arul · Stefan Katzenbeisser · Ingrid Verbauwhede

Received: date / Accepted: date

Abstract Physical(ly) Unclonable Functions (PUFs) have become a promising security building block for many cryptographic applications over the last decade. The stable portion of a PUF measurement can be utilized by cryptographic protocols in order to store secret keys and to attest the configuration of a remote device. In contrast, the noisy portion of a PUF measurement allows for generating random numbers. Provision for the implementation of PUF features has to be made at circuit level. Therefore, components that feature PUF behavior have to be either included in the hardware design, as in the case of ASICs, or can be implemented after manufacturing using FPGAs. In contrast, intrinsic PUFs are built out of components that already exist in hardware modules. For example, most commercial commodity hardware contain Static Random-Access Memory (SRAM) modules, which can potentially be used to implement intrinsic PUF instances. Consequently, the characteristics of embedded SRAM modules have to be thoroughly assessed before such intrinsic PUFs can be employed in security-related applications. In this paper we evaluate the on-board SRAM modules contained in a broad range of commercial off-the-shelf (COTS) devices regarding their quality as PUFs and sources of entropy. In particular, we perform measurements on popular microprocessors featuring ARM-, Atmel- and PIC-based architectures in order to analyze the on-chip SRAM with regard to the quality metrics of robustness, randomness and entropy. Our evaluation shows that most of the tested modules are suitable for PUF usage and as a source of entropy.

Keywords Physically Unclonable Functions · Identification · Authentification · Random Number Generation

Anthony Van Herrewege · Ingrid Verbauwhede
KU Leuven Dept. Electrical Engineering-ESAT/SCD-COSIC and iMinds, Kasteelpark Arenberg 10, 3001 Leuven, Belgium
E-mail: firstname.lastname@esat.kuleuven.be

André Schaller · Stefan Katzenbeisser
Security Engineering Group, Technische Universität Darmstadt and CASED, Germany
E-mail: lastname@seceng.informatik.tu-darmstadt.de

Tolga Arul
CASED, Mornewegstraße 32, 64293 Darmstadt, Germany
E-mail: tolga.arul@cased.de

Vincent van der Leest
Intrinsic-ID, Eindhoven, The Netherlands
E-mail: vincent.van.der.leest@intrinsic-id.com
http://www.intrinsic-id.com

1 Introduction

Physical(ly) Unclonable Functions (PUFs) have become a promising security building block for many cryptographic applications over the last decade. For example, PUFs can be used to identify silicon chips on the basis of their physical structure [10], which is assumed to be unclonable by both the manufacturer and any other party. Their characteristics make PUFs appealing for use in conjunction with cryptographic applications. Besides for identification and authentication purposes, PUFs were proposed to be used as a secure key-storage solution [9], since a key generated out of a PUF only persists in memory while the device is powered on. PUFs can also be used as building blocks in cryptographic primitives or to bind hardware and software.

PUF responses consist of a stable part, which can be used as unique fingerprints, as well as an unstable part, which induces noise to the PUF measurements. This unstable part of a PUF measurement can not be used for identification or key generation purposes. However, as this noise is based on fluctuating physical ambient

conditions it can be used to generate true random numbers, which are regarded as hard to produce in software.

In literature, most studied PUF instances are custom hardware components that have to be created as part of an ASIC or that have to be implemented in an FPGA. However, some SRAM modules present in commercial off-the-shelf (COTS) devices exhibit good PUF behavior, even though such modules were generally not designed for the purpose of using them as PUFs [17]. Given that SRAM modules can be found in almost every COTS device, this PUF behavior can be found in many devices on the market. Using COTS devices for PUF use is a worthwhile idea as it allows for lightweight cryptographic solutions, especially for devices with constrained resources. Such devices often cannot be equipped with additional cryptographic hardware modules due to hardware or economic limitations. Utilizing intrinsic PUFs typically only requires changes to software, making them easy to be applied to already deployed devices.

In this work, we empirically assess the PUF characteristics of SRAM present in a broad range of popular COTS devices which were not designed for PUF use during production. We analyze intrinsic PUFs for robustness, uniqueness and noise entropy.

To the best of our knowledge, this work is the first one which qualifies and discusses such a large number of COTS devices with regards to their usability in PUF-related applications.

2 PUFs and PUF Applications

2.1 Physically Unclonable Functions

PUFs are complex physical structures that generate a value in response to a stimulus. The class of silicon PUFs (e.g., static random-access memory PUFs) is based on manufacturing variabilities and can be found in certain integrated circuits. Silicon PUFs can be grouped into delay- and memory-based PUFs. For an exhaustive overview of PUFs and details on their taxonomy we refer to [13].

It has been shown that the content of SRAM after power up shows PUF-like behavior [10]. Further research in this area supports the applicability of SRAM as a PUF [12]. Using SRAM modules as PUFs exploits manufacturing variations, which manifest themselves as a bias of memory cells inside the SRAM modules. SRAM cells are designed as cross-coupled inverters that exhibit a bi-stable behavior. As a result of the manufacturing variations, one inverter is more dominant than the other in most of the individual cells. When powered on, these cells eventually perform a transition from the meta-stable state to a stable state, leading to bit values of either one or zero. It was shown by Guajardo et al. [9] that most memory cells are biased either towards zero or towards one after start-up. These cells allow for the generation of a fingerprint, which is unique for individual devices.

2.2 PUFs as instances of hardware fingerprints

Besides uniquely identifying hardware, PUFs can be used in combination with a Fuzzy Extractor, which eliminates noise from PUF responses, to realize secure key storage. In this case, SRAM start-up values are used to reconstruct a predefined cryptographic key with support of so-called helper data. This scenario is comprised of two phases: *i*) the enrollment; and *ii*) the reconstruction phase. During the enrollment phase helper data is produced from an enrollment PUF measurement and a predefined secret. The helper data is necessary to reliably reconstruct the same secret from several noisy measurements of a single device during the reconstruction phase. Secure key storage based on SRAM PUFs allows for the implementation of several security-related applications. A popular use case with respect to intrinsic SRAM PUFs is to use the reconstructed cryptographic key to decrypt the firmware or parts of the bootloader in order to bind a given software instance to a specific device [17].

To reliably identify a device respectively reconstruct a key, the noise of SRAM measurements should not exceed a certain threshold. Furthermore, to guarantee the uniqueness of the SRAM start-up values and thus the derived key, the SRAM measurements also need to be sufficiently different across different device instances. A detailed explanation as well as an evaluation of these characteristics is presented in Section 4.1.

2.3 Random number generation

To generate random numbers for cryptographic applications, two basic methods can be used. The first method requires a physical source which is truly random and whose output can be directly used to construct random bits. Such non-deterministic sources derive their randomness from underlying physical properties that exhibit unpredictable behavior. Examples of such sources of randomness in chips are free running oscillators connected to a shift register [16] and noise on the lowest bits of AD converters [14], but many more exist. These Random Number Generators (RNGs) are referred to as True Random Number Generators (TRNGs). There

are two important downsides to most of these physical RNG constructions. Firstly, they require specific hardware to extract the randomness from the physical entities on the device. Secondly, the throughput of such RNGs is often too low for cryptographic applications, where large streams of random bits are required.

The second approach to generate randomness is by using pseudo-RNGs (PRNGs), which are constructed using deterministic algorithms. The output of such a generator only seems to be random to observers who do not have any knowledge of the initial state of the generator. If an observer knows, which data has been used as a seed for the PRNG, he is able to calculate all output values of the generator due to its deterministic nature. Hence, the seed value should be chosen randomly and kept secret. One advantage of PRNGs is that they can be implemented completely in software and therefore do not require any additional hardware. Moreover, PRNGs are able to produce streams of (pseudo-)random bits at a sufficient output rate for cryptographic applications.

The noise present in PUF responses can be used to derive truly random seeds for a PRNG. The basic idea of using noisy SRAM start-up states as a source for PRNG seeds was investigated in [10], as well as in [11]. The former paper proposes to use a universal hash to generate a single random number at start-up. The authors verify their approach by using an external SRAM module. However, they do not investigate whether their approach is feasible on embedded SRAM, which is integral to most COTS microcontrollers. In the latter paper, the feasibility of creating a strong PRNG with the use of random data from an ASIC containing SRAM-based Physically Unclonable Functions (PUFs) is investigated. Van Herrewege et al. [18] propose to exploit the intrinsic noise entropy of SRAM measurements right after power up and post-process them in order to extract a seed that is subsequently used as an input for a PRNG. The construction collects entropy in the SRAM start-up values to produce a secure seed for a hash function, which then provides a continuous stream of pseudo-random numbers.

In order to use SRAM start-up values as a reliable basis for PRNGS, they must provide enough entropy. The noise entropy is evaluated in Section 4.2.

2.4 Metrics for Identification and Entropy Extraction

Metrics for Identification. For identification purposes a PUF instance should show properties that mitigate the prediction of start-up values (partially reflected by the Hamming weight), enable a robust repeated identification of single devices (within-class Hamming distance) and lastly generate a unique pattern among a pool of similar devices (between-class Hamming distance).

The *Hamming weight* (HW) of individual measurements from the same device indicates whether the start-up values are biased to zero or one. This metric gives a first impression about the randomness present in the start-up values. The ideal Hamming weight follows a Gaussian distribution with a mean of 50%, representing no bias of the start-up values towards zero or one.

The *Within-class Hamming distance* (WCHD) gives an indication whether the PUF results for a single device are stable when queried repeatedly. Robustness of the start-up values is required to reliably identify a given device and subsequently reconstruct the corresponding cryptographic key. WCHD is a normalized count of bits which differ between subsequent PUF measurements and thus is a rational number between zero and one. An optimal value for the within-class Hamming distance is close to zero. However, all start-up values show a certain amount of noise, which originates from SRAM cells that flip their start-up value over multiple trials.

The *between-class Hamming distance* (BCHD) expresses whether the start-up values of different devices are independent. This metric states whether start-up values can be used for identification without enabling adversaries to predict a measurement for a second device on the basis of measurements of a first device. In the optimal case, the between-class Hamming distance follows a Gaussian distribution with mean 50%. If this is true, then the start-up values are most likely independent. Devices with an optimal value of 50% exhibit a maximum distinguishability regarding their PUF responses.

Metrics of Entropy. For the purpose of extracting random seeds from SRAM start-up values, it is important to investigate their entropy content. In this paper, we use the min-entropy to quantify the entropy of the SRAM patterns [11]. This method is based on the NIST specification [7] that defines min-entropy as the worst-case (i.e., the greatest lower bound) metric of uncertainty for a random variable.

For a binary source, we define the min-entropy as

$$H_{min} = -\log_2(\max(p_0, p_1)), \tag{1}$$

where p_0 and p_1 are the probabilities of an occurence of 0 or 1. Assuming that all bits from the start-up pattern are independent, each bit i is an individual binary source. For each of these sources we estimate the probabilities p_0^i and p_1^i of powering up in state 0 or 1, by repeatedly measuring the power-up values of the SRAM. In case m subsequent measurements are performed, p_0^i

denotes the number of occurrences of a zero, divided by m and $p_1^i = 1 - p_0^i$. For a start-up pattern of length n we have

$$H_{min} = \sum_{i=1}^{n} -\log_2(\max(p_0^i, p_1^i))). \quad (2)$$

Hence, under the assumption that all bits are independent, we can sum the entropy of each individual bit to derive the min-entropy of the entire SRAM.

Fractional Metrics. All metrics in this work are presented as fractional values in order to improve comparability between various devices. E.g. a fractional min-entropy of 6.0% of a 8 KiB memory is approximately equal to 491.5 bytes of min-entropy.

3 Experimental Setup

3.1 Measured devices

In this section, we present the devices and microcontrollers for which we analyze the start-up patterns of their embedded SRAM. As a basis for our investigation, we selected popular commercial devices ranging from light-weight microcontrollers to more complex system-on-a-chip platforms. Our test bed comprises chips from Atmel, Microchip, STMicroelectronics, and Texas Instruments. We selected widely-used microcontrollers, representing industry standards in their respective class of 8-, 16- and 32-bit processors. For all device types, the investigated SRAM instances are on-chip, i.e. we do not test external SRAM modules. Table 1 gives an overview of the tested device types as well as details on important hardware aspects.

PIC16F1825: The PIC16F1825 [4] is an 8-bit low-power microprocessor developed by Microchip. The microprocessor holds 1 KiB of static RAM, 8 KiB of flash memory and 256 bytes of EEPROM. The microcontroller was specifically designed to be used in medical devices, in automotive scenarios or as part of home appliances.

ATmega328P: The ATmega328P [1] is a low-power 8-bit processor produced by Atmel as part of the megaAVR series. It has 2 KiB of static RAM, 32 KiB of flash and 1 KiB EEPROM on board. The ATmega328P was designed for applications in highly competitive markets where production costs have to be very low.

MSP430F5308: The MSP430F5308 [2] is a low-energy 16-bit processor produced by Texas Instruments. The device holds 6 KiB of static RAM and 16 KiB of flash memory. It is optimized for low current drain and thus is used in battery-backed devices for energy constrained applications which require a long service life.

STM32x: The STM32F100Rx series [5] is a 32-bit microprocessor equipped with an ARM Cortex-M3 CPU. We investigated two models of this device: *i*) the STM32VL-Discovery evaluation board, which contains an STM32F100RB microprocessor; and *ii*) the STM32F100R8 microprocessor. Both versions hold 8 KiB static RAM and respectively 64 KiB flash memory (STM32F100R8) and 128 KiB flash memory (STM32F100RB). The evaluation board additionally provides several peripherals and a debugging interface. The STM32F100 series was developed with focus on industrial-control applications. The embedded Cortex-M3 is a low-power microcontroller used in power sensitive and high performance applications.

LM4F120H5QR: The LM4F120H5QR is a 32-bit ARM Cortex-M4F microcontroller with 32 KiB SRAM, 256 KiB flash memory and several programmable interfaces. For our measurements we use the Texas Instruments EK-LM4F120XL development board [6], which is based on this microprocessor. It was developed for a variety of industrial applications ranging from electronic point-of-sale machines, and network appliances to factory automation. The Cortex-M4F is conceptually equivalent to the Cortex-M3 but additionally supports instructions for digital signal processing and features a floating point unit.

PandaBoard: The PandaBoard is based on an OMAP4 system-on-a-chip (SoC) platform [3]. It integrates two ARM Cortex-A9 processors as well as two Cortex-M3 co-processors for signal processing. The PandaBoard is based on a Texas Instruments OMAP4430 SoC running at 1.0 GHz, the PandaBoard ES is based on an OMAP4460 running at 1.2 GHz. The platforms also come equipped with 1 GiB of external DDR2 SDRAM. The dual-core Cortex-A9 is a high-performance application processor for low power or cost sensitive devices. With the PandaBoard's variety of interfaces – including USB, HDMI, DVI-D, ethernet, and WiFi – it is mainly used as a development platform for modern smartphones and tablets.

3.2 Measurement setup

In this subsection, we present the soft- and hardware setup used to evaluate COTS microcontrollers for their PUF characteristics, based on the work presented in [18]. First, we present the functionality and requirements of firmware that is loaded onto each microcontroller in order to extract the start-up values. Thereafter, we describe the hardware setup used to automatically perform measurements on several devices.

3.2.1 Firmware design

In order to extract the raw SRAM start-up values we developed a custom firmware, which performs the following steps on power-up:

1. Initialize the serial port (UART).
2. Transmit every SRAM byte over the UART.
3. Idle the device.

While designing the firmware we had to take into account the important constraint that, if possible, no SRAM should be used while executing the commands to complete the steps described above. The reason for this is that any writes to SRAM remove parts of the start-up value. This requirement is easily met on most microcontrollers which possess several working registers, such that we could, e.g., store a pointer to an SRAM byte. However, some microcontrollers, such as those of the Microchip PIC16 family, only have a single working register and therefore we need to store variables in unused configuration registers to avoid writes to SRAM.

3.2.2 Hardware setup

In order to obtain an initial set of SRAM start-up values, we performed the measurement of SRAM start-up patterns a few times by hand. For this purpose we connected the power lines and the serial port of the device under test to an external serial TTL-to-USB converter. The converter, in turn, was attached to a self-powered USB hub. After receiving all start-up values we removed the lines for the power supply for at least 10 seconds in order to ensure a fresh SRAM pattern for the next start. We observed that this cycle delivered reasonable values for all device types, except for the *PIC16F1825*. The SRAM patterns in these devices persisted for over 10 minutes after the power lines had been disconnected.

In order to extract start-up patterns faster and more efficiently, we used two custom measurement boards. The boards can simultaneously connect multiple microcontrollers. They turn on every connected device, receive the respective SRAM start-up patterns and forward them to the desktop PC, turn of the microcontroller and idle to give the respective microcontroller's SRAM a chance to discharge. This procedure is repeated automatically. For a detailed explanation of the the custom controller boards, we refer to [18].

4 Evaluation

In this section we present the results of analysis of the SRAM start-up values of different device types with respect to the metrics introduced in Section 2.4. We first deal with metrics concerning robustness and uniqueness. Subsequently, we go into detail with respect to noise entropy. All devices have been tested at room temperatures of approximately 20°C. For specific devices, we also conducted measurements in a climate chamber at different extreme temperatures: −30 °C, 20 °C and 85–90 °C. Not all devices were exposed to temperatures tests because some device types hold components, which would get corrupted at extreme temperatures. For example, the the PandaBoard includes an LCD display and other hardware elements which would be destroyed if exposed to extreme temperatures. An overview of the measurement conditions, including the number of devices for each device type as well as the number of measurements performed at respective ambient temperatures can be found in Table 1. For all devices, the first measurement at room temperature has been used for enrollment. All other measurements are compared to the enrollment measurement. The odd numbers of measurements are due to the fact that during the evaluation some measurement errors occurred, which have been removed from the data set before analysis.

Metrics are plotted as whisker plots, which are constructed as follows: the central line of each box shows the median value, whereas the bottom and top of the box show respectively the 1st and 3rd quartile. The two whiskers extending from the box show respectively the 2nd and 98th percentile. Finally, we plot the minimum and maximum value with a × symbol.

4.1 Evaluation of uniqueness and robustness

Hamming Weight

In the following section we will present the results of the Hamming weight (HW) characteristics of the measured devices. Exemplary, Figure 1 shows average Hamming weight results for the individual *STM32F100R8*

Table 1 Details of devices analyzed for PUF characteristics of on-chip SRAM start-up values.

Device Type	Processor Type	Manufacturer	SRAM size [KiB]	Device count	Measurements per device		
					20 °C	−30 °C	90 °C
ATmega328P	AVR	Atmel	2	16	9 695	2 916	2 989
PIC16F1825	PIC16	Microchip	1	16	3 700	3 662	3 671
MSP430F5308	MSP430	Texas Instruments	6	15	3 174	3 339	3 359
STM32F100R8	ARM Cortex-M3	STMicroelectronics	8	11	3 419	7 745	9 003
STM32F100RB	ARM Cortex-M3	STMicroelectronics	8	14	1 069	—	—
LM4F120H5QR	ARM Cortex-M4F	Texas Instruments	30	15	1 000	—	—
PandaBoard (ES)	ARM Cortex-A9/M3	Texas Instruments	12	6	1 000	—	—

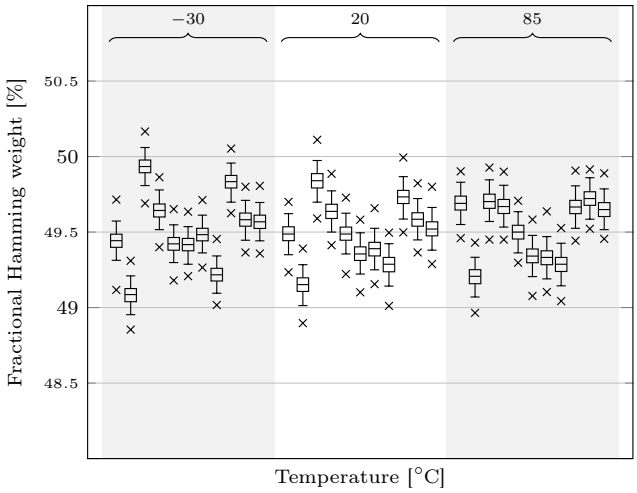


Fig. 1 Comparison of average Hamming weights for the individual STM32F100R8 devices.

microcontrollers that exhibit almost perfect HW characteristics. For each individual microcontroller corresponding whisker boxes display the HW measurements for three ambient temperatures. All the values gather nicely close to the optimal value of 50% with almost no differences among the different temperature measurements, showing almost perfect HW characteristics for this device type. Moreover, Figure 2 compares the average HW per device type. Several device types exhibit start-up values with good or reasonable properties. The *ATmega328P* and the *MSP430F5308* show undesired results at any temperature. Other devices show highly varying HW values, e.g. the *LM4F120H5QR* as well as the *PIC16F1825* or the *PandaBoard*.

The results for the PUF responses of the *STM32F100RB*, the *STM32F100R8* and the *PandaBoard* indicate properties that are close to the desired distribution, indicating that the start-up values contain almost the same proportion of zeros and ones, although the *PandaBoard* shows a notable high standard derivation. The HW for the *LM4F120H5QR* show reasonable characteristics with a worst-case Hamming weight of 43.29%[1]. This reveals that there is a certain correlation among the start-up values of different devices. Thus, the required portion of SRAM to derive a unique fingerprint increases. Another notable characteristic for the *LM4F120H5QR* is the variance among the Hamming weight values for the different devices. The maximum difference between Hamming weights of different devices is around 11%. We could not find any reason for this behavior. Also the spread of the HW for the *PIC16F1825* is rather large compared to the rest of the devices. Again, the actual reason for this behavior could not be identified.

The Hamming weight of the start-up values generated by the *ATmega328P* as well as by the *MSP430F5308* are significantly higher than 50% at any of the tested temperatures. This indicates a higher portion of ones than zeros in the PUF responses.

The *PIC16F1825's* fractional Hamming weight is close to 50%, which at first glance represents desired characteristics. However, a visual examination of the PUF responses reveal a repeating pattern, which will be discussed in detail in Section 4.2.

In summary, the numbers for all devices suggest that their SRAM start-up values are suitable inputs for commonly known Fuzzy Extractors, even for those devices with a HW that can not be regarded as optimal. In the latter cases those characteristics will lead to an increased PUF response size to reliably extract an identifier by the Fuzzy Extractor algorithm. However, these requirements can be fulfilled at the cost of using more SRAM. Detailed numbers of the average Hamming weights can be seen in Table 2. The values shown are the minimum and maximum values from the

[1] The measurements of the *LM4F120H5QR* exhibit a series of null bytes of approximate size of 2 KiB at the beginning of the SRAM region. These null bytes influence the Hamming weight of the devices towards zero. The exact reason for this behavior is unclear. We assume that these null bytes were introduced due to the usage of ROM code to initialize the UART interface. Some ROM functions may allocate space, which is never used and thus remain as a series of zeros. Thus, we excluded the first 2 KiB from the analysis.

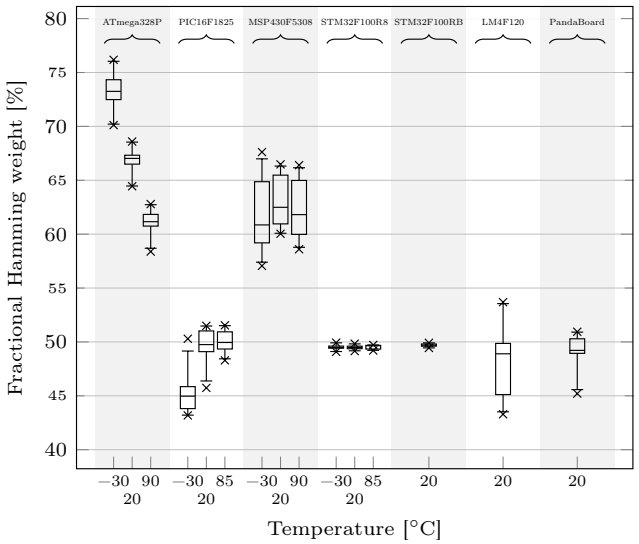


Fig. 2 Comparison of average Hamming weights among device types.

Table 2 Fractional Hamming weight (HW) of investigated devices under different thermal conditions.

Device Type	HW [%] (min; max)		
	20 °C	−30 °C	85–90 °C
ATmega328P	64.44; 68.60	70.12; 76.18	58.38; 62.79
PIC16F1825	45.74; 51.50	43.18; 50.29	48.29; 51.54
MSP430F5308	60.04; 66.47	57.06; 67.62	58.60; 66.40
STM32F100R8	49.15; 49.84	49.08; 49.93	49.20; 49.72
STM32F100RB	49.44; 49.92	–	–
LM4F120H5QR	43.29; 53.69	–	–
PandaBoard (ES)	45.20; 50.94	–	–

averaged HW results of each individual microcontroller instance.

Within-class Hamming distance

The results of the averaged within-class Hamming distances (WCHD) of the analyzed device types can be seen in Figure 4. Since the WCHD refers to the noise exhibited in the SRAM start-up values of a given device across multiple activation operations, it is desired to have a WCHD as low as possible. The existing noise needs to be eliminated using an error correcting scheme. For this purpose, commonly a Fuzzy Extractor is used. A reference Fuzzy Extractor design presented in [8] can correct up to 15% noise. Thus, the besides the requirement of the WCHD to be as low as possible, it should not exceed this threshold. For a more detailed reference we present the averaged WCHD values for each individual *STM32F100R8* microcontroller in Figure 3. Again, measurements are represented using whisker boxes for the individual devices at three different temperatures. For a fixed temperature the WCHD values show almost no spread regarding their standard deviation. Although, the WCHD values vary among different temperature measurements, all WCHD values are below the noise threshold, showing that PUF measurements of this device type are robust also under extreme conditions.

The maximum within-class Hamming distance at room temperature for the *PIC16F1825* and the *ATmega828P* is below 3%, indicating nearly perfect characteristics such that noise can be easily corrected using commonly known Fuzzy Extractors. The WCHD results for the *MSP430F5308*, the *LM4F120H5QR* and the *PandaBoard* show good properties at room temperature, whilst the *STM32F100R8* as well as the *STM32F100RB* exhibit reasonable characteristics.

However, looking at the temperature measurements of the *ATmega328P*, the *PIC16F1825* and the*MSP430F5308* our observation is that at low temperatures the SRAM start-up patterns exhibit a significant increase of the WCHD values for all tested devices.

To summarize, the gathered WCHD values for all the tested devices are well below the critical threshold of 15% such that the exhibited noise can be correct using a common Fuzzy Extractors. However, looking at the relative differences between room temperature and extreme temperature of some device, it is obvious that microcontrollers need to be tested thoroughly with regard to ambient conditions if they are supposed to be used in security-related applications. Detailed within-class Hamming distance results can be found in Table 3. We list only the maximum recorded value here, since that is what matters when selecting appropriate parameters for error correction algorithms (e.g. a Fuzzy

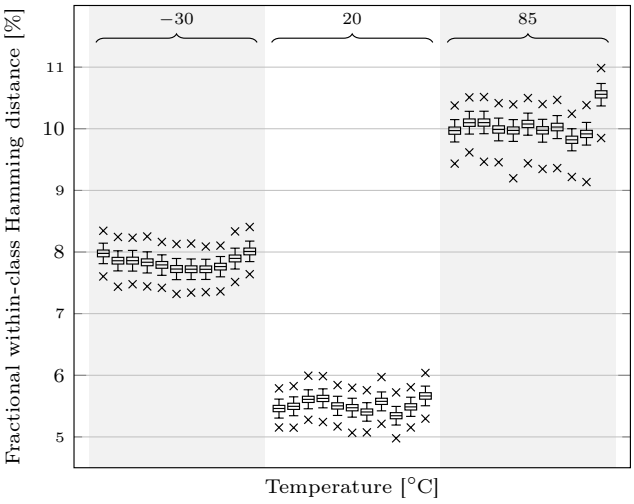


Fig. 3 Comparison of average Within-class Hamming distances for the individual STM32F100R8 devices.

Extractor). Again, the listed values are the maximum results from the averaged WCHD values of each measured microcontroller instance.

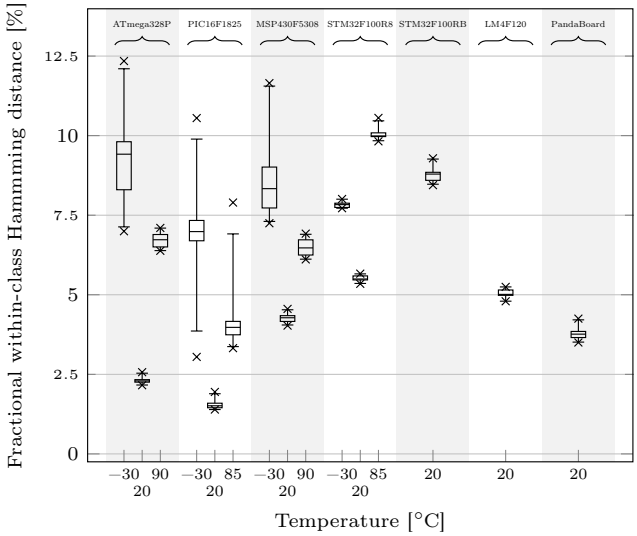


Fig. 4 Within-class Hamming distance for all measured device types.

Table 3 Maximum within-class Hamming distances (WCHD) of investigated devices under different thermal conditions

Device Type	Max. WCHD [%]		
	20 °C	−30 °C	85–90 °C
ATmega328P	2.57	12.34	7.10
PIC16F1825	1.95	10.55	7.90
MSP430F5308	4.56	11.65	6.92
STM32F100R8	5.66	8.01	10.56
STM32F100RB	9.29	—	—
LM4F120H5QR	5.25	—	—
PandaBoard (ES)	4.25	—	—

Between-class Hamming distance

Figure 5 compares the average between-class Hamming distances (BCHD) to each other. Whilst some device types exhibit near-perfect between-class Hamming distances, e.g. the *PandaBoard*, this is not the case for all measured devices. The *PIC16F1825* is especially bad in this regard, having a maximum measured BCHD of only 24.54%, making it unsuitable for unique identification. The BCHD of the *PIC16F1825* is much lower than required for a PUF implementation. The PUF measurements fit a Gaussian distribution with a mean value of 21.29%. This low mean value indicates that there is a very high correlation between the PUF responses from different devices. This makes them unsuitable to be used as part of cryptographic algorithms. The origin of this low BCHD is a repetitive pattern which appears in every measured PIC16F1825 device (see Section 4.2).

The PUF responses of the other device types exhibit a between-class Hamming distance which is close to the optimal 50%. They can be uniquely identified as the HD between different devices is much higher than the noise measured for each individual device (i.e. the WCHD).

Especially the BCHD of the *LM4F120H5QR* and the *PandaBoard* are remarkable, since they are almost exactly 50%. These numbers guarantee to provide a unique fingerprint for individual devices among a pool of devices of the same type.

An overview of average BCHD is given in Table 4. Note that the measurements at extreme temperatures are omitted as the BCHD values are computed on enrollment data. In the scenario of extracting a device fingerprint the keys are derived at room temperature. During reconstruction, the keys are merely recreated. Thus, the uniqueness of the key – and subsequently the SRAM start-up values – must be guaranteed at the point of enrollment. Given that enrollment data was measured at room temperature it is not useful to compare it to measurements at other temperatures.

4.2 Evaluation of entropy

Figure 6 shows the average min-entropy results for the tested devices. From the measurements it is clear that

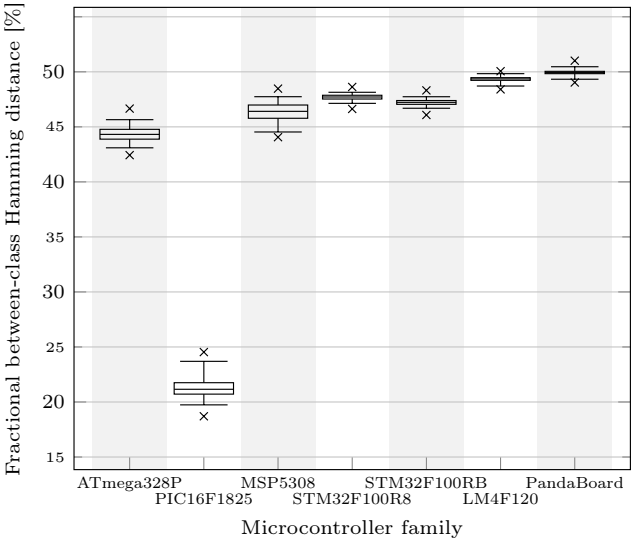


Fig. 5 Between-class Hamming distances for all measured device types.

the investigated device types behave differently. Whilst most of the devices show good results when it comes to deriving a truly random seed from the noise on SRAM start-up patterns, the *PIC16F1825* microcontrollers are not appropriate for this purpose. Min-entropy values of the noise are displayed for each measured *STM32-F100R8* device in Figure 7.

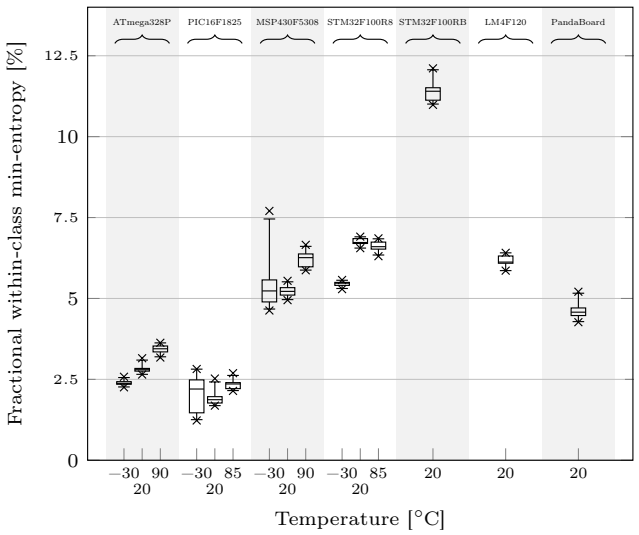


Fig. 6 Comparison of average fractional min-entropy of the noise exhibited among different device types.

In general, the behavior of the *PIC16F1825* is peculiar for several reasons. The *PIC16F1825* enrollment visualization shows an obvious repetitive pattern (see Fig. 8b). To be more precise: the bits of every alternating byte have a preference to start-up either as a 0 or a 1. This pattern is present in every *PIC16F1825* device we measured. Furthermore, as seen in Fig. 6 the standard deviation, for the min-entropy values of the *PIC16F1825* as well as of the *MSP430F5308* is much higher at −30 °C compared to the standard deviation at the other two temperatures. Whereas for other ICs, the min-entropy generally goes down at freezing temperatures, for the *PIC16F1825* it increases for some of the measured devices. The same behavior can be observed for the *MSP430F5308* to an even larger extent. Whilst the standard deviation for the measurements at room temperature and high ambient temperatures are as expected, it dramatically increases when lowering the ambient temperature. For these microprocessors, some start-ups at lower temperature generate patterns similar to those at room temperature, while others are much more different. Therefore, these particular devices have a higher min-entropy, since their start-up pattern is harder to guess. It thus seems that −30 °C is a temperature at which expected behavior of the *PIC16F1825* as well as of the *MSP430F5308* is not assured, since some measured devices behave as expected, whereas others clearly do not.

It is noteworthy that the min-entropy results for almost all tested device types are relatively stable under different thermal conditions. All the device types except

Table 4 Average between-class Hamming distances (BCHD) of investigated devices at room temperature.

Device type	BCHD [%]
ATmega328P	44.31
PIC16F1825	21.29
MSP430F5308	46.35
STM32F100R8	47.69
STM32F100RB	47.22
LM4F120H5QR	49.33
PandaBoard (ES)	49.94

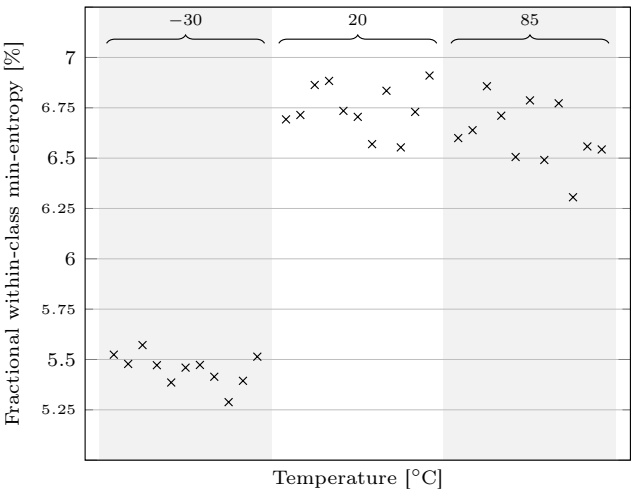
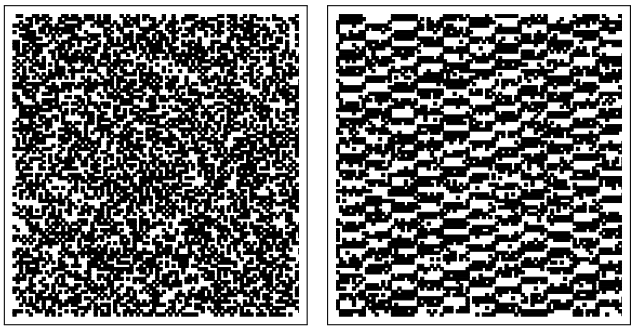


Fig. 7 Comparison of average fractional min-entropy of the noise extracted from the individual STM32F100R8 microcontrollers.



(a) (b)

Fig. 8 Example 1 KiB start-up patterns at 25 °C. Left: STM32F100R8, right: PIC16F1825. White represents a bit with value 0, black one with value 1.

for the *PIC16F1825* show fairly good entropy results, sufficient to extracting enough entropy for a strong seed for a PRNG. For example, the *STM32F100R8* devices exhibit a minimum amount of 5.29% min-entropy in every thermal test condition. Given the fact that the measured SRAMs have a size of 8 KiB, it is evident that one can extract a lot of entropy (e.g. using a cryptographic hash) to generate a truly random seed for a PRNG. Whilst most of the devices show good results when it comes to deriving entropy from the noise on SRAM start-up patterns, the *PIC16F1825* microcontrollers are unfit for the purpose of extracting a truly random seed from this noise. For the implementation of a PRNG seeded with a random seed derived from SRAM start-up noise, the *PIC16F1825* should not be used, since barely 128 bits of entropy can be generated from the start-up pattern of this microcontroller, whereas one would want a certain safety margin above this.

An overview of the minimum measured min-entropy per device type is shown in Table 5. We list the minimum measured min-entropy, since this is the value to take into account when determining how much SRAM is required to generate a strong random seed.

Table 5 Minimum within-device min-entropy of investigated devices under different temperatures.

Device Type	Minimum min-entropy [%]		
	20 °C	−30 °C	85–90 °C
PIC16F1825	1.69	1.23	2.14
ATmega328P	2.65	2.25	3.17
MSP430F5308	4.95	4.63	5.87
STM32F100R8	6.55	5.29	6.31
STM32F100RB	10.98	—	—
LM4F120H5QR	5.86	—	—
PandaBoard (ES)	4.27	—	—

4.3 Consequences relating to Applications

In summary, it should be asserted that for assessing the applicability of SRAM start-up patterns for security-related applications, in terms of their PUF as well as entropy characteristics, it is required to consider all the presented metrics. For example, regarding the *PIC16F1825*, solely taking the Hamming weights or the within-class Hamming distances into account would not reveal the visible pattern, which is reflected in the microcontrollers' between-class Hamming distances. The bad BCHD values disqualify the *PIC16F1825* devices for identification or entropy extraction purposes. Furthermore, it is necessary to test the metrics under different ambient temperatures as the device behavior might be significantly different to what was measured at room temperature. This is supported by the large relative differences of the within-class Hamming distances among different temperatures of the *PIC16F1825*, *ATmega328P* as well as the *MSP430F5308* devices. Except for the *PIC16F1825*, all devices can be used as a PUF instance and to extract entropy. However, considering the bias in the start-up pattern of the *ATmega328P* and the *MSP430F5308* specific processing steps have to be performed for identification purposes besides the common Fuzzy Extractor. The rather high bias in the SRAM start-up values of both devices decreases the entropy and thus requires more SRAM data for key reconstruction. To make sure that a robust reconstruction on such memory-constrained devices is possible, one could apply a de-biasing algorithm like the Von Neumann extractor [15] on the PUF measurement prior to enrollment. Such an randomness extractor in combination with a statistical analysis of the bias and the available SRAM size and additional Helper Data allows for an implementation of a PUF-based key reconstruction using biased SRAM values on memory-constrained devices.

5 Conclusion

In this work we evaluated a broad range of commercial off-the-shelf (COTS) devices with regard to their SRAM start-up values being either used as an device identifier or source of randomness. We introduced measures of two major groups of characteristics, namely *i*) uniqueness and robustness; and *ii*) noise entropy of the start-up values. Furthermore, we evaluated the devices with respect to these characteristics. We show that many SRAM instances that can be found on COTS devices can be successfully used to extract an identifier for the device. This enables security-related applications and protocols, as for example secure key storage or the implementation of security protocols based on PUFs. In contrast to traditional approaches such solutions are lightweight as they do not require additional hardware to store cryptographic keys (e.g. TPM chips). Furthermore we show that some SRAM modules contain enough randomness to make them a worthwhile source of entropy. Especially for low-end devices this is a desirable application as on such devices it is rather challenging to generate random numbers with high entropy.

We show that many SRAM instances that can be found on COTS devices can be successfully used to extract an identifier for the device. This enables security-related applications and protocols, as for example se-

cure key storage or the implementation of security protocols based on PUFs. In contrast to traditional approaches such solutions are lightweight as they do not require additional hardware to store cryptographic keys (e.g. TPM chips). Furthermore, our measurements show that some SRAM modules contain enough randomness to make them a worthwhile source of entropy. Especially for low-end devices this is a desirable application as on such devices it is rather challenging to generate random numbers with high entropy.

However, the evaluation of the PIC16F1825 in particular revealed repeating patterns in the SRAM start-up values, which heavily decrease the uniqueness of the microcontrollers. Furthermore, the analysis at extreme temperatures reveals a dramatic decrease of noise entropy, which makes this device type unsuitable for random number generation. This demonstrates that it is absolutely necessary to thoroughly assess the characteristics of the SRAM modules before employing them in security-related applications as some modules might not hold the required quality criteria.

References

1. ATmega328P. URL `http://www.atmel.com/devices/atmega328p.aspx?tab=parameters`. Last accessed on July 10th 2014
2. MSP430F5308. URL `http://www.ti.com/product/msp430f5308`. Last accessed on July 10th 2014
3. PandaBoard Platform. URL `http://pandaboard.org/`. Last accessed on March 24th 2014
4. PIC16F1825. URL `http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en546902`. Last accessed on July 15th 2014
5. ST STM32F100 Value Line. URL `http://www.st.com/web/catalog/mmc/FM141/SC1169/SS1031/LN775`. Last accessed on March 24th 2014
6. Stellaris LM4F120 LaunchPad Evaluation Kit. URL `http://www.ti.com/tool/ek-lm4f120xl`. Last accessed on March 24th 2014
7. Barker, E., Kelsey, J.: Recommendation for Random Number Generation Using Deterministic Random Bit Generators. NIST Special Publication 800-90A (2012)
8. Bösch, C., Guajardo, J., Sadeghi, A.R., Shokrollahi, J., Tuyls, P.: Efficient Helper Data Key Extractor on FPGAs. In: Proceedings of the 10th International Workshop on Cryptographic Hardware and Embedded Systems (CHES), *Lecture Notes in Computer Science*, vol. 5154, pp. 181–197. Springer-Verlag (2008)
9. Guajardo, J., Kumar, S.S., Schrijen, G.J., Tuyls, P.: FPGA Intrinsic PUFs and Their Use for IP Protection. In: Proceedings of the 9th International Workshop on Cryptographic Hardware and Embedded Systems, *Lecture Notes in Computer Science*, vol. 4727, pp. 63–80 (2007)
10. Holcomb, D.E., Burleson, W.P., Fu, K.: Power-Up SRAM State as an Identifying Fingerprint and Source of True Random Numbers. IEEE Trans. Comput. **58**(9), 1198–1210 (2009). DOI 10.1109/TC.2008.212
11. Leest, V., Sluis, E., Schrijen, G.J., Tuyls, P., Handschuh, H.: Efficient Implementation of True Random Number Generator Based on SRAM PUFs. In: Cryptography and Security: From Theory to Applications, Lecture Notes in Computer Science, pp. 300–318 (2012)
12. Maes, R., Tuyls, P., Verbauwhede, I.: Low-Overhead Implementation of a Soft Decision Helper Data Algorithm for SRAM PUFs. In: Proceedings of the 11th International Workshop on Cryptographic Hardware and Embedded Systems, CHES '09, pp. 332–347 (2009)
13. Maes, R., Verbauwhede, I.: Physically Unclonable Functions: A Study on the State of the Art and Future Research Directions. In: Towards Hardware-Intrinsic Security, Information Security and Cryptography, pp. 3–37 (2010)
14. Moro, T., Saitoh, Y., Hori, J., Kiryu, T.: Generation of Physical Random Number Using the Lowest Bit of an A-D Converter. Electronics and Communications in Japan (Part III: Fundamental Electronic Science) **89**(6), 13–21 (2006). DOI 10.1002/ecjc.20215
15. von Neumann, J.: Various Techniques Used in Connection with Random Digits. National Bureau of Standards Applied Math **12**, 36–38 (1951)
16. Petrie, C., Connelly, J.: A Noise-based IC Random Number Generator for Applications in Cryptography. Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on **47**(5), 615 –621 (2000). DOI 10.1109/81.847868
17. Schaller, A., Arul, T., van der Leest, V., Katzenbeisser, S.: Lightweight Anti-counterfeiting Solution for Low-End Commodity Hardware Using Inherent PUFs. In: Trust and Trustworthy Computing, *Lecture Notes in Computer Science*, vol. 8564, pp. 83–100 (2014)
18. Van Herrewege, A., van der Leest, V., Schaller, A., Katzenbeisser, S., Verbauwhede, I.: Secure PRNG Seeding on Commercial Off-the-shelf Microcontrollers. In: Proceedings of the 3rd International Workshop on Trustworthy Embedded Devices, TrustED '13 (2013)