Single-Chip Motes and SRAM PUF: Feasibility Study

Sara Faour*, Blaz Korecic*, Mališa Vučinić*, Filip Maksimovic*,
David C. Burnett†, Paul Muhlethaler* Thomas Watteyne*

*Inria, Paris
†Portland State University, Portland, OR, USA
e-mail: first.last@inria.fr*, dburnett@pdx.edu†

***Abstract*—Physically unclonable functions (PUFs) are used as low-cost cryptographic primitives that extract keying material from manufacturing variabilities of a device. All microcontrollers have on-chip SRAM; the power-up state of SRAM cells provides one way of obtaining a random output. However, not every SRAM can be used as a PUF. SRAM PUFs need to fulfill certain requirements: randomness, reliability, unpredictability and uniqueness. In this paper, we study whether the SRAM present on a novel class of devices, single-chip motes, satisfies these requirements. Single-chip motes are wireless sensor nodes that integrate computation, communication, power and sensing on a single chip. We analyze the SRAM characteristics on nine different single-chip motes to understand the potential of using these integrated SRAMs as PUF. Our experiments on 55 kB SRAM for each mote indicate that SRAM start-up values satisfy the requirements. Therefore, these embedded memories can be used for different PUF applications, including secret key extraction and true random number generation.**

***Index Terms*—SRAM PUF, single-chip mote, hardware security, evaluation.**

I. Introduction

The advancement of the fabrication technology in the past few decades has allowed the reduction of the size of integrated circuits (ICs), enabling the development of compact, yet high-performance wireless sensor nodes, or **motes**. Taking advantage of this progress, the "Smart Dust" project [1] proposed in 1997, aimed at realizing a cheap, low-power and complete self-contained mote on a micro scale. This vision produced the Single-Chip micro Mote [2]. (SCμM), a chip measuring $2 \times 3 \times 0.3\text{mm}^3$ and weighing 4.2 mg, fabricated in 65 nm CMOS process. This chip operates without a crystal and integrates sensing, computation, and communication capabilities within its compact design. It relies on the minimum number of external components and requires only three wires: power, ground, and a bond-wire antenna.

Transitioning to massive networks of single-chip motes increases the need for reliable and secure solutions. Doing cryptography on constrained devices is no longer considered a challenge: various lightweight algorithms and authentication protocols exist [3] which have been demonstrated to run on a wide variety of constrained devices. This is made possible by different cryptographic hardware accelerators readily available in modern chips. The prerequisite of many cryptographic protocols, however, is that a device attempting to authenticate or encrypt data possesses a secret key.

Existing methods for securely generating and storing such secrets in a large number of devices often rely on keys stored in non-volatile memory. Using non-volatile memory becomes impractical when dealing with the massive scale of the Internet of Things (IoT), due to management complexity, cost constraints, security risks, and scalability issues.

In response to these challenges, a class of cryptographic primitives called Physically Unclonable Functions (PUFs) have been proposed as a promising technology for lightweight embedded security. PUFs rely on the inherent unique device fingerprint. Due to deep-submicron manufacturing process variations, every transistor in an IC has slightly different physical properties that lead to measurable differences in terms of its electronic properties e.g. threshold voltage, gain factor. Since these process variations are uncontrollable during manufacturing, the physical properties of a device can neither be copied nor cloned. Creating a device with a specific electronic fingerprint is very hard, costly, and economically impractical.

The only type of PUFs that can be completely implemented in software without requiring a dedicated circuit is the SRAM-based PUF. The SRAM-based PUF works because the initial state ('0' or '1') of each SRAM cell after a new power-up is determined by inherent and uncontrollable variations during the manufacturing process [4]. All microcontrollers (MCs) used in IoT systems today have on-chip SRAM. Thus, SRAM PUFs on MCs offer the potential to bootstrap security on embedded IoT nodes by generating random seeds and private keys from the uninitialized SRAM memory.

Not every SRAM implementation is suitable to serve as a PUF. It is important to analyze some important SRAM characteristics such as randomness, reliability, unpredictability and uniqueness. Tests on these characteristics were performed on commercial SRAM memories integrated in 90 nm [5] and 65 nm [6] CMOS technologies. Bohm et al. analyzed two SRAM blocks on the same microcontroller [4], showing that the structure differs between the two SRAM blocks, and as a result one of the blocks was not suitable to be used as a PUF.

SCμM has a novel embedded architecture, and its integrated SRAM has not been previously analyzed. The contribution of this paper is two-fold:

979-8-3503-6330-2/24/$31.00 ©2024 IEEE
DOI 10.1109/CrystalFreeIoT62484.2024.00009

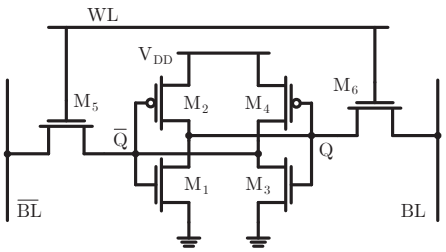

Fig. 1: 6T SRAM cell schematic.

- We survey different evaluation metrics in the literature to assess the initial SRAM PUF quality; we then present an adequate classification according to the goal of these metrics.
- We evaluate experimentally for the first time the characteristics of the initial – on start-up – SRAM memory in the context of crystal-free single-chip motes, and discuss the ability for their SRAM to be used for different PUF applications.

The remainder of this paper is organized as follows. Section II provides a background on SRAM PUF technology and its applications. Section III describes the different SRAM PUF requirements that we need to test, and discusses the importance of each requirement for different PUF applications. Section IV describes the measurement setup of our experiment, presents and discusses the evaluation results. Section V concludes the paper.

II. Preliminaries: SRAM PUF

In this section, we describe the internals of an SRAM cell and how on power up it reaches its final state, the start-up value. We then overview different SRAM PUF applications.

A. SRAM PUF technology

We show the standard 6T-SRAM cell structure in Fig. 1. It consists of two access transistors (M_5 and M_6), and two cross-coupled inverters (M_1-M_2 and M_3-M_4), which create a latch. Access to the cell is enabled by the word-line (WL) which controls the two access transistors M_5 and M_6 which, in turn, control whether the cell should be connected to the bit-lines: BL and $\overline{BL}$. They are used to transfer data for both read and write operations.

On SRAM power-up, when V_{DD} is raised, cells remain in a hold operation, meaning access transistors are cut-off (WL = 0 V), and thus both internal cell nodes are isolated from the bit-lines (BL and $\overline{BL}$). Provided that initially $V_{DD} = 0$, both SRAM internal nodes are discharged (Q = $\overline{Q}$ = 0 V) and, as V_{DD} increases, Q and $\overline{Q}$ voltages follow the power supply ramp, increasing its voltage. An SRAM cell has three equilibrium states: logic '1' and logic '0', which are stable, and a third one known as the meta-stable state. For low-power supply voltages, the cell is near its meta-stable state, rendering it incapable of retaining a definitive value as it could transition to either of the two potential final states ('1' or '0'). This situation continues until one of the internal nodes' voltage becomes slightly closer to one of the stable states than the other. When the power supply voltage reaches that specific value (flip-point), the final reached state will depend on several factors, including the inverter transistor mismatch.

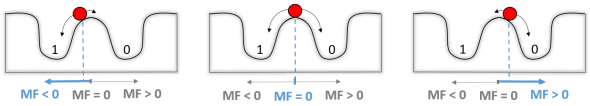

Fig. 2: The three states of an SRAM cell on start-up having negative, null and positive mismatch factor (MF) values, resp.

An SRAM-based PUF is achieved relying on this intrinsic mismatch between the transistors that form the SRAM cell. One of the main sources of this mismatch is caused by random threshold voltage (V_{th}) variability of the transistors [7]. The mismatch factor (MF) measures the variation between SRAM cell inverters, as a function of the threshold voltage (V_{th}) of the four transistors that form the cross coupled inverters of the cell. When MF is near 0, the cell shows an unpredictable behavior, depending on the amount of noise present, and it is considered *unstable*. Despite this, all cells with high enough MF absolute values show on start-up values that can be predicted by the MF parameter, and these cells are considered *stable*. If MF is negative, the cell will start-up to '1'. Conversely, if MF is positive the cell will start-up to '0'. These three states are illustrated in Fig. 2.

B. Applications

Generating private keys and random seeds are the two main scenarios for using SRAM PUFs. They constitute the base for other SRAM PUF based applications including cryptography, identification and authentication [7].

Secure Key Generation. The most common application of SRAM PUFs is generating volatile secret keys that can be used for cryptographic operations [8]. For cryptographic purposes it is essential that the generated key is reliable, meaning that the key is always exactly the same for the same PUF. A bit error rate (BER) of at least 10^{-6} is assumed to be quite conservative for any realistic application [9]. Since the initial values of SRAM at start-up impose a higher BER (10% and more), techniques using error correction codes (ECCs) [7], pre-selection methods [10], or hardening [11] are used to reduce the BER of the PUF. A deployment of the first two techniques proceeds in two phases, enrollment and reconstruction. The enrollment produces helper data [12], which is later used to reconstruct the PUF value. Helper data is publicly stored in non-volatile memory. Additionally, the extracted key should be unique and random in order to be considered as secure.

True Random Number Generation. The unstable cells in a SRAM memory are influenced by the electrical noise in the circuit. This variation in the unstable cells values results in noisy measurements of start-up pattern on SRAM. By utilizing these noises, SRAM PUF can serve as a True Random Number Generator (TRNG) to provide a seed or refresh value to cryptographic protocols [13]. In this application, the noisy output of the SRAM should offer sufficient randomness,

meaning that unstable cells are independent and do not present any pattern. Moreover, noise should be unpredictable, such that by knowing the values of unstable cells during the previous power-ups, the next power-up value cannot be foreseen. A random number generator streaming seeds with bad randomness and unpredictability would directly reduce the security level of a cryptographic mechanism using such seeds.

III. SRAM PUF Requirements

Not every SRAM is appropriate for serving as a PUF. In order to be able to use a PUF in security applications, it should possess few properties that determine the quality of the PUF. Several metrics exist in the literature to quantify the properties of PUFs [7], [14]. In this paper, we classify the metrics according to properties they serve for. These properties are randomness, reliability, unpredictability and uniqueness.

A. Randomness

Randomness is evaluated to assess the feasibility of using the PUF as a random source. The binary output of an SRAM PUF should achieve a balanced and unbiased distribution of bits, for this purpose we first calculate the fractional Hamming Weight. Then, we make sure that the PUF output lacks non-random patterns or dependencies by deriving the correlation between the bits. These two randomness-related metrics are presented below.

- **Fractional Hamming Weight**[1]: This metric evaluates the *bias* of SRAM PUF towards either 0 or 1. It constitutes the *mean* of the power-up values of the SRAM, used in order to assess the distribution and balance of bits in the SRAM PUF. If the SRAM PUF is perfectly unbiased, the mean should be 0.5. The mean of n power-up values x_i can be calculated using the following formula:

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{1}$$

- **Correlation Between Bits**: The goal of calculating the correlation between bits in SRAM PUF is to primarily assess the level of independence and randomness in the generated bit sequences. If the output bits were highly correlated, it could potentially introduce patterns or predictability that might be exploited by an adversary. The correlation between n bits with a distance (lag) of j can be expressed using the unbiased autocorrelation estimator for binary sequences [15], defined by:

$$r_j = \frac{1}{n-j}\sum_{i=1}^{n-j} x_i x_{i+j} \tag{2}$$

for $0 \leq j \leq n-1$. To make zero as the neutral element, zero bits values are substituted by -1. So $r_j = 1$ and $r_j = -1$ means that the bits are totally correlated, $r_j = 0$ means that the bits of distance of k are totally uncorrelated, and in this case $n - j$ should be even. Please note that the implementation of the autocorrelation formula can change slightly according to the specific implementation of the library used in programming, and to parameters configuration.

B. Reliability

Reliability is another important property for PUFs to be studied. It measures the stability or consistency when some environmental conditions change, such as ambient temperature, supply voltage, etc. The variations in the SRAM start-up values due to environmental changes are influenced by key transistor parameters like threshold voltage, leakage current, delay, and others. When PUF is used to extract keys, it is crucial to minimize the impact of these variations for two primary reasons. Firstly, the device must exhibit inherent resistance as it can be naturally exposed to environmental changes. Additionally, an attacker should not be able to leak information from the device by simply manipulating, for instance, the temperature. We have identified three different tests for evaluating the reliability of the PUF.

- **Intra-Chip Hamming Distance**[2] **(HD_{intra})**: This measure evaluates the number of PUF output bits that change when re-generated again from the same chip with or without environmental changes. It is also known as the *error rate*, indicating the reliability of the PUF output. When PUF is used to generate a secret key, the intra-chip hamming distance should be ideally 0%. However, in practice, an error rate of 10^{-6} is considered sufficient [9]. HD_{intra} can be measured using the Hamming distance with respect to a reference vector. The first read-out pattern is used as the reference, and the other measurements of the same chip are compared to this reference using fractional Hamming Distance.
- **Stability**: The stability of an SRAM PUF can be evaluated from the fraction between the number of stable SRAM cells and the total number of cells. An SRAM cell is considered as stable when it is always powered up to state 0 or 1 over a large number of power-ups. In a more quantitative approach, the stability of SRAM cells is evaluated by introducing the empirical *one-probability* term. The one-probability (p_i) of a specific cell i within an SRAM is the probability that the response value of this cell (X_i) is '1' across multiple power-up instances. Formally, it is defined as follows:

$$p_i := \Pr(X_i = 1) \tag{3}$$

C. Unpredictability

Unpredictability has a wider meaning than randomness, and an unpredictable sequence may or may not be truly random. This property refers to the inability to foresee future outcomes of SRAM PUFs based on previous observations, and it is especially important when the SRAM PUF is used to generate random numbers. An SRAM PUF output is influenced by noise, hence two measurements of start-up values on an SRAM will present several different bits, even under the same

[1]Hamming Weight (HW) is defined as the number of non-zero bits in a bit string. A fractional HW is divided by the length of the string.

[2]Hamming Distance (HD) is defined as the number of different bits between two bit strings. A fractional HD is divided by the length of the string.

external conditions. To leverage this phenomenon as a source of randomness, a sufficient amount of bits should change between measurements. Furthermore, the bits changed should be unpredictable. To asses the unpredictability, the entropy metric is used. The lower bound of PUF entropy, called min. entropy [16], quantifies the minimum amount of information needed to describe the source of randomness.

Intra-Chip Min. Entropy: This metric is used to assess intra-chip variations across multiple measurements of the SRAM PUF on the same chip. An unstable cell is sensitive to noise and thus contributes to this entropy, thus this metric is also known as *noise entropy*. In the context of security, SRAM PUFs with high minimum entropy are desirable for TRNG, ensuring the generated numbers are difficult to guess or replicate. Given a binary source with probabilities p_0 and p_1 for producing '0' and '1' respectively, the min. entropy of this binary source is:

$$H_{min} = -\log_2(\max(p_0, p_1)) \tag{4}$$

Assuming all bits from the SRAM PUF are independent, each cell location can be regarded as an individual binary source. To estimate the min. entropy of noise of one bit cell, we use the one-probability p_i defined in (3) and obtained over multiple power-ups of SRAM PUF on the same chip. Where an ideal probability of $p_i = 0.5$ (very unstable cell) maximizes the min. entropy to $H_{min,noise} = 1$. Hence, the average min. entropy of noise on a single SRAM PUF over n bits is:

$$(H_{min,noise})_{avg} = \frac{1}{n}\sum_{i=1}^{n} -\log_2(\max(p_i, 1-p_i)) \tag{5}$$

D. Uniqueness

Each chip should be unique due to the inherent fabrication variability, meaning that the probability for two chips having a PUF response close to each other is negligible. This uniqueness is crucial for applications like cryptographic key generation, secure device authentication, and anti-counterfeiting measures, as it enhances security, prevents unauthorized access, and facilitates the identification and verification of individual chips. The dissimilarity between PUFs can be evaluated using both the Hamming Distance and entropy as follows.

- **Inter-Chip Hamming Distance (HD_{inter})**: This measure evaluates the number of different output bits between SRAM PUFs on two different chips. It helps assess the uniqueness of a PUF, so that the PUF of a certain chip is distinguishable from others. If the PUF produces uniformly distributed independent random bits, the inter-chip variation should be 50% on average. HD_{inter} can be calculated using the fractional Hamming Distance between the first SRAM read-out of every two different chips. Measuring this metric necessitates obtaining numerous samples, a task that proves particularly challenging due to the involvement of numerous chips.
- **Inter-Chip Min. Entropy**: This metric is used to assess inter-chip variations between the SRAM PUF of multiple chips. It aims at evaluating device uniqueness and the impact of bias. Assuming all bits from different SRAM PUFs are independent, for each bit location i, p_{i_0} and p_{i_1} are computed as probabilities for producing '0' and '1' respectively, over all measured SRAMs. An ideal probability of $p_{i\_0} = 0.5$ (uncorrelated cells from different SRAMs at the same address), maximizes the min. entropy to $H_{min,PUF} = 1$. The average min. entropy of SRAM PUFs over n bits is:

$$(H_{min,PUF})_{avg} = \frac{1}{n}\sum_{i=1}^{n} -\log_2(\max(p_{i\_0}, p_{i\_1})) \tag{6}$$

IV. Experimental evaluation of SRAM PUF on SCμM

The experimental implementation was done on single-chip mote version SCμM-3C [2]. The chip contains a Cortex-M0 microprocessor with 64 kB each of program and data SRAM, where the data SRAM has a width of 32 bits and a depth of 16384. SCμM-3C does not include a flash memory, so every time the power is removed from the chip, both the program and data stored in SRAM are lost. To flash SCμM-3C – placed on Sulu v2.0 experimental board – we use a nRF52840-DK commercial development kit to send the firmware binary file from the laptop to the chip. The program SRAM is used to store the firmware, thus, we can only use the data SRAM memory to read the uninitialized cells at start-up.

In our experiment we evaluated 55 kB of data SRAM memory starting at address `0x20002400`, on each of nine SCμM chips. For each chip, we collected 1,000 readings of the SRAM start-up values at ambient temperature. To execute multiple power-ups cycles, we used a programmable Yepkit USB Switchable Hub, connected on its output USB ports to the nRF52840-DK and SCμM circuit, and on its input to the laptop USB ports serving as power supply source. For every power cycle, the power down time was 5 s. If the power down period is too short, then the data will deterministically revert to the previous written state, and SRAM contents will not be completely erased. Liu et al. [10] have shown previously that an SRAM cell requires at most 600 ms to completely collapse and to power up to its uninitialized state. Thus, 5 s provide a comfort power down interval. As an online addition to this paper, the code used to collect[3] and analyze[4] the data is available on GitHub.

The analysis of our data brought up the following results:

A. Randomness

1) Fractional Hamming Weight (Mean): The mean of the power-up values in a SRAM follows a binomial distribution, since each SRAM cell is considered as an independent and identically distributed Bernoulli random variable. The probability density function (PDF) of the binomial distribution is plotted in Fig. 3, and it is given by $P_k = \binom{n}{k} p^k q^{n-k}$, where n is the total number of bits (n = 450,560), k is the number of ones (0 to n) and $p = q = 0.5$ is the probability of one/zero. Since n is large, the binomial can be approximated by the normal distribution. k is normalized by $\frac{1}{n}$ on the figure. The mean of the power-up values should

[3]https://github.com/bkorecic/scum-automated-sram-read

[4]https://github.com/bkorecic/scum-sram-evaluation

TABLE I: Comparison of SRAM PUF Characteristics on different SCμM-3C Chips

Chip ID	BER (%)			Hamming Weight			Correlation Between Bits		Noise Entropy (%)	Stability (%)
	$E_{\min}$	$E_{\max}$	E_{avg}	$W_{\min}$	$W_{\max}$	W_{avg}	$C_{\max}$	C_{avg}	H_{avg}	S_{avg}
L45	4.59	4.77	4.67	0.49848	0.49996	0.49930	0.03130	2.46×10^{-7}	5.703	73.696
M2	4.54	4.77	4.64	0.49837	0.49992	0.49918	0.03993	1.94×10^{-6}	5.618	74.439
M17	5.06	5.22	5.13	0.49905	0.50058	0.49987	0.04945	4.29×10^{-8}	6.330	71.121
M22	4.96	5.15	5.06	0.49863	0.50042	0.49946	0.07453	5.53×10^{-7}	6.199	71.694
M39	4.42	4.61	4.52	0.50001	0.50153	0.50081	0.02623	1.51×10^{-6}	5.536	74.447
M42	4.98	5.13	5.06	0.49795	0.49954	0.49872	0.03153	1.62×10^{-6}	6.152	71.726
M44	4.60	4.79	4.69	0.49931	0.50081	0.50008	0.03657	2.46×10^{-8}	5.583	74.149
M47	4.44	4.71	4.57	0.49952	0.50104	0.50021	0.07534	5.54×10^{-8}	5.602	74.155
M49	4.36	4.54	4.45	0.49970	0.50105	0.50040	0.02377	1.07×10^{-7}	5.430	74.805

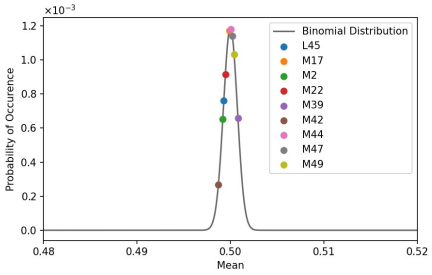

Fig. 3: Average mean values of SRAM PUF on different chips.

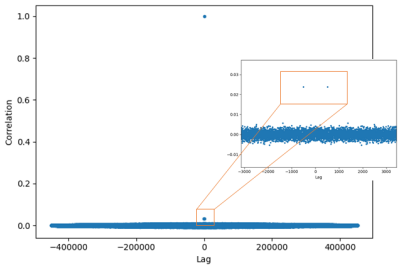

Fig. 4: Auto-correlation of chip L45.

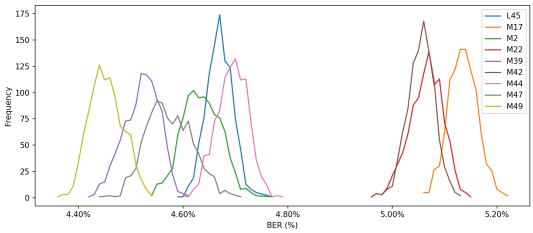

Fig. 5: Bit Error Rate (BER) of SRAM PUF on different chips.

ideally be 0.5. Table I gives the resulting mean values averaged over 1,000 runs for each chip. As shown in Fig. 3, the results of all the chips fall within an acceptable range on the binomial PDF. The confidence interval (CI) can be used to determine whether the available data lie within an expected range. By considering a confidence interval of 95%, CI can be determined as $[0.49854, 0.50146]$. Therefore, all averaged mean values (W_{avg}) of the test chips lie within this confidential interval. Hence, all the studied chips are *unbiased*.

2) Correlation Between Bits: The measurement results of the correlation between single cells for seven of the chips are similar, and we show the auto-correlation results for one of these chips - L45 - in Fig. 4. The data are correlated only at lag, or distance $j = 0$, and the auto-correlation is negligible for any other lag value. The other two chips - M22 and M47 - shows slightly higher auto-correlation outlier values compared to the other chips. We noticed a common trend for all nine chips: for lag $j = \pm 512$, the auto-correlation values are slightly higher than all of the other values (except $j = 0$). The aforementioned two points are zoomed in Fig. 4. This interesting remark can be attributed to the existence of a very small correlation between SRAM cells on the starting address `0x20002400` and the cells 512 bits away, equivalent to four raw bits away, according to the 32 bits width of the memory. Based on all the results we had, the worst correlation coefficient was given by the chip M47 at lag ± 512, and it is equal to 0.07534. According to the scale of Pearson's Correlation Coefficient (PCC), this value is below 0.19, hence it is considered negligible. This signifies a high degree of *uncorrelation* and randomness in any sequence extracted from SRAM, whether it is a key or a random seed.

B. Reliability

1) Intra-Chip Hamming Distance (BER): In an ideal PUF, the BER value is 0, but due to the influence of noise this value exceeds 0. The intra-chip hamming distance HD_{intra} is shown in Fig. 5 for 1,000 runs with respect to a reference vector, defined as the first output PUF. We notice that the BER over multiple readings of the same SRAM PUF approaches a normal distribution. This can be attributed to the central limit theorem and the nature of the physical variations including the electrical noise and temperature that cause an unstable SRAM cell to flip from one value to another during the next power-up. The average HD_{intra} values at different chips are shown in Table I. The obtained BER values are close, indicating a similar structure between different SRAM memories. These results necessitates using error correction or other techniques to compensate for the ≈4.75% BER for key generation applications. Note that the 4.75% value is far below the acceptable boundaries (approximately 25% errors) for efficient error correction within the fuzzy extractor, where the efficiency is measured in terms of required hardware resources [9].

2) Stability: The percentage of stable bits - that never change their values - is given in Table I. Thus, about 26% of the bits on a memory are causing the errors quantified using the BER. By calculating the one-probability defined in (3), the tendency for each SRAM cell to start-up with either "1" (red) or "0" (blue) is illustrated in Fig. 6. There are always a few bits flipping for each iteration and therefore the stability of 100% required for generating reliable secret keys could not be achieved. Selecting only stable cells can help increasing the stability of a SRAM PUF [10], and thus to generate a reliable key. Otherwise, we can deal with the existing instability by using ECC to correct the errors occurring. From a TRNG application perspective, more unstable SRAM cells means that more randomness can be harvested based on SRAM PUFs. The length of the extracted random seed can be quantified using the min. entropy, a metric that will be evaluated next.

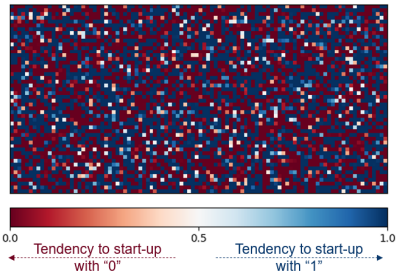

Fig. 6: Visualization of the probability of each cell to start up with "1", for 5k memory on chip M42.

C. Unpredictability

1) Intra-Chip Min. Entropy: This metric measures the unpredictability of next SRAM power-up response given the information of prior responses, mainly known as the noise entropy. Using 1000 consecutive measurement data, our results are shown in Table I. Unstable SRAM cells causes the start-up read-out to be noisy at each run. This noise can be used to derive a seed for a TRNG. To make sure that the seed is truly random, the SRAM pattern should be conditioned (e.g. hashed, compressed, etc.) in order to reach the full entropy target of 100%. Our experimental results show that SRAM PUFs can serve around 5.8% of noise entropy on average. This amount of entropy is feasible to be conditioned and then used as a source of randomness for a TRNG according to a previous work [13]. On the other side, if SRAM PUF is used to extract secret keys, then the noise needs to be removed for reliably reproducing the exact key.

D. Uniqueness

1) Inter-Chip Hamming Distance: An average inter-chip hamming distance HD_{inter} of 49.93% was obtained between node pairs. The minimum HD_{inter} value is 48.3%, and the maximum is 50.64%. Since the number of available chips is small these results can only provide a hint. The obtained values are close to 50% and they show high uncorrelation between chips, which enables the generation of unique keys and seeds among different SRAM PUFs.

2) Inter-Chip Min. Entropy: This metric is the lower bound of PUF entropy calculated based on single measurement of multiple SRAMs, evaluating the uniqueness of SRAM PUFs. The value found by this test for different memories is a very conservative estimate, since this value would increase with more devices. We will need more nodes to reach the min. entropy convergence [14]. This reduces the inter-chip min. entropy in our experiment to 66.46%, and thereby the number of unpredictable bits per chip. In the literature, inter-chip min. entropy values between 64% [16] and 90% [17] are reported. Higher values means less required input SRAM data to generate both secure keys and random seeds. Our conservative entropy indicates that to generate 128 unique bits with maximum entropy, we need at most about 200 bits.

V. Conclusion

In this paper, we have tested embedded SRAM memory on nine different SCμM chips, to study the feasibility of this memory to be used as PUF. We used various evaluation metrics from the literature and implemented them to analyze 1,000 read-outs obtained from SRAM start-up values on each chip. Our experimental results reveal that all of the tested SRAM memories are applicable to either generate reliable secure keys by providing reliability, or generate random entropy source by providing unpredictability. In fact, the tested SRAM PUFs have an error rate of ≈4.75% and unpredictability of ≈5.8%. But those values can be adjusted using different methods to achieve the desirable performance. Additionally, the SRAM start-up values were proved to be unique, unbiased and random within the same chip, and between different chips. These properties are also crucial for PUF applications.

References

[1] B. Warneke, M. Last *et al.*, "Smart Dust: Communicating with a cubic-millimeter Computer," *Computer*, vol. 34, no. 1, pp. 44–51, 2001.

[2] F. Maksimovic, B. Wheeler, D. C. Burnett *et al.*, "A Crystal-Free Single-Chip Micro Mote with Integrated 802.15.4 Compatible Transceiver, sub-mW BLE Compatible Beacon Transmitter, and Cortex M0," in *2019 Symposium on VLSI Circuits*. IEEE, 2019, pp. C88–C89.

[3] M. Vučinić, G. Selander *et al.*, "Lightweight Authenticated Key Exchange with EDHOC," *Computer*, vol. 55, no. 4, pp. 94–100, 2022.

[4] C. Böhm, M. Hofer, and W. Pribyl, "A Microcontroller SRAM-PUF," in *2011 5th International Conference on Network and System Security*. IEEE, 2011, pp. 269–273.

[5] G. Selimis, M. Konijnenburg, M. Ashouei *et al.*, "Evaluation of 90nm 6T-SRAM as Physical Unclonable Function for Secure Key Generation in Wireless Sensor Nodes," in *2011 IEEE International Symposium of Circuits and Systems (ISCAS)*, 2011, pp. 567–570.

[6] M. Claes, V. Van Der Leest, and A. Braeken, "Comparison of SRAM and FF PUF in 65nm Technology," in *Information Security Technology for Applications: 16th Nordic Conference on Secure IT Systems, NordSec 2011, Tallinn, Estonia, October 26-28, 2011*. Springer, 2012, pp. 47–64.

[7] C. Böhm and M. Hofer, *Physical Unclonable Functions in Theory and Practice*. Springer Science & Business Media, 2012.

[8] Y. Gao, Y. Su, W. Yang *et al.*, "Building Secure SRAM PUF Key Generators on Resource Constrained Devices," in *2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. IEEE, 2019, pp. 912–917.

[9] C. Bösch, J. Guajardo, A.-R. Sadeghi *et al.*, "Efficient Helper Data Key Extractor on FPGAs," vol. 5154, 2008, pp. 181–197.

[10] M. Liu, C. Zhou *et al.*, "A data remanence based approach to generate 100% stable keys from an SRAM physical unclonable function," in *2017 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, 2017, pp. 1–6.

[11] H. Shinohara, B. Zheng, Y. Piao *et al.*, "Analysis and Reduction of SRAM PUF Bit Error Rate," in *2017 international symposium on VLSI design, automation and test (VLSI-DAT)*. IEEE, 2017, pp. 1–4.

[12] J. Delvaux *et al.*, "Helper Data Algorithms for PUF-Based Key Generation: Overview and Analysis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 6, pp. 889–902, 2015.

[13] V. Van der Leest *et al.*, "Efficient Implementation of True Random Number Generator based on SRAM PUFs," in *Cryptography and Security: From Theory to Applications*. Springer, 2012, pp. 300–318.

[14] P. Kietzmann *et al.*, "PUF for the Commons: Enhancing Embedded Security on the OS Level," *arXiv preprint arXiv:2301.07048*, 2023.

[15] I. D. Mercer, "Autocorrelations of Random Binary Sequences," *Combinatorics, Probability and Computing*, vol. 15, no. 5, pp. 663–671, 2006.

[16] R. Wang *et al.*, "Long-Term Continuous Assessment of SRAM PUF and source of Random Numbers," in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2020, pp. 7–12.

[17] P. Koeberl *et al.*, "Entropy loss in PUF-based key generation schemes: The repetition code pitfall," in *2014 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, 2014, pp. 44–49.