Hardware Intrinsic Security from D flip-flops

Vincent van der Leest
Intrinsic-ID
High Tech Campus 9
5656AE Eindhoven
The Netherlands
vincent.van.der.leest@intrinsic-id.com

Geert-Jan Schrijen
Intrinsic-ID
High Tech Campus 9
5656AE Eindhoven
The Netherlands
geert.jan.schrijen@intrinsic-id.com

Helena Handschuh
Intrinsic-ID
High Tech Campus 9
5656AE Eindhoven
The Netherlands
helena.handschuh@intrinsic-id.com

Pim Tuyls
Intrinsic-ID
High Tech Campus 9
5656AE Eindhoven
The Netherlands
pim.tuyls@intrinsic-id.com

ABSTRACT

In this paper we describe the results of our investigations[1] on the randomness and reliability of D flip-flops when used as a Physically Unclonable Function (PUF). These D flip-flops are hardware components which present a random start-up value when powered up. We show that against all odds, enough randomness exists in such elements when implemented on an Application-Specific Integrated Circuit (ASIC) to turn the responses of a number of D flip-flops into a secret random sequence allowing to derive keys for use in conjunction with cryptographic algorithms. In addition to being unpredictable, these flip-flops have the advantage that they can be spread over random locations in an ASIC. This makes them very difficult to reverse-engineer when used to hide a secret key in a design at a relatively small cost in resources.

Categories and Subject Descriptors

B.7.m [**Hardware**]: Integrated Circuits—*Miscellaneous*

General Terms

Algorithms, Measurement, Reliability, Security

1. INTRODUCTION

The term Hardware Intrinsic Security (HIS) is used for security mechanisms that are based on the intrinsic hardware properties of an electronic device. Such hardware instrinsic properties are very similar in nature to human biometrics and can be seen as the unique fingerprint of an electronic circuit. In particular security mechanisms using Physically Unclonable Functions (PUFs) belong to the category of hardware intrinsic security mechanisms. Physically Unclonable Functions were introduced by Pappu [18] in 2001.

Due to deep-submicron manufacturing process variations every transistor in an integrated circuit (IC) has slightly different physical properties that lead to measurable differences in terms of its electronic properties e.g. threshold voltage, gain factor etc. Since these process variations are uncontrollable during manufacturing, the physical properties of a device cannot be copied or cloned. It is very hard, expensive and economically not viable to purposely create a device with a given electronic fingerprint. A PUF implementation requires an electronic circuit that measures the responses of hardware to certain given inputs or challenges. These responses depend on the unique physical properties of the device. Hence PUFs are functions which are easy to challenge and whose response is easy to measure, but very hard to reproduce by construction.

One of the major applications for a PUF is to use it as an identification or authentication primitive, either by using its inherent biometric properties or by using it to derive a device unique cryptographic key. This implies that keys can be stored "without storing them" and hence are present in the device only during a minimal time window (when the PUF is challenged) and therefore minimally vulnerable to attacks on non-volatile storage. In order to be able to uniquely identify a device, a PUF must both be reliable and unique. By reliable we mean the fact that one is able to reproduce the same behaviour of the function when challenged with the same input over and over again. The behavioural characteristics of electronic components depend on the environment they are exposed to, namely in terms of parameters such as the ambient temperature, the voltage ramp-up curves, high and low voltage supply, and electromagnetic interference. It is of crucial importance that the function has a stable behaviour across a range of environmental conditions. Another important aspect of reliability is the fact that the electronic component will age well, i.e. that it will not change

[1]Supported by EU FP7 project UNIQUE

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
STC'10, October 4, 2010, Chicago, Illinois, USA.
Copyright 2010 ACM 978-1-4503-0095-7/10/10 ...$10.00.

its behaviour after ten or twenty years. Typically it is observed that PUFs exhibit a noisy behaviour; this means the electronic read-out circuitry must include some type of error correction process to stabilize the PUF responses both over environmental conditions and over time. The second important aspect when using a PUF for identification or authentication purposes is that its responses should uniquely identify the device. By this we mean for instance that one device's key will never be the same as another device's key. Specifically when used for key generation a PUF should also guarantee that all generated device keys are distributed uniformly at random.

Many different PUF variants are known today. They include so-called delay PUFs such as Arbiter PUFs first described by Lee et al. in 2004 [12] and ring oscillator based PUFs first described by Gassend et al. in 2002 [6], SRAM based PUFs introduced by Guajardo et al. in 2007 [8], so-called Butterfly PUFs introduced in 2008 by Kumar et al. [11] and finally D flip-flop PUFs also introduced in 2008 by Maes et al. [14]. Implementations exist for dedicated Integrated Circuit(ICs), programmable logic devices such as Field Programmable Gate Arrays (FPGAs) and also for programmable ICs such as microcontrollers. Specific PUF variants are described in later sections of this paper and we also provide more details about their relevant properties namely when used as a cryptographic key storage and/or key generation primitive.

1.1 Related Work

The flip-flop PUF was introduced by Maes et al. in [14], where an implementation on FPGA devices was considered. The presented measurement results show that the entropy in the measured flip-flop start-up values is rather limited due to the presence of a strong bias, namely the strong preference of most of the flip-flops to start up with a zero value. Post-processing in the form of majority voting is used in [14] to reduce the observed bias. However, the resulting bit strings still contain sequences biased towards the zero value which disqualifies them for use as strong cryptographic keys. The stability of the derived bit strings under external stress conditions and over time has not been investigated either, which means one cannot tell whether such PUFs are suitable for practical real-life applications. In addition, the flip-flop PUF principle has been tested on only three devices, which does not provide enough insight into its randomness and uniqueness properties across many devices. It remains an open problem to propose a construction which can qualify flip-flops as strong random key generators on FPGAs or on ASICs, and how reliable such flip-flops are in the presence of environmental stress or over time.

1.2 Our contribution

In this paper, our investigations focus on D flip-flop PUFs implemented on ASICs. Compared to SRAM PUFs, D flip-flops present a real security advantage against invasive attacks such as probing attacks. They can be randomly spread across a design which makes it much harder for an attacker to locate them and their signal lines connecting them to the read-out and error correction circuitry. We focus on ASIC implementations as these are more secure than implementations in reconfigurable logic. As far as we know this paper proposes the first investigation of ASIC implementations of D flip-flop PUFs.

In order to be able to exploit these PUFs for practical purposes, their reliability must be tested. We describe how our devices behave when subjected to extensive tests across highly varying temperature conditions and static ageing conditions. Our main result in this area is to show that D flip-flop PUFs on ASICs are by far sufficiently stable to be used for practical applications.

Another contribution of our experiments is that we are able to obtain measurements on many more devices than previous experiments on FPGAs. More specifically, in this paper PUF responses are measured from 40 different devices, fabricated in 130nm technology at the UMC semiconductor foundry. This is all the more relevant since it allows to gain more confidence in the fact that the generated bit strings are indeed device unique and randomly distributed. Note that it is impossible to predict this before the devices have been manufactured and that our findings stem from *real* test chips. After observing a strong bias in the distribution of the generated bit strings while measuring their Hamming weights, we propose a number of (non-cryptographic) post-processing methods to eliminate that bias and proceed to select the best option which allows us to completely eliminate it. Note that in principle our processing and compression methods also apply to other types of PUFs such as SRAM PUFs and Butterfly PUFs. Next we verify that the distribution of bit strings among all devices is indeed such that they can be uniquely identified. We further apply a compression algorithm to our bit strings in order to evaluate the entropy level contained in them. Measuring the exact entropy contained in these devices is difficult since the number of samples available is limited. In this paper, we use an algorithm which allows us to show that enough entropy is contained in the available bit strings to qualify them as cryptographic keys. By further feeding the resulting output bit strings of our best post-processing method into the appropriate standard NIST randomness tests, we are able to show that this method is effective[2]. With the bias being completely removed, enough entropy being present in the bit strings and all results of the performed randomness tests lying within acceptable boundaries, we can finally show that the derived bits qualify for use as strong cryptographic keys.

1.3 Organization of the paper

In section 2 we provide an overview of the general PUF framework and a selected number of existing PUFs in the literature. Section 3 provides a description of the expected characteristics and important parameters for the security of a PUF. In this section we also describe the results of our first measurements on D flip-flops and provide some statistics for the observed results. In section 4 we propose an efficient processing method to remove the bias observed in the generated bit strings and provide a thorough security analysis of our resulting construction. Section 5 concludes the paper.

2. PHYSICALLY UNCLONABLE FUNCTIONS

This section starts with a general model presenting how PUFs are used in applications and then proceeds to describe the working principle of a specific selection of such PUFs.

[2] Note that not all NIST randomness tests can be performed since a limited number of bits are available per device and we can only run our tests on a limited number of samples.

2.1 The PUF Framework

Physically Unclonable Functions are physical structures (consisting of many random components) that are easy to measure but hard to characterize. An important application of PUFs is their use as a secure cryptographic key storage mechanism [20]. In this application one can distinguish two phases: Enrollment and Key Reconstruction.

2.1.1 Enrollment

In the enrollment phase the key is programmed into a device. Hence this phase can be compared to the key programming phase for other secure key storage mechanisms based on non-volatile memory. In order to do this the PUF in the device is challenged and the measured response (called the reference PUF response) is input to a so-called fuzzy extractor [2, 3, 13]. The fuzzy extractor derives a cryptographic key from this reference PUF response and computes helper data. Later on, in the key reconstruction phase, the helper data enables the fuzzy extractor to reconstruct the exact same ("programmed") cryptographic key from a PUF response[3]. The helper data is stored in non-volatile memory attached to the device and is not sensitive (public information).

2.1.2 Key Reconstruction

In the key reconstruction phase the PUF is challenged and the measured response is fed into the fuzzy extractor. The fuzzy extractor reads out the helper data stored in non-volatile memory and derives the cryptographic key that was "programmed" during enrollment based on the helper data and the PUF response. If the measured PUF response is close enough to the reference PUF response, the original key can be successfully reconstructed.

2.1.3 Fuzzy Extractor

Internally, the fuzzy extractor performs the following steps to derive a cryptographic key from a measured PUF response:

- Information reconciliation: Use the helper data to correct errors on the measured PUF response.
- Processing: Remove any biasing (unequal distribution of zeros and ones) in the error-corrected PUF response.
- Privacy amplification: Assuming that an attacker may learn a part of the bit string, compress the resulting bit string into a cryptographic key with maximum entropy.

In this paper we mainly focus on the processing phase. We do not address the privacy amplification step as this can be achieved using well-known methods based on secure extractors [3, 20].

2.2 PUF Instantiations

In this section we describe the most recent PUF schemes that are specifically relying on the random start-up behaviour of a memory-like electronic component. Such PUF schemes are used for deriving a secret key for cryptographic purposes, without storing this key in non-volatile memory. These schemes do not make use of a challenge-response mechanism, such as some other PUFs (e.g. optical PUF). This set includes PUFs based on the random start-up behaviour of an SRAM cell, of a D flip-flop or of cross-coupled latches.

[3]Note that in principle this key can either be used as a device unique key directly or as an encryption key to further store an application or user specific key in encrypted form.

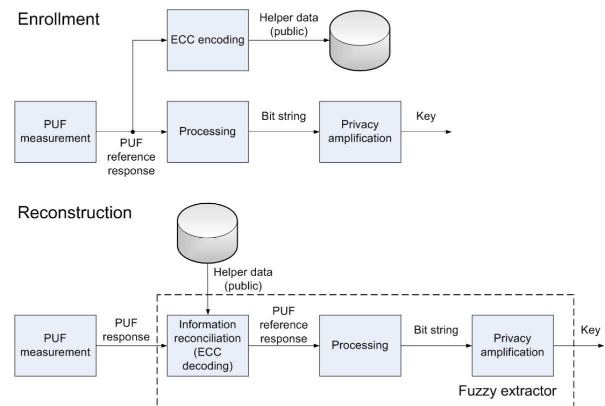

Figure 1: Enrollment and Key Reconstruction for the described PUF model.

In table 1 a systematic comparison of these memory based PUF schemes can be found. All values in the table are based on estimations by the authors of this paper or other publications. The required die area per bit is highly depending on technology. Therefore an estimation has made based on a possible number of required gates per memory cell. The entropy estimation of butterfly PUFs from [11] seems to be too high to us, while the estimation of Flip-flop PUF entropy is still preliminary. Using a more sophisticated processing method (e.g. cryptographical processing), it should be possible to derive more random bits from the Flip-flop PUF. SRAM PUFs are rare on FPGAs because the presence of uninitialized SRAM is rare in these devices.

2.2.1 SRAM PUFs

SRAM PUFs [8] use the start-up values of unitialized SRAM memory cells. An SRAM cell is typically constructed from 6 transistors: 2 access transistors connected to a circuit of 4 transistors that implement a pair of cross-coupled inverters. Each inverter consists of a PMOS and an NMOS transistor. When the SRAM cell is powered up, the strongest inverter (the one with the lowest NMOS threshold voltage or highest PMOS threshold voltage) starts switching first and will pull the memory cell to either one or zero. Roughly half of the memory cells power-up as a one value and half as a zero value. The pattern is unique for each memory and device. Different challenges are implemented by selecting different SRAM parts. SRAM PUFs are built from standard technology components that are available in every technology node.

2.2.2 Butterfly PUFs

The butterfly PUF [11] derives from the same kind of principle as the SRAM PUF. In case of butterfly PUFs, the SRAM is replaced by cross-coupled latches to construct a bi-stable cell. When powering up this cell, it is temporarily in an unstable condition after which it collapses into either the zero or the one output state. Different cells collapse

Table 1: Property comparison of different memory based PUFs

PUF type	Standard Component	Spread on die	Nr of Gates per bit	Entropy Estimation	Tested on FPGA/ASIC
SRAM	Yes	Limited	2 (=1 SRAM cell)	76% (see [7]) - 93% (see [4])	see [8]/ see [9]
Butterfly	No	Yes	8 (=2 Latches)	? (78% assumed in [11])	see [11]/unknown
Flip-flop	Yes	Yes	6 (=1 DFF)	>17% (see section 4)	see [14]/This paper

into different output states depending on variations in the production process.

2.2.3 Flip-flop PUFs

Flip-flop PUFs [14] are based on the power-up characteristic of (uninitialized) D flip-flops. Due to uncontrolled process variations, each flip-flop will have the tendency to switch its output to either the zero state or the one state when the IC is powered up. The main security advantage of a flip-flop PUF, compared to an SRAM PUF, is the fact that flip-flops are easily spread over an IC and hence are very difficult to locate by an attacker trying to reverse-engineer the chip and to probe each individual start-up bit.

3. PROPERTIES OF A PUF

When PUFs are used for security applications, they are either used as unique device identifiers, i.e. as the devices' unique unforgeable fingerprint, or as primitives to store keys in a secure way in a device. In some applications, one can also choose to use them to create a device-unique encryption key which enables to store a user-defined key in encrypted form. In all of these configurations two main properties become relevant when assessing a PUF mechanism, namely reliability and randomness.

3.1 Reliability

The first important parameter for PUFs is reliability. By this we mean that for a given device, whenever the PUF responses are measured anew, one should be able to recognize the reference measurement which was originally taken during the so-called enrollment phase. This principle is similar to human biometrics. When PUF responses are measured on the same device multiple times (either under varying or stable conditions) a number of errors (bit flips) will occur. The information reconciliation step in the fuzzy extractor algorithms allows to handle a certain amount of noise in those measurements, depending on the implemented error correction code. Smaller noise percentages in the PUF responses make it possible to use more efficient error correcting codes that require less redundant information[1].

When used in practical applications, PUFs can be subjected to all kinds of external conditions, such as high and low temperatures, high humidity and different voltage ramp-up curves. Under all these conditions the fuzzy extractor needs to be able to correctly reconstruct the cryptographic key and hence correct the errors due to noise. In this paper, we show the stability of our PUF construction with the following reliability tests:

1. **Temperature Test**: PUF responses are measured while the devices are subjected to extreme ambient temperatures varying from -40°C to +80°C.

2. **Ageing Test**: a set of devices is kept for several weeks at a high ambient temperature (+80°C) and increased core voltage (10% above normal operating level), while PUF responses are being measured every hour.

The mobility of carriers in semiconductor devices depends on temperature. Therefore temperature is likely to influence the PUF behaviour of D flip flops and hence it is important to investigate its impact. Besides the ability to function correctly under a variety of environmental conditions, it is also important to guarantee the working of the system over time. It is known that silicon slowly degrades when in use for a long time. There are several failure mechanisms that contribute to this ageing process such as electromigration, hot carrier injection and negative bias temperature instability (NBTI). Most of these failure mechanisms are accelerated when an IC is subjected to high operating temperatures and increased core voltages. The goal of the ageing test is to speed up the failure mechanisms and be able to measure the ageing effects in a relatively short amount of time.

The details of these reliability tests (also called intra-class stability tests) are discussed below. The tests have been performed on D flip-flops from test ICs that were produced on a Multi-Project-Wafer (MPW) in 130nm UMC technology. In the design of these test ICs, the flip-flops were distributed in small arrays of four and five flip-flops each. We show that the PUF responses remain stable throughout all of the performed tests. This means in particular that the observed noise levels always remain within the limits where they can still be error-corrected.

3.1.1 Temperature Test

We measured D flip-flop start-up values on 40 different ICs that were placed inside a temperature chamber. Each device contains (among other things) 1024 D flip-flops, whose start-up values are measured as a bit string of length 1024 bits. D flip-flop PUF responses were measured at the following environmental temperatures: -40°C, -20°C, 0°C, +20°C, +40°C, +60°C and +80°C. At each of these temperatures 50 PUF responses were measured on each of the 40 devices. For each device the fractional Hamming distance[4] between the measured PUF responses and a reference measurement (taken in the enrollment phase at +20°C ambient temperature) is computed. The resulting intra-class Hamming distance values are plotted in Fig. 2. The figure shows that noise levels steadily remain below 13% no matter at what temperature the measurements are taken. At most common temperatures between 0°C and +40°C, this level even reduces to about 7%. This means that the reconstructed values are extremely stable and the noise levels lie well within the acceptable boundaries for efficient error correction within the fuzzy extractor.

[4]By fractional Hamming distance we mean the Hamming distance (i.e. the number of bit flips) between the reference measurement and the sample, divided by the total length of the sample in bits.

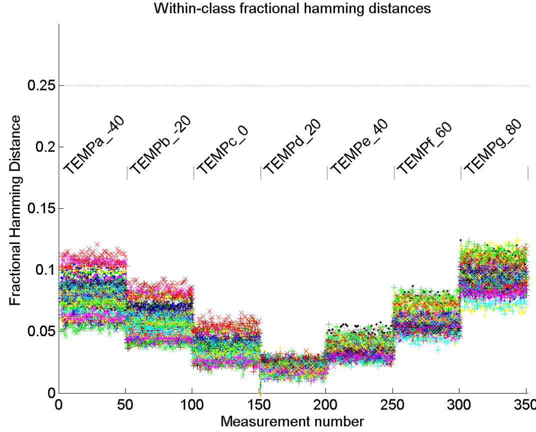

Figure 2: Fractional Hamming distance over temperature (w.r.t. enrolment at +20 ^{o}C) for 40 different ICs. At each temperature 50 PUF responses have been measured for each device.

3.1.2 Ageing Test

Five ICs have been placed in an oven that was set to an ambient temperature of $+80^{o}$C. The core voltage of the ICs was increased to 1.1*VDD, where VDD denotes the normal operating voltage of the ICs. The ICs were kept under these conditions for approximately 8 weeks continuously. Every hour the ICs were repowered and the flip-flop start-up values were measured. For each device the fractional Hamming distance between the measured PUF responses and a reference measurement (taken in the enrollment phase at $+20^{o}$C ambient temperature) is computed. The results are depicted in Fig. 3.

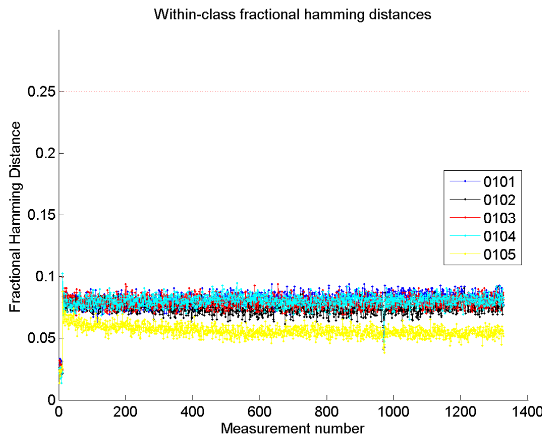

Figure 3: Fractional Hamming distance over time (spanning 8 weeks) during ageing experiment for 5 different ICs.

The figure shows that during the time the ICs are subjected to ageing conditions (high temperature and increased core voltage), the Hamming distance with respect to the reference enrollment measurements is not increasing. The fractional Hamming distance of the PUF responses remains well below 10% for all of the tested devices. This is well within the boundaries of what error correcting codes in the fuzzy extractor can correct. Although only one specific ageing condition was tested here, the results so far are very promising since no ageing effects are visible.

3.2 Randomness

An important security parameter for PUFs is randomness. This means that from a set of devices, the PUF responses of a specific device are random and unpredictable, even given all the PUF responses of the other devices in the set. The physical process used as a source of randomness should be such that

- there is enough entropy in the source across devices; this means that statistically speaking, each device is unique, and the probability that two devices have a PUF response that is "close" to each other is negligibly small.
- each PUF response is random and unpredictable; this means in particular that bits in a PUF response can only be predicted with negligible probability.

In order to assess the randomness of the PUF responses we use the following methods:

1. **Hamming Weight Test**: calculate the Hamming weight of a set of PUF responses in order to detect if these strings are biased towards zero or one.
2. **Inter-class Uniqueness Test**: compute the inter-class Hamming distances between PUF responses of different devices in order to assess whether these responses are unique.
3. **CTW Compression Test**: use the CTW (Context Tree Weighting) algorithm [22, 23, 10] in order to find out whether PUF responses can be compressed. If (lossless) compression is possible, the PUF responses do not have full entropy.
4. **NIST Randomness Tests**: use the PUF responses of different devices as input for the NIST randomness test suite [19] to see whether enough PUF responses pass these tests.

The details of these randomness tests are discussed below. All tests are performed on the enrollment PUF responses (measured at $+20^{o}$C ambient temperature) of 40 different ICs. Each PUF response consists of the 1024 start-up bits of the D flip-flops that are present in each IC.

3.2.1 Hamming Weight Test

The fractional Hamming weight of the measured PUF responses at different temperatures is depicted in Fig. 4. The Hamming weights of the enrollment measurements (that have been performed at $+20^{o}$C), are shown inside the circled area of Fig. 4.

The fractional Hamming Weights of all devices during enrollment are between 0.68 and 0.84. Since these values are significantly larger than 0.5, the measurements are clearly biased towards one.

Table 2: CTW compression result on concatenated string of 40 times 1024 flip-flop bits

Input data	Input length	Output length	Compression ratio	Optimal context length
PUF reference responses	40960	33282	81.3%	13

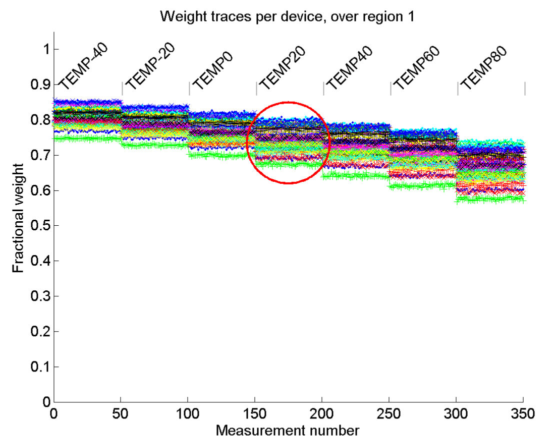

Figure 4: Hamming weight over temperature.

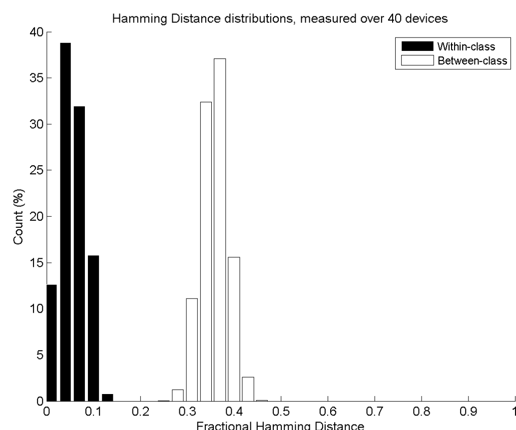

Figure 5: Intra-class and inter-class distributions of fractional Hamming distances.

3.2.2 Inter-class Uniqueness Test

When performing uniqueness tests, we are interested in finding out whether it is possible to distinguish between different devices given their PUF responses. This is mandatory when considering PUFs for authentication purposes or applications requiring unique identifiers. For our testing purposes, this means that we need to be able to show that no two devices will have the same resulting identifier or secret key; as a matter of fact the probability that two identifiers collide is so low that it is never observed in practice. This mainly depends on the entropy level available in the devices, and is a parameter which can be tuned by using more or less D flip-flops in the system. In Fig. 5, we plot the inter-class Hamming distance between different devices, and the intra-class Hamming distance between different measurements on the same device. The intra-class distribution has been computed by calculating the Hamming distance for each measurement on a device to the measurement that has been used for enrollment of that specific device. In order to create an inter-class distribution, each measurement has been compared to all measurements that are of a different device.

It can be seen that the noise level of all devices taken individually remains below 13%. When compared to one another, the devices exhibit a distance of at least 26%, which means they are sufficiently different to be able to identify each one of them uniquely.

In Fig. 6, we zoom in on the inter-class distributions. This distribution can be approximated as a Gaussian distribution, with mean $\mu = 0.36$ and standard deviation $\sigma = 0.029$.

Because of the bias in the PUF responses this distribution is not centered around 0.5, which would be the inter-class distribution with most uniqueness between devices. The fact that all PUF responses from the different devices contain more ones than zeros therefore leads to less uniqueness between devices. However all devices are still sufficiently different (the distance is at least 26%, while intra-class distribution is never larger than 13%) to be able to identify each one of them uniquely.

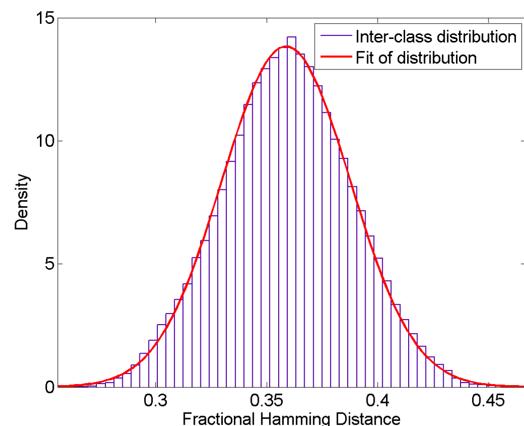

Figure 6: Inter-class distribution of fractional Hamming distances.

3.2.3 CTW Compression Test

The Context-Tree Weighting method (CTW) [22, 23, 21] is an optimal compression method for stationary ergodic sources. The compression that CTW achieves on bit strings is therefore often used as an estimator for the entropy rate, see for example [5]. We use the CTW compression method as follows. If the CTW algorithm manages to compress PUF responses, this indicates that the responses do not have full entropy[5]. Our test was conducted by first concatenating all

[5]With full entropy we mean that the entropy in bits of a certain bit string is equal to the length of the bit string.

PUF responses into one string of $40 \times 1024 = 40960$ bits. Compressing this string with CTW gave the result shown in table 2.

For each bit in a bit string that is to be compressed, the CTW algorithm needs a certain context on which it can base its internal probability tree [22, 23, 21]. We have chosen the previous bits of the same bit string as context for each bit. Furthermore we have tested several context lengths between 1 and 20 bits. The optimum context length (with which CTW achieved the best compression results) was 13 bits, as is indicated in the 5th column of table 2.

The fact that the PUF responses can be compressed by CTW, indicates that the PUF responses do not have the full entropy.

3.2.4 NIST Randomness Tests

The results of the previous tests show that the PUF responses are not random: the PUF responses are biased towards one, their inter-class fractional Hamming distance is not centered around 50% and the PUF responses can be easily compressed by CTW. Already from the fact that bits in the PUF responses are clearly biased towards one, it is clear that these PUF responses will not pass the NIST randomness tests (for instance the frequency test will not be passed). Therefore the NIST randomness tests are omitted here.

4. PROCESSING

In order to make the PUF responses usable for cryptographic applications, processing needs to be applied that removes the bias that is present in these responses. For this purpose two non-cryptographic processing methods have been examined.

The reason for using non-cryptographic processing for removing bias is that the properties of cryptographic processing will make resulting strings appear to be random, even when this is not true. Given the limited amount of data that is available for performing randomness tests (40 PUF responses of 1024 bits each), using cryptographic post-processing, such as hashing for instance, will always make the resulting strings seem random. Patterns that might occur due to limited entropy of the input of the hash can only be detected when more data is available for statistics. In a practical application however (when data is not used for randomness testing), cryptographical processing can be used to randomize PUF responses.

The processing methods that have been examined reduce the number of bits in order to increase the source's entropy per bit. The first method used is the Von Neumann extractor [17]. This method extracts randomness from the temporal ordering of ones and zeros. In this case PUF responses are scanned from left to right while reading non-overlapping successive pairs of bits. If the two bits of a pair differ, the first bit will be used in the resulting bit string. If the two bits are equal a pair will simply be ignored, while moving on to the next pair. This process is depicted in Fig. 7. The Von Neumann extractor requires a high number of bits as input, compared to the number of output bits.

Each PUF response has a length of only 1024 bits. Applying the Von Neumann extractor to these responses results in bit strings with an approximate length of 200 bits. Since biasing is not reduced completely by the extractor (i.e. the number of ones is still larger than the number of zeros in

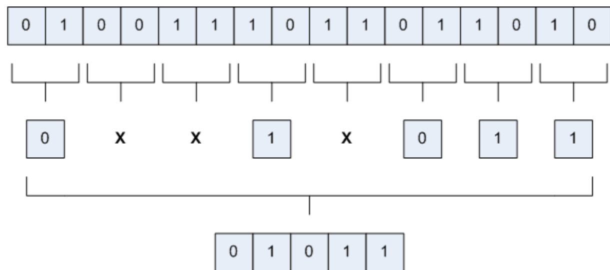

Figure 7: The Von Neumann extractor principle.

all of the resulting bit strings), a second processing step is required. It is not possible to use the Von Neumann twice on the PUF responses, because the length of the resulting bit strings will be insufficient for use in the fuzzy extractor. Therefore, a second method of processing is required.

The second method consists of splitting a string into x non-overlapping substrings of equal length. These substrings are combined into a single output bit string by XOR-ing them together. Adding two independent biased random sequences of bias ϵ together reduces the bias of the resulting sequence to $2\epsilon^2$ (piling-up lemma [15]). In our case the sequences cannot be considered independent, but adding several substrings of a fixed length together will iteratively reduce the bias to the required level. Using these two post-

Table 3: Post-processing methods

Name of resulting data set	Von Neumann	XOR	bit string length
1VNM3XOR	1	3	69
0VNM4XOR	0	4	256
0VNM5XOR	0	5	204
0VNM6XOR	0	6	170

processing methods, four different data sets were created from the PUF responses. Each data set consists of 40 bit strings with equal length. Table 3 presents these data sets and shows the way they have been formed. The name of each data set is based on the processing that has been performed to create it. For example: '1VNM3XOR' has been created by applying the Von Neumann extractor once and splitting the resulting string into 3 substrings of equal length, which are XOR-ed together in order to get the resulting set of 40 bit strings.

Table 4: Nr. of bit strings that fail freq. test

Frequency test	$\alpha = 0.1$	$\alpha = 0.05$	$\alpha = 0.01$
1VNM3XOR	5	5	2
0VNM4XOR	7	4	3
0VNM5XOR	5	4	1
0VNM6XOR	2	1	0

In order to determine which of these data sets is most likely to possess the required randomness, two basic randomness tests from [16] have been performed on all of these sets: the frequency and the serial test. The results of both tests can be found in tables 4 and 5. These tables show how many of the 40 bit strings of each data set do not pass the

Table 6: CTW compression results on derived bit strings

Input data	Input length	Output length	Compression ratio	Optimal context length
PUF reference responses	6800	6807	100%	-

Table 5: Nr. of bit strings that fail serial test

Serial test	$\alpha = 0.1$	$\alpha = 0.05$	$\alpha = 0.01$
1VNM3XOR	5	3	2
0VNM4XOR	9	7	2
0VNM5XOR	5	4	2
0VNM6XOR	1	0	0

randomness tests for different values of α. The significance level α of the test of statistical hypothesis H_0 is the probability of rejecting H_0 when it is true. In this case H_0 is the hypothesis that the input of the test has been created by a truly random source.

Based on the results from tables 4 and 5, data set '0VNM6XOR' was chosen to be most likely to have properties of a collection of random bit strings. The reason for this choice is that it is the only data set where the number of strings that fail the two basic tests is always smaller than $40 \times \alpha$.

In the remainder of this section the randomness of this data set is examined more thoroughly by applying the same methods as in section 3. The details of these randomness tests are discussed below. All tests are performed on the 40 bit strings of data set '0VNM6XOR'.Each bit string consists of 170 bits.

4.1 Hamming Weight Test

The fractional Hamming weight of the 40 bit strings is depicted in Fig. 8. As can be seen in this figure, the Hamming weights of the bit strings are now centered around 50%.

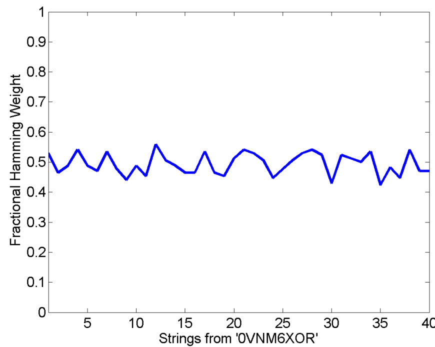

Figure 8: Hamming weight of bit strings from 40 different devices (after processing).

4.2 Inter-class Uniqueness Test

Each bit string from '0VNM6XOR' has been compared to the bit strings that were derived from all other devices by computing the mutual Hamming distance. The results are plotted as a histogram in Fig. 9. As can be seen in this figure, the distribution is perfectly centered around a mean of $\mu = 0.50$ with standard deviation $\sigma = 0.038$. This distribution shows that there is more uniqueness between the strings from '0VNM6XOR' compared to the PUF responses, because this distribution is centered around 0.5.

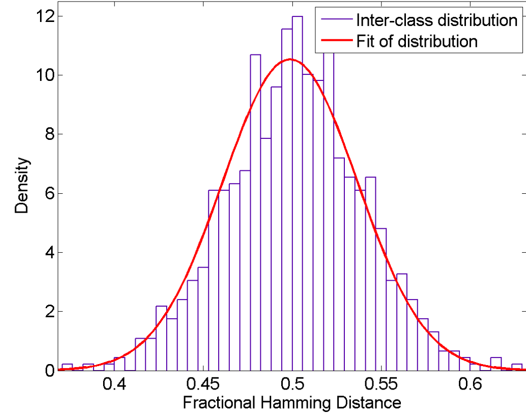

Figure 9: Inter-class distribution of keys after bias reduction.

4.3 CTW Compression Test

In order to perform the CTW compression test, we first concatenate all the bit strings into a single bit string of length $40 \times 170 = 6800$ bits. Compressing this string with CTW gives the result shown in table 6.

We have chosen the previous bits of the same bit string as a context for the compression of each bit. Furthermore we have tested several context lengths between 1 and 20 bits. All of the tested context lengths give the same compression result in this case. This clearly shows that CTW cannot further compress the derived bit strings.

4.4 NIST Randomness Tests

In order to evaluate the randomness of data set '0VNM6XOR', randomness tests from the Special Publication 800-22 issued by the National Institute of Standards and Technology (NIST) [19] have been used. The significance level of each test in NIST SP800-22 is set to 1%, which means that 99% of the test samples will pass the tests if the random numbers are truly random. We evaluate the passing ratio of tests with 40 samples. When the number of samples are n and the probability of passing each test is p , then the number of samples that pass the test follows a binomial distribution. Based on this distribution, the value of p' (observed ratio to pass test) should be between the following values ([19]):

$$p' = p \pm 3\sqrt{\frac{p(1-p)}{n}} = 0.99 \pm 3\sqrt{\frac{0.99 \times 0.01}{40}} \geq 0.9428 \quad (1)$$

Furthermore, a P-value is introduced to evaluate whether the sequence of results per randomness tests are uniformly distributed in order to indicate randomness. This uniformity

Table 7: NIST randomness tests

NIST test	p'	P-value	PASS/FAIL	Settings
Frequency (monobit) test	1.000	0.057146	PASS	
Frequency test within a block	1.000	0.534146	PASS	block-size = 20
Runs test	0.975	0.090936	PASS	
Test for longest run of ones in block	1.000	0.242986	PASS	block-size = 8
Serial test	1.000	0.015065	PASS	m = 4
Approximate entropy test	1.000	0.242986	PASS	m = 4
Cumulative sums (Cusum) test	1.000	0.048716	PASS	mode = forward

is determined by a χ^2 test, which produces the P-value. In order to indicate randomness, this P-value should be at least 0.0001 ([19]). Therefore a NIST test is only passed when:

$$p' \geq 0.9428 \ \cap \ \text{P-value} \geq 0.0001 \quad (2)$$

The results from the performed NIST randomness tests can be found in table 7. In this table it can be seen that the strings from data set '0VNM6XOR' pass all the NIST randomness tests that we are able to perform. Note that we have limited our test set to those tests that can be performed with bit strings of length 170 bits. The NIST randomness tests that require larger input bit strings have been omitted.

5. CONCLUSIONS

In this paper we have shown that after all D flip-flops contain sufficient randomness in their start-up behaviour to qualify as a strong, randomly distributed but reliable Physically Unclonable Function. We have presented a processing method that is able to remove the bias that is naturally present in the start-up values of D flip-flops: splitting the PUF responses into 6 parts and XOR-ing those parts into a single bit string. After applying this method, the derived bit strings pass the NIST randomness tests. We have assessed the reliability of D flip-flop PUFs over a large range of different temperatures and have shown that static ageing does not degrade the PUF responses. As a result we believe that D flip-flops offer great promise for implementing secure key storage and key derivation mechanisms for security sensitive applications in which one is reluctant to store the key within the system when the power is turned off.

6. REFERENCES

[1] C. Bosch, J. Guajardo, A. Sadeghi, J. Shokrollahi, and P. Tuyls. Efficient helper data key extractor on fpgas. In *Cryptographic Hardware and Embedded Systems, CHES 2008*, pages 181–197, 2004.

[2] X. Boyen. Reusable cryptographic fuzzy extractors. In *Proceedings of the 11th ACM Conference on Computer and Communications Security (CCS 2004)*, pages 82–91, 2004.

[3] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, 2008.

[4] I. Edwards, P. Newell, and C. Trufan. *SRAM PUF Analysis and Fuzzy Extractors.* (http://users.wpi.edu/˜ martin/ MQP/edwardsetal.pdf), 2010.

[5] Y. Gao, I. Kontoyiannis, and E. Bienenstock. Estimating the entropy of binary time series: Methodology, some theory and a simulation study. *Entropy*, 10(2):71–99, 2008.

[6] B. Gassend, D. E. Clarke, M. van Dijk, and S. Devadas. Silicon physical random functions. In *Vijayalakshmi Atluri editor, Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS 2002)*, pages 148–160. ACM, 2002.

[7] J. Guajardo, S. Kumar, G. Schrijen, and P. Tuyls. Physical unclonable functions, fpgas and public-key crypto for ip protection. In *Intl. Conference on Field Programmable Logica and Applications - FPL 2007*, 2007.

[8] J. Guajardo, S. S. Kumar, G.-J. Schrijen, and P. Tuyls. Fpga intrinsic pufs and their use for ip protection. In *Pascal Paillier and Ingrid Verbauwhede, editors, Cryptographic Hardware and Embedded Systems (CHES 2007)*, volume 4727, pages 63–80. Springer-Verlag, 2007.

[9] D. Holcomb, W. Burleson, and K. Fu. Power-up sram state as an identifying fingerprint and source of true random numbers. In *IEEE Transactions on Computers Volume 58 Issue 9*, pages 1198–1210, 2009.

[10] T. Ignatenko, G.-J. Schrijen, B. Skoric, P. Tuyls, and F. Willems. Estimating the secrecy-rate of physical unclonable functions using the context-tree weighting method. In *Proceedings of International Symposium on Information Theory (ISIT 2006)*, pages 499–503. IEEE, 2006.

[11] S. S. Kumar, J. Guajardo, R. Maes, G.-J. Schrijen, and P. Tuyls. The butterfly puf: Protecting ip on every fpga. In *Mohammed Tehranipoor and Jim Plusquellic, editors, IEEE International Workshop on Hardware-Oriented Security and Trust (HOST 2008)*, pages 67–70. IEEE Computer Society, 2008.

[12] J. W. Lee, D. Lim, B. Gassend, G. E. Suh, M. van Dijk, and S. Devadas. A technique to build a secret key in integrated circuits for identification and authentication applications. In *Proceedings of the IEEE VLSI Circuits Symposium*, pages 176–179, 2004.

[13] J. Linnartz and P. Tuyls. New shielding functions to enhance privacy and prevent misuse of biometric templates. In *J. Kittler and M. Nixon, Editors, Proceedings of the 4th Conference on Audio and Video Based Biometric Person Authentication, LNCS*, volume 2688, pages 393–402. Springer-Verlag, 2003.

[14] R. Maes, P. Tuyls, and I. Verbauwhede. Intrinsic pufs from flip-flops on reconfigurable devices. In *3rd Benelux Workshop on Information and System Security (WISSec 2008)*. 17 pages, 2008.

[15] M. Matsui. Linear cryptanalysis method for des

cipher. In *Proceedings of EUROCRYPT 1993, LNCS*, volume 765, pages 386–397. Springer-Verlag, 1993.

[16] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1997.

[17] J. V. Neumann. *See http://en.wikipedia.org/wiki/Randomness_extractor.*

[18] R. S. Pappu. *Physical one-way functions.* PhD. Thesis, Massachusetts Institute of Technology, March 2001.

[19] N. I. S. T. *Special Publication 800-22, A Statistical Test Suite for Random and Pseudo-Random Number Generators for Cryptographic Applications.* (http://csrc.nist.gov/rng/), 2001.

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

[21] F. Willems. The context-tree weighting method: Extensions. *IEEE Trans. Inform. Theory 1998*, 44:792–798, 1998.

[22] F. Willems, Y. Shtarkov, and T. Tjalkens. Context tree weighting: Basic properties. *IEEE Trans. Inform. Theory 1995*, 41:653–664, 1995.

[23] F. Willems, Y. Shtarkov, and T. Tjalkens. Context weighting for general finite-context sources. *IEEE Trans. Inform. Theory 1996*, 42:1514–1520, 1996.