A Spatial Majority Voting Technique to Reduce Error Rate of Physically Unclonable Functions

Patrick Koeberl, Jiangtao Li, and Wei Wu

Intel Labs, Intel Corporation
{patrick.koeberl,jiangtao.li,wei.a.wu}@intel.com

Abstract. The Physically Unclonable Function (PUF) is a promising hardware security primitive with a wide range of applications, such as secure key generation, device authentication, IP protection, and hardware entangled cryptography. Due to their physical construction, PUF responses are inherently noisy. Error correction codes can be used to turn noisy PUF responses into keys or static values for these applications. However, a general construction of error correction is expensive and could introduce high entropy loss for PUFs with high error rates. Some PUF pre-processing techniques have been proposed, such as temporal majority voting and dark bit schemes, applied before error correction. In this paper, we introduce a simple and yet effective method to reduce PUF error rate called Spatial Majority Voting (SMV). The idea is to group PUF bits together to produce a single, more stable bit from the group. Experimental data show that SMV works very well, reducing the mean error rate from 6.5% to 0.3% with a group size of 9 on SRAM PUFs implemented in 65 nm CMOS. We also show that SMV can be combined with the dark bits method to further reduce the error rate to less than 0.01%, thus avoiding the need for expensive error correction schemes.

1 Introduction

A Physically Unclonable Function (PUF) can be described as a physical system which when measured provides unique, unpredictable, and repeatable responses. Creating a physical copy of the PUF with an identical behavior is hard, thus resulting in a structure which is unclonable even by the manufacturer. Pappu introduced the PUF concept in his thesis [19]. In 2002, Gassend et al. introduced silicon PUFs in [6]. Silicon PUFs exploit the uncontrollable manufacturing variations which are a result of the integrated circuit fabrication process. Manufacturing variation of process parameters such as dopant concentrations and line widths manifest themselves as differences in timing behavior between physical instances of the same integrated circuit design.

Since the first development of silicon PUFs, a number of silicon PUF constructions have been proposed. Lee et al. [12] proposed the first arbiter PUF in 2004. Guajardo et al. [7] proposed the SRAM PUF in 2007. In 2008 Kumar et al. [11] introduced Butterfly PUFs and Maes et al. [17] proposed D type Flip-Flop PUFs. In 2012 Simons et al. [20] proposed Buskeeper PUFs. Note that

SRAM PUFs, Butterfly PUFs, D type Flip-Flop PUFs, and Buskeeper PUFs are all memory based PUFs.

PUFs have become a promising security primitive with a wide range of applications. For example, PUFs are used for secure key generation in silicon [14,21,7,16], which eliminates the need for storing keys in non-volatile memory. It was stated in [18] that PUF based key generation provides advantages like physical unclonability and tamper evidence, compared to storing the keys in non-volatile memory. PUF has been proposed for online device authentication in a challenge-response authentication protocol [22] and for offline device authentication [10]. Recently, PUFs have been used in various hardware entangled cryptographic schemes and protocols, such as a PUF-based block cipher [1] and PUF-based oblivious transfer and key exchange protocols [4]. Many of these applications can be used for trusted computing or building trusted systems.

Since PUF responses are inherently noisy and may not be uniformly random, a post-processing algorithm is needed to convert noisy PUF responses into keys or static values in these applications. This process is known as the fuzzy extractor or helper data algorithm in the literature. Part of the fuzzy extractor is error correction. Several practical fuzzy extractor schemes [3,18] have been proposed for memory-based PUFs using various error correction codes, such as BCH, Reed-Muller, Golay codes, or repetition codes. It is easy to see that the higher error rate in the PUF response, the more error correction is required. Complex error correction codes are capable of correcting high error rates, but are much more expensive to build in hardware. Furthermore a general construction of error correction could introduce high entropy loss [5]. Therefore it is important to reduce the PUF bit error rate before applying the fuzzy extractor.

In this paper, we introduce a new technique to reduce the PUF error rate called Spatial Majority Voting (SMV). This technique is effective and simple to implement. The idea is to group a few PUF response bits together from which a subgroup is chosen. A more stable bit from the group is produced based on majority voting of the subgroup. Experimental data show SMV works well, reducing the mean error rate from 6.5% to 0.3% with a group size of 9 on SRAM PUFs implemented in 65 nm CMOS. Combining SMV with the dark bits method [1], the noisy rate in PUF can be reduced to less than 0.01%. A simple hardware implementation of Hamming codes, BCH codes (with small code size), or repetition codes can be used to remove the remaining PUF errors. Our solution opens a wide possibility of applications where expensive error correction codes are not possible, such as RFID and resource-constrained devices.

1.1 Related Work

There are techniques in the literature to reduce PUF noise rate before applying error correction or a fuzzy extractor to the PUF responses, namely, Temporal Majority Voting (TMV) [1], dark bits [1], and index based syndromes [23]. In this paper we term these methods pre-processing techniques and consider SMV similarly.

The basic idea of TMV [1] is as follows: each PUF bit is evaluated multiple times and if most of the time it is evaluated as 1, then we set this PUF bit value to 1; otherwise, we set this bit value to 0. Observe that if a PUF bit is relatively stable and only flips its value occasionally, TMV can effectively stabilize this bit. However, if a PUF bit is very unstable (e.g., probability of one half of being 0 and 1 respectively), TMV does not work well on this bit.

In the dark bits method presented by Armknecht et al. [1], each PUF bit is evaluated multiple times during the setup phase. If a PUF bit is not stable, we mark it as a "dark bit". In the evaluation phase, the PUF is evaluated again and the dark bits are discarded, as these bits were noisy during the setup phase. Observe that if a PUF bit is very unstable, it will very likely be detected as a dark bit. It is easy to see that filtering the PUF responses dark bits results in a lower error rate.

Yu and Devadas [23] provided a different technique called index based syndrome. This approach assumes the PUF output is a real value instead of a single bit. The idea is that only PUF bits with a strong representation of a "1" or a "0" are chosen. In other words, only relatively stable PUF bits are picked. This technique is only applicable to some PUF designs such as the Ring Oscillator PUF, but not applicable to memory-based PUFs with binary outputs.

Both TMV and the dark bits method require multiple PUF measurements which could be impractical in many applications according to [13]. The index based syndrome method only works for a few type of PUFs with real-valued output. In contrast, our SMV method does not require multiple PUF measurements and is applicable to all PUF types. Furthermore, the SMV method is complementary to these three techniques. We shall show in Section 5 that SMV can be combined with the dark bits method to further reduce PUF error rate significantly.

Another approach to reduce PUF error rate is to use a repetition code [3,18,2]. Using a repetition code of length n, a random bit b is repeated n times into an n-bit string and then XORed with an n-bit PUF response as the helper data. In the evaluation phase, the random bit b is recovered by XORing the helper data and the PUF response and then running a majority voting on the XORed result. Repetition codes are indeed more efficient than our SMV method in terms of PUF error rate reduction. However, there are two limitations to the repetition code method. First, repetition codes suffers high entropy loss. For an n-bit PUF response, the entropy loss is $n - 1$ and there is at most one bit of leftover entropy [5]. For low entropy PUFs, there could be zero leftover entropy using repetition coding. We believe there is less entropy loss in our SMV approach. Detailed analysis of entropy loss in SMV remains an open question and is future work. Second, the repetition code method can only be used in the code-offset construction (also known as fuzzy commitment [8] in the literature). In the application of PUF-based key generation using repetition codes, the key has to be chosen external to the device by a trusted manufacturer. In our SMV scheme, the key can be completely "unknown" to the manufacturer which is an attractive property for many applications.

1.2 Our Contributions

We summarize our contributions of this paper as follows: we provide an efficient pre-processing method to reduce PUF error rate using Spatial Majority Voting. Our SMV method is efficient to implement in hardware. We provide both theoretical analysis and experimental data to show that the SMV method works well. We note that if the raw PUF responses are biased, SMV would increase the bias in the processed PUF response. We provide an alternative SMV scheme such that the bias in PUF is not be amplified. We also present how to combine SMV with dark bits method such that the PUF error rate can be further reduced.

1.3 Organization of the Paper

The rest of this paper is organized as follows. We first introduce our SMV scheme in Section 2 with a theoretical analysis. We show the effectiveness of SMV with experimental studies on real SRAM PUF data in Section 3. We present an alternative SMV scheme for PUFs with biased responses in Section 4. In Section 5, we show that the SMV scheme can be combined with the dark bits method to further reduce the PUF error rate. We discuss the potential applications of our SMV scheme in Section 6 and conclude our paper in Section 7.

2 Basic Spatial Majority Voting Scheme

In this section, we first review a simplified PUF model and define the PUF pre-processing process. We then describe our Spatial Majority Voting (SMV) scheme and provide some theoretical analysis to show the effectiveness of SMV.

2.1 PUF Model and PUF Pre-processing Process

Roughly speaking, a PUF is a random function based on a physical system with a small amount of noise. Although there have been earlier attempts at formal definitions of PUF [19,6,7,1], we use a simplified PUF definition based on [1] with a focus on PUF stability. The other properties of PUF such as unclonablity, randomness, and tamper evidence are neglected from the following simplified model.

Definition 1 (Physically Unclonable Functions). *A (n, m, p_e)-family of physically unclonable functions is a set of probabilistic algorithms with the following procedures:*

Instantiate. *The output of the* Instan *procedure is a unique probabilistic function $f : \{0,1\}^n \to \{0,1\}^m$.*

Evaluate. *Given a physically unclonable function f, the* Eval *procedure on each challenge $x \in \{0,1\}^n$ outputs a noisy response $f(x) \in \{0,1\}^m$.*

On two separate evaluations of same f and challenge x, denoted as r_1 and r_2, the noise vector between two evaluations is $r_1 \oplus r_2$. The p_e is the average noise rate between any two PUF measurements.

Assuming the PUF noise is randomly distributed and is drawn independently for each bit of the PUF response, then the PUF noise rate p_e is also called the PUF bit error rate. In other words, the noise vector between two PUF measurements is a vector of m independent Bernoulli distributed random variables with probability p_e. This is commonly assumed in the PUF literature [7,1,9].

Katzenbeisser et al. [9] show that the average PUF bit error ranges from 2% to 30% for various silicon PUF constructions. As we discussed earlier, many applications of PUF require static PUF outputs. It is expensive to use general constructions of error correction or fuzzy extractors to reduce the PUF noise. We define the following "pre-processing" process to reduce PUF noise. The pre-processing process has two procedures: *setup* and *processing*. Let f be a physically unclonable function with parameters (n, m, p_e). The optional setup procedure outputs a helper data h, given f and a challenge x as input. The processing procedure outputs a pre-processed PUF response w, given f, x, and h as input.

Setup Procedure. On input of a PUF f and an n-bit challenge x, this procedure outputs a helper data h.

Processing Procedure. On input of a PUF f, a challenge x, and a helper data h, this procedure outputs pre-processed PUF response w.

In the setup or processing procedure, PUF may be evaluated once or multiple times. The goal of the pre-processing function is to reduce the noise in the "pre-processed" PUF responses for any given PUF f and challenge x, i.e., to reduce Hamming distance between two processed responses w and w'.

For example, Temporal Majority Voting (TMV) can be defined as follows. Let k be a small odd number and $t = (k+1)/2$ be the majority voting threshold. The TMV process has no setup procedure and has the following processing procedure.

TMV Processing. On input of a PUF f and a challenge x, f is evaluated k times and k responses obtained $w^{(1)}, \ldots, w^{(k)}$. Let $w_i^{(j)}$ be the i-th bit of $w^{(j)}$. This procedure outputs $w = w_1 \cdots w_m$, where $w_i = 1$ if $w_i^{(1)} + \cdots + w_i^{(k)} \geq t$ and $w_i = 0$ otherwise, for $i = 1, \ldots, m$.

Similarly the dark bits method can be defined as follows. Let k be a parameter for identifying dark bits. The idea of this method is to filter the noisy PUF bits that were observed during the setup procedure. These noisy PUF bits are recorded as a dark bits mask.

Dark Bits Setup. On input of a PUF f and a challenge x, f is evaluated k times and k responses obtained $w^{(1)}, \ldots, w^{(k)}$. Compute helper data $h = (w^{(1)} \oplus w^{(2)}) \vee \cdots \vee (w^{(1)} \oplus w^{(k)})$, where $\oplus$ is bitwise XOR and $\vee$ is bitwise OR. This h is called the dark bits mask.

Dark Bits Processing. On input of the PUF f, challenge x, and dark bits mask h. The processing procedure first evaluates f and obtains $\tilde{w}$, then discards the all bits marked in h from $\tilde{w}$, and produces a shorter output w.

2.2 Our SMV Scheme

In TMV, the PUF is measured multiple times and majority voting is applied on each bit location to filter the PUF noise. Inspired by TMV, we propose SMV in this section. The idea of SMV is to group a few PUF bits together to produce a more stable bit. A naïve method is to perform majority voting on the group, i.e., if there is more '1' in the group, set the group bit as 1, otherwise, set the group bit as 0. Unfortunately, this approach increases the PUF error rate as shown in Figure 1(right) instead of reducing the error rate. The analysis of the naïve SMV method is given in Appendix A.

The basic idea of our SMV scheme is that we divide PUF responses into groups and extract a stable bit from each group as follows: choose a majority subgroup from each group during the setup procedure, and then do majority voting on the subgroup. In the processing procedure, only the bits in subgroup are used for majority voting. Observe that all the bits in the subgroup have the same value in the setup procedure. If there is a small noise in the PUF response, e.g., only a small number of bits in the group flip, the majority voting would filter the noise.

We now formally describe the basic SMV method with the following parameters: k as the group size, k' as the subgroup size, and t as the voting threshold value. For our basic SMV scheme, we set $k = 2k' - 1$ and $k' = 2t + 1$. For example, $k = 9$, $k' = 5$, and $t = 2$.

SMV Setup. On input of a PUF f with parameters (n, m, p_e) and an n-bit challenge x, it runs the following steps:

1. Evaluate f on challenge x and obtains an m-bit PUF response w.
2. Divide w into l groups $G_1, \ldots, G_l$ of size k, where $l = \lfloor m/k \rfloor$.
3. For each group G_i, where $i = 1, \ldots, l$, do the following steps:
   (a) Let group bit b_i be the majority bit of the group G_i.
   (b) Let h_i be a k-bit mask that marks the majority subgroup, i.e., the location of first k' bits of b_i in G_i.
4. This function outputs the helper data $h = h_1, \ldots, h_l$.

SMV Processing. On input of the PUF f, challenge x, and helper data $h = h_1, \ldots, h_l$, it runs the following steps:

1. Evaluate f and obtains PUF response w.
2. Divide w into l groups $G_1, \ldots, G_l$ of size k, where $l = \lfloor m/k \rfloor$.
3. For each group G_i, where $i = 1, \ldots, l$, do the following steps:
   (a) Set t_i as the number of '1' in the subgroup of G_i marked by h_i.
   (b) Set the group bit $b_i = 1$ if $t_i > t$ and set $b_i = 0$ otherwise.
4. Output $b_1, \ldots, b_l$ as processed PUF response.

For example, if the PUF bits in a group G_i are read as $\{0, 0, 1, 0, 1\}$ during the setup procedure, the majority subgroup is the 1st, 2nd, and 4th bits of the group, and h_i is output as $\{1, 1, 0, 1, 0\}$. If PUF bits are read as $\{0, 1, 1, 0, 1\}$ in the processing procedure, the majority voting is conducted on $\{0, 1, *, 0, *\}$, where $*$ denotes the bits outside the voting subgroup. As a result, the group bit b_i is evaluated as 0.

We now calculate how SMV would reduce the PUF error rate. The raw PUF error rate is p_e. Observe that all the k' bits in the subgroup are the same during the setup procedure. The group bit in the setup procedure is the majority bit in the group. In the processing procedure, the group bit changes only if more than t bits in the subgroup flip their values during the PUF evaluation. Let us define binocdf as a binomial cumulative distribution function:

$$\text{binocdf}(t, n, p) = \sum_{i=0}^{t} \binom{n}{i} p^i (1-p)^{n-i}.$$

The error rate after SMV is then $1 - \text{binocdf}(t, k', p_e)$. We plot the error rate on the PUF response after SMV in Figure 1(left) and list a few error rate values in Table 1.

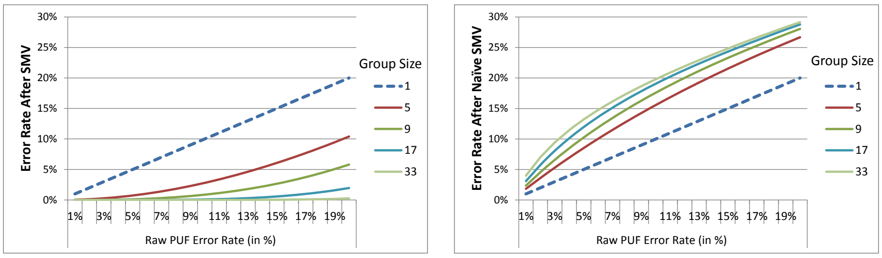

Fig. 1. Error rate reduction using our SMV (left) and naïve SMV (right)

Table 1. Estimated error rate after SMV with $k = 9, 13, 17$, respectively

Raw error rate	.01	.03	.05	.07	0.09	.11	.13	.15	.17
$k = 9$	.000010	.00026	.0012	.0031	.0063	.011	.018	.027	.037
$k = 13$	.000000	.000026	.00019	.00071	.0018	.0039	.0072	.012	.019
$k = 17$	.000000	.000003	.000033	.00017	.00055	.0014	.0030	.0056	.0098

Clearly, SMV is very effective even for a small group size, e.g., $k = 5$ or 9. It can quickly reduce the PUF error rate. Take $k = 9$ as example, if the raw PUF error rate is 15%, the error rate after SMV becomes 2.7% which is a 5x improvement. If the raw error rate is 5%, the error rate after SMV drops to 0.12% which is more than 40x improvement. As we shall see in Section 3, our theoretical analysis matches the SMV performance on real PUF data closely.

3 Experimental Result

We evaluate the performance of our SMV scheme using data collected from a silicon PUF implementation in 65 nm CMOS [15]. The dataset comprises 280 SRAM PUF measurements obtained across 96 device instances over a range of voltage supply and ambient temperature conditions. Table 2 tabulates the number of measurements collected at each operating condition.

Table 2. Number of test measurements recorded per device operating condition

	V=1.08V	V=1.2V	V=1.32V
T=25^{o}C	30	60	30
T=+85^{o}C	20	40	20
T=-40^{o}C	20	40	20

The metrics of inter-distance and intra-distance are used as a measure of PUF performance both of the raw SRAM PUF and our SMV scheme. *Intra-distance* is a measure of the hamming distance between PUF responses taken from the same physical PUF instance. This metric is a measure of the PUF noise rate and should be as close as possible to zero. In the rest of this paper, we use the intra-distance μ_{intra} to refer the average PUF error rate in the PUF experimental data. *Inter-distance* measures the Hamming distance between two PUF responses taken from different physical PUF instances. The inter-distance metric is a measure of the uniqueness of a PUF response and ideally should be 50%.

Inter-distance and intra-distance results for the raw SRAM PUF on 96 devices using the operating conditions described above are shown in Figure 2. We chose an SRAM PUF size of 8704 bits, i.e., $m = 8704$. The results show near-ideal behavior with a mean fractional intra-distance of 6.5% and standard deviation of 0.010. The fractional intra-distance is 49.5% with a standard deviation of 0.006.

With the baseline performance of the raw SRAM PUF determined we now evaluate the performance of our SMV scheme for parameters of $l = 512$ and $k = 9, 13, 17$. Table 3 tabulates the results. In addition to the mean and standard deviation results for inter- and intra-distance we give the maximum intra-distance and mean bias in terms of ones in the SMV output. The intra-distance results show that SMV gives a significant reduction over the raw PUF noise rate, reducing the raw error rate from 6.5% to 0.31%, 0.09%, and 0.02%, respectively, with $k = 9$, 13, and 17. This aligns well with our theoretical analysis in Table 1, where if the raw PUF error rate is 7%, the error rate after SMV improves to 0.31%, 0.071%, and 0.017%, respectively, with $k = 9$, 13, and 17. Figure 3 shows the inter- and intra-distance results for the $k = 13$ and $k = 17$ case.

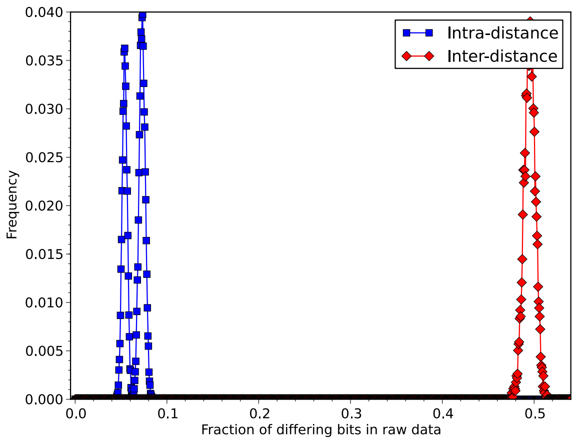

Fig. 2. Inter- vs intra-distances for raw SRAM PUF data (8704 bits)

Table 3. Experimental results before (raw PUF) and after SMV processing

	μ_{intra}	σ_{intra}	max_{intra}	μ_{inter}	σ_{inter}	$\mu_{bias(ones)}$
Raw PUF	.065119	.009900	.085133	.495468	.006009	.502098
SMV, k=9	.003080	.002653	.019531	.484900	.023696	.506249
SMV, k=13	.000852	.001350	.011718	.483650	.023797	.504457
SMV, k=17	.000241	.000691	.007812	.485729	.023695	.507814

4 Alternative SMV Scheme for Biased PUF Responses

Note that if the raw PUF responses are biased, then the bias will be amplified after SMV processing. For example, if the PUF response is biased toward 1, then it is more likely that the majority bit from the group has a similar bias. Let p_1 be the probability that a PUF bit response is 1. For our basic SMV scheme with parameters k, k', t,

$$\begin{aligned}\Pr\left[\text{group bit} = 1\right] &= \Pr\left[\text{number of `1' bits in the group} \geq k'\right] \\ &= 1 - \text{binocdf}(k' - 1, k, p_1)\end{aligned}$$

Figure 4(left) shows that the bias rate quickly increases to 1 or decreases to 0 if the raw PUF bit is biased. This property makes our basic SMV scheme unusable for highly-biased PUFs, in particular for a large group size k. Studies have shown that [9] some PUF constructions (e.g., SRAM PUFs) are unbiased while other PUF constructions (e.g., Latch PUFs or Flip-Flop PUFs) exhibit a strong bias.

In this section, we provide an alternative SMV scheme for PUFs, such that SMV does not exacerbate an existing bias. The basic idea is as follows: let k be the group size, k' be the subgroup size, and t be the voting threshold value, where $k' = 2t + 1$. Instead of choosing $k = 2k' - 1$, choose a larger k, e.g., $k = 3k'$.

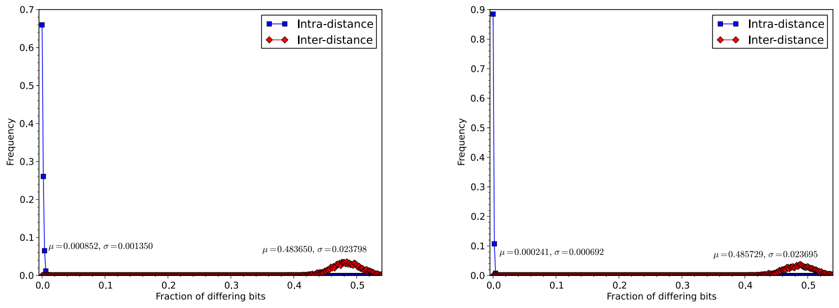

Fig. 3. Inter- vs intra-distances for SMV, k=13 (left) and k=17 (right)

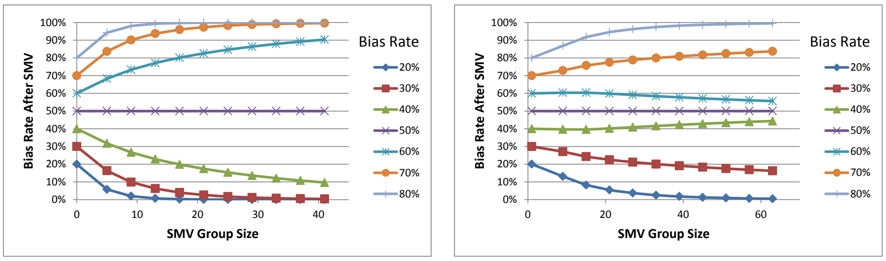

Fig. 4. Bias rate after basic SMV (left) and our alternative SMV (right)

During the setup procedure, for a group G_i, instead of setting the group bit as the majority bit, choose a random bit b_i as the group bit and choose the subgroup based on b_i. For cases where there are not enough b_i in the group, i.e., the number of b_i bits in the group is less than k', then we have to choose the opposite bit as the group bit. The processing procedure in the alternative SMV method is the same as the basic SMV method, i.e., conduct a majority voting on the subgroup. The alternative SMV setup procedure is as follows:

Alternative SMV Setup. On input of a PUF f and an n-bit challenge x, it runs the following steps:

1. Evaluate f on challenge x and obtains an m-bit PUF response w.
2. Divide w into l groups $G_1, \ldots, G_l$ of size k, where $l = \lfloor m/k \rfloor$.
3. For each group G_i, where $i = 1, \ldots, l$, do the following steps:
   (a) Choose a random bit b_i.
   (b) If the number of b_i in G_i is less than k', set $b_i = 1 - b_i$.
   (c) Randomly choose k' number of b_i from G_i as the subgroup and use h_i to mask the subgroup location.
4. This function outputs the helper data $h = h_1, \ldots, h_l$.

In both basic SMV and alternative SMV, the majority voting is conducted on the subgroup. Therefore, the noise rate reduction depends on the subgroup size

k'. For the same k' value, the alternative SMV scheme requires a larger group size k, thus it is less efficient than the basic SMV. We now analyze the bias of the group bit. We use b to denote the group bit and b' to denote the random bit chosen in step 3(a) above. Let t_1 be the number of '1' bits in the group and t_0 be the number of '0' bits in the group.

$$\begin{aligned}\Pr[b=1] &= \Pr[b=1 \vee b'=1]\cdot\Pr[b'=1] + \Pr[b=1 \vee b'=0]\cdot\Pr[b'=0]\\ &= 0.5\cdot\Pr[t_1 \geq k'] + 0.5\cdot\Pr[t_0 < k']\\ &= 0.5(1-\mathrm{binocdf}(k'-1,k,p_1)) + 0.5(1-\mathrm{binocdf}(k-k',k,p_1))\end{aligned}$$

Figure 4(right) plots the bias rate of the group bit with respect to different bias rates in the raw PUF response and different group sizes. We use $k = 3k'$ and $k' = 3, 5, \ldots, 21$. Observe that if the bias rate in the raw PUF response is 60%, the bias rate after the alternative SMV decreases. However, if the raw PUF response is very biased, the bias rate in the group bit increases. This shows that the alternative SMV scheme is fairly effective for slightly biased PUF but less effective for highly biased PUF.

5 Combining SMV with the Dark Bits Method

We observe from Figure 1(left) that for a given group size SMV is more efficient at low raw error rates. In other words, SMV achieves a super-linear rather than linear error-rate reduction. Taking $k = 9$ as an example, Table 1 shows that if the raw PUF error rate is 15%, the error rate after SMV becomes 2.7% which is a 5x improvement. If the raw error rate is 5%, the error rate after SMV drops to 0.12% which is more than 40x improvement. The reduction ratio continues to increase with decreasing raw error rate. The same trend is observed analytically for other group sizes.

In order to exploit the efficiency of SMV at low raw error rates, we propose to combine SMV with other techniques, for example using a dark bit scheme. The dark bit method is a way to remove unstable PUF bits. More specifically, we run multiple PUF measurements during the setup procedure. If a PUF bit is noisy, we mark it as a dark bit and exclude it from further processing. The resultant dark bit mask is stored externally as part of the helper data. We apply the dark bit technique first and reduce the raw error rate to a reasonably low level. SMV is subsequently applied to further reduce the overall error rate.

A formal description of the dark bit augmented SMV method follows. It has the following two parameters o as the number of PUF measurements in the setup procedure and k as the SMV group size. We use DarkSMV(o, k) to denote this method with parameters o and k.

DarkSMV Setup. On input of a PUF f and a challenge x, f is obtained

1. f is evaluated o times and o responses obtained $w^{(1)}, \ldots, w^{(o)}$. Compute dark-bit mask mask $= (w^{(1)} \oplus w^{(2)}) \vee \cdots \vee (w^{(1)} \oplus w^{(o)})$.
2. Divide $w^{(1)}$ into l groups $G_1, \ldots, G_l$ of size k, where $l = \lfloor m/k \rfloor$.

3. Divide mask into l groups of masks $\text{mask}_1, \ldots, \text{mask}_l$ of size k.
4. For each group G_i, where $i = 1, \ldots, l$, do the following steps:
   (a) Discard the dark bits in G_i using the dark bits mask mask_i.
   (b) Let group bit b_i be the majority bit of the group G_i, after the dark bits have been discarded. If there is a tie, choose b_i randomly.
   (c) Let h_i be a k-bit mask that marks the majority subgroup, i.e., the location of bits with value b_i in G_i without the dark bits.
5. This function outputs the helper data $h = h_1, \ldots, h_l$.

DarkSMV Processing. On input if the PUF f, challenge x, and and helper data $h = h_1, \ldots, h_l$.

1. Evaluate f and obtains PUF response w.
2. Divide w into l groups $G_1, \ldots, G_l$ of size k, where $l = \lfloor m/k \rfloor$.
3. For each group G_i, where $i = 1, \ldots, l$, do the following steps:
   (a) Set t_i as the number of '1' in the subgroup of G_i marked by h_i.
   (b) Set u_i as the size of subgroup h_i, i.e., number of '1' in h_i.
   (c) Set the group bit $b_i = 1$ if $t_i > u_i/2$ and set $b_i = 0$ if $t_i < u_i/2$. If $t_i = u_i/2$, set t_i randomly because of voting tie.
4. Output $b_1, \ldots, b_l$ as processed PUF response.

Note that in the DarkSMV method above, the size of the voting subgroup is not fixed. It depends on the number of dark bits in the group. Since the size of the subgroup may not be an odd number, we may need to break the voting tie using a random value, as in step 3(c) above.

We now evaluate the performance of our DarkSMV method using the dataset introduced in Section 3. For the dark-bit mask generation we choose 20 dark-bit reference measurements from the T=25^{o}C, V=1.2V operating condition. Using the dark bit technique in isolation results in an mean intra-distance or error rate 0.75%, a reduction of more than 8x over the raw PUF error rate of 6.5%. The results of combining SMV with the dark bit technique are shown in Table 4 for DarkSMV parameters of $n = 512$ and $k = 9, 13, 17$.

Table 4. Experimental results for DarkSMV(20, k)

	μ_{intra}	σ_{intra}	max_{intra}	μ_{inter}	σ_{inter}	$\mu_{bias(ones)}$
k=9	.000202	.000652	.007812	.483790	.023894	.503783
k=13	.000028	.000242	.003906	.481679	.024121	.504234
k=17	.000012	.000179	.003906	.483748	.024117	.509123

The results show that DarkSMV results in significant further reductions in error rate over that obtained from dark bits in isolation. The mean intra-distance μ_{intra} shows reductions of over 90% while the max intra-distance max_{intra} is reduced to the sub-bit level even for $k = 9$. Figure 5 shows the inter- and intra-distance results for the $k = 13$ and $k = 17$ case.

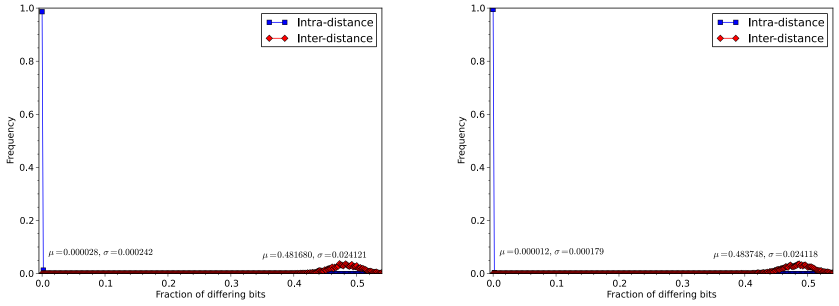

Fig. 5. Inter- vs intra-distances for DarkSMV, k=13 (left) and k=17 (right)

6 Applications of SMV

As we have shown in the last few sections, SMV can significantly reduce the PUF error rate. In this section, we discuss how low PUF error rates can benefit specific PUF applications. We use PUF-based key generation as a representative PUF application. According to [21,7,18], PUF-based key generation eliminates the need to store keys in the clear in non-volatile memory technologies such as fuses or flash. They are thus attractive in applications where high-assurance design techniques such tamper-resistant hardware are too expensive to deploy due to cost or form-factor considerations. Let $\mathcal{C}[n, k, d]$ be an error correcting code, where n is the length of the code, k is the size of message, and d is the minimum distance of the code. In order to extract a 128-bit key from the PUF, Bösch et al. suggested some fuzzy extractor constructions to extract 171-bit entropy from SRAM PUFs [3]. We list a few parameters from [3] in Table 5. We need approximately 4000 bits of PUF in order to extract 128 bits, assuming that PUF raw error rate is 15%. The failure probability represents the expected failure rate of the overall PUF-based key generation scheme.

Table 5. Parameters to extract a 128-bit cryptographic key for SRAM PUFs [3]

Error Correction Codes	PUF Size	Failure Probability
BCH$[1020, 43, 439]$	4080	1.44E-8
RM$[32, 6, 16]$ & Rep$[5, 1, 5]$	4640	1.49E-6
Golay$[24, 13, 7]$ & Rep$[11, 1, 11]$	3696	5.41E-7

Using the combined SMV and dark bits method described in Section 5, we can reduce the error rate significantly, thus reducing the need for expensive error correction codes. For example, using the result from Table 4, the error rate is reduced to 0.0002, 0.000028, and 0.000012, respectively, with $k = 9$, 13, and 17.

Table 6 shows the PUF and error correction parameters required to extract a 128-bit cryptographic key from an SRAM PUF. Note that, using the SMV group size $k = 9$, we only need a 2349-bit PUF and a simple BCH code BCH$[29, 19, 5]$ to correct up to 2 bit errors per 29 bits. Using the SMV group size $k = 13$, we can use a simple Hamming code Hamming$[7, 4, 3]$ as the error correction scheme, due to the low error rate. Implementing such Hamming code is almost free in hardware.

Table 6. Parameters to extract a 128-bit cryptographic key for SRAM PUFs using SMV and dark bits method

Method	Error Correction Codes	PUF Size	Failure Probability
DarkSMV(20, 9)	BCH$[29, 19, 5]$	2349	2.62E-7
DarkSMV(20, 13)	Hamming$[7, 4, 3]$	3913	7.08E-7
DarkSMV(20, 17)	Hamming$[63, 57, 3]$	3213	8.43E-7

Based on these results we believe that SMV is a valuable PUF pre-processing technique in situations where non-volatile storage is cheap (and may be off-chip) while the area cost for logic is expensive. In these scenarios it is beneficial to reduce the cost of ECC to a minimum.

7 Conclusion

In this paper we introduce a new PUF pre-processing technique we term Spatial Majority Voting (SMV). We show analytically SMV if effective in reducing the raw error rate of SRAM PUF responses to low levels and empirically confirm these results using test data obtained from a 65 nm SRAM PUF characterization vehicle. An alternative SMV scheme is introduced for biased PUF responses and shown to be effective for bias rates of up to 60%. Finally, we combine SMV with the dark bits method and show empirically that this approach is capable of reducing the PUF error rate to the sub-bit level where the required ECC complexity is significantly lowered, resulting in a reduced system cost for applications where minimizing logic-cost is the primary design constraint.

Acknowledgement. We thank Intrinsic-ID and EU FP7 programme UNIQUE for providing the SRAM PUF data which enabled the evaluation of our scheme. We thank the anonymous reviewers for providing helpful comments to this paper.

References

1. Armknecht, F., Maes, R., Sadeghi, A.-R., Sunar, B., Tuyls, P.: Memory leakage-resilient encryption based on physically unclonable functions. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 685–702. Springer, Heidelberg (2009)

2. Böhm, C., Hofer, M., Pribyl, W.: A microcontroller SRAM-PUF. In: 5th International Conference on Network and System Security, pp. 269–273. IEEE (2011)
3. Bösch, C., Guajardo, J., Sadeghi, A.-R., Shokrollahi, J., Tuyls, P.: Efficient helper data key extractor on FPGAs. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 181–197. Springer, Heidelberg (2008)
4. Brzuska, C., Fischlin, M., Schröder, H., Katzenbeisser, S.: Physically uncloneable functions in the universal composition framework. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 51–70. Springer, Heidelberg (2011)
5. Dodis, Y., Reyzin, L., Smith, A.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 523–540. Springer, Heidelberg (2004)
6. Gassend, B., Clarke, D., van Dijk, M., Devadas, S.: Silicon physical random functions. In: ACM Conference on Computer and Communications Security, pp. 148–160. ACM Press, New York (2002)
7. Guajardo, J., Kumar, S.S., Schrijen, G.-J., Tuyls, P.: FPGA Intrinsic PUFs and Their Use for IP Protection. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 63–80. Springer, Heidelberg (2007)
8. Juels, A., Wattenberg, M.: A fuzzy commitment scheme. In: ACM Conference on Computer and Communications Security (CCS), pp. 28–36. ACM (1999)
9. Katzenbeisser, S., Kocabaş, Ü., Rožić, V., Sadeghi, A.-R., Verbauwhede, I., Wachsmann, C.: PUFs: Myth, fact or busted? A security evaluation of physically unclonable functions (PUFs) cast in silicon. In: Prouff, E., Schaumont, P. (eds.) CHES 2012. LNCS, vol. 7428, pp. 283–301. Springer, Heidelberg (2012)
10. Koeberl, P., Li, J., Rajan, A., Vishik, C., Wu, W.: A practical device authentication scheme using SRAM PUFS. In: McCune, J.M., Balacheff, B., Perrig, A., Sadeghi, A.-R., Sasse, A., Beres, Y. (eds.) TRUST 2011. LNCS, vol. 6740, pp. 63–77. Springer, Heidelberg (2011)
11. Kumar, S.S., Guajardo, J., Maes, R., Schrijen, G.J., Tuyls, P.: The butterfly PUF: Protecting IP on every FPGA. In: IEEE International Workshop on Hardware-Oriented Security and Trust (HOST), pp. 67–70 (June 2008)
12. Lee, J.W., Lim, D., Gassend, B., Edward Suh, G., van Dijk, M., Devadas, S.: A technique to build a secret key in integrated circuits for identification and authentication application. In: Proceedings of the Symposium on VLSI Circuits, pp. 176–179 (2004)
13. van der Leest, V., Preneel, B., van der Sluis, E.: Soft decision error correction for compact memory-based PUFS using a single enrollment. In: Prouff, E., Schaumont, P. (eds.) CHES 2012. LNCS, vol. 7428, pp. 268–282. Springer, Heidelberg (2012)
14. Lim, D., Lee, J.W., Gassend, B., Edward Suh, G., van Dijk, M., Devadas, S.: Extracting secret keys from integrated circuits. IEEE Transactions on Very Large Scale Integration (VLSI) Systems 13(10), 1200–1205 (2005)
15. Maes, R., Rozic, V., Verbauwhede, I., Koeberl, P., van der Sluis, E., van der Leest, V.: Experimental evaluation of physically unclonable functions in 65 nm cmos. In: 2012 Proceedings of the ESSCIRC (ESSCIRC), pp. 486–489 (September 2012)
16. Maes, R., Van Herrewege, A., Verbauwhede, I.: PUFKY: A fully functional PUF-based cryptographic key generator. In: Prouff, E., Schaumont, P. (eds.) CHES 2012. LNCS, vol. 7428, pp. 302–319. Springer, Heidelberg (2012)
17. Maes, R., Tuyls, P., Verbauwhede, I.: Intrinsic PUFs from flip-flops on reconfigurable devices. In: 3rd Benelux Workshop on Information and System Security (WISSec 2008), Eindhoven, NL, p. 17 (2008)

18. Maes, R., Tuyls, P., Verbauwhede, I.: Low-overhead implementation of a soft decision helper data algorithm for SRAM PUFs. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 332–347. Springer, Heidelberg (2009)
19. Pappu, R.S.: Physical one-way functions. PhD thesis, Massachusetts Institute of Technology (March 2001)
20. Simons, P., van der Sluis, E., van der Leest, V.: Buskeeper PUFs, a promising alternative to D Flip-Flop PUFs. In: 2012 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), pp. 7–12 (June 2012)
21. Simpson, E., Schaumont, P.: Offline hardware/Software authentication for reconfigurable platforms. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 311–323. Springer, Heidelberg (2006)
22. Edward Suh, G., Devadas, S.: Physical unclonable functions for device authentication and secret key generation. In: Design Automation Conference, pp. 9–14. ACM Press, New York (2007)
23. Yu, M.-D., Devadas, S.: Secure and robust error correction for physical unclonable functions. IEEE Design Test of Computers 27(1), 48–65 (2010)

A Analysis of Naïve SMV

In this section, we analyze the efficiency of the naïve SMV scheme. In naïve SMV, majority voting is performed on all bits in a group: if there are more '1', then '1' is output as the group bit, otherwise '0'. Since the PUF results are random and exhibit little bias, a single flip in the bits with the majority value will overturn the result of majority voting.

Let p be the PUF bit error rate. Let l_0 be the total number of zeros in the group. Let x be the total number of bit flips in the group, among them x_0 is the number of $0 \rightarrow 1$ flips and x_1 is for $1 \rightarrow 0$ flips, where $x_0 + x_1 = x$. The number of zeros in an evaluated result of the group equals to $\ell_0' = \ell_0 - x_0 + x_1$. A majority voting fault is defined as the condition of '0' as majority bit differing before and after the evaluation.

$$\mathrm{Flip}(\ell, \ell_0, x_0, x_1) = ((\ell_0 > \lfloor \ell/2 \rfloor) \oplus (\ell_0 - x_0 + x_1) > \lfloor \ell/2 \rfloor) = \texttt{true})$$

The probability of a group bit error which is the probability of a majority voting flip is defined as:

$$P_{\mathrm{group}} = \sum_{0 \le \ell_0 \le \ell} \sum_{0 \le x \le \ell} \sum_{0 \le x_0 \le x} \mathrm{Flip}(\ell, \ell_0, x_0, x - x_0) \cdot \mathbf{p_1} \cdot \mathbf{p_2} \cdot \mathbf{p_3}$$

Where the three conditional probabilities are defined as following, and P_0 and P_1 are the probability that a bit is equal to 0 and 1 respectively:

1. $\mathbf{p_1}$: the probability of having ℓ_0 0's out of ℓ-bit voting group: $\binom{\ell}{\ell_0} P_0^{\ell_0} P_1^{\ell - \ell_0}$
2. $\mathbf{p_2}$: the probability of having x bit flips in a ℓ-bit group: $\binom{\ell}{x} p^x (1-p)^{\ell - x}$
3. $\mathbf{p_3}$: the probability of having x_0 flips to be $0 \rightarrow 1$ flip: $\binom{\ell_0}{x_0}\binom{\ell-\ell_0}{x-x_0} / \binom{\ell}{x}$

Figure 1 (right) shows the P_{group} under different group sizes and PUF error rate p ranging from 1% to 20%. The PUF evaluation results are evenly distributed, i.e. $P_0 = P_1 = 0.5$. The larger PUF bit error rate leads to a larger group error rate. The blue dashed line illustrates the error rate with group size 1 or no grouping. It's obvious that grouping is bad in terms of error rate, as the grouping size increases the rate increases as well. Due to the small bias of the PUF bit value, the number of '0' and '1' are so close that voting is performed at the borderline, any small disturbance may easily change the voting result.