Sensitized Path PUF: A Lightweight Embedded Physical Unclonable Function

Matthias Sauer Pascal Raiola Linus Feiten Bernd Becker
University of Freiburg
Georges-Köhler-Allee 51
79110 Freiburg i. Br., Germany
{sauerm,raiolap,feiten,becker}@tf.uni-freiburg.de

Ulrich Rührmair
Ruhr University Bochum
Universitätsstrasse 150
44801 Bochum, Germany
ruehrmair@ilo.de

Ilia Polian
University of Passau
Innstrasse 41
94032 Passau, Germany
ilia.polian@uni-passau.de

***Abstract*—Physical unclonable functions (PUFs) can be used for a number of security applications, including secure on-chip generation of secret keys. We introduce an embedded PUF concept called sensitized path PUF (SP-PUF) that is based on extracting entropy out of inherent timing variability of modules already present in the circuit. The new PUF sensitizes paths of nearly identical lengths and generates response bits by racing transitions through different paths against each other. SP-PUF has lower area overhead and higher speed than earlier embedded PUFs and requires no helper data stored in non-volatile memory beyond standard error-correction information for fuzzy extraction. Compared with standalone PUFs, the new solution intrinsically and inseparably intertwines PUF behavior with functional circuitry, thus complicating invasive attacks or simplifying their detection.**

We present a systematic design flow to turn an arbitrary (sufficiently complex) circuit into an SP-PUF. The flow leverages state-of-the-art sensitization algorithms, formal filtering based on statistical analysis, and MaxSAT-based optimization of SP-PUF's area overhead. Experiments show that SP-PUF extracts 256-bit keys with perfect reliability and nearly perfect uniqueness after fuzzy extraction for the majority of standard benchmark circuits.

I. Introduction

Physical unclonable functions (PUFs) [1]–[3] are security primitives that provide "digital fingerprints" of electrical circuits or other physical objects. Formally, a PUF P is described by a function $R = P(C, D)$, where D represents the *internal disorder* of a specific object instance, C is a *challenge* that is applied to the object's inputs, and R is the *response* produced by the PUF P upon the application of C [4]. PUFs have numerous applications in secure key storage, key-exchange protocols, digital rights management, or advanced cryptographic schemes like oblivious transfer and bit commitment, making them a broadly usable security tool [2], [3], [5], [6].

While several basic physical phenomena such as optics, magnetic effects, or radio waves have been proposed for PUFs, the PUFs with the currently largest potential for practical applications are based on the electrical properties of electronic circuit elements, such as power-up states of SRAM cells [7], [8], component delays [9] or analog voltage waveforms [10]. One can broadly distinguish between *weak PUFs*, which support only one or very few challenges and are useful for on-chip secret-key generation, and *strong PUFs*, which can process a very large number of challenge-response pairs (CRPs) and can be employed in authentication protocols [5].

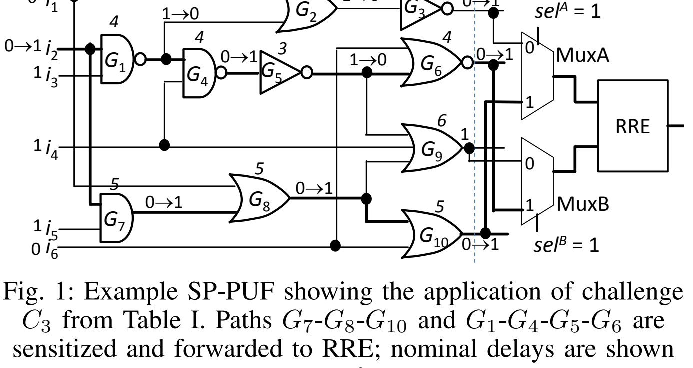

Fig. 1: Example SP-PUF showing the application of challenge C_3 from Table I. Paths G_7-G_8-G_{10} and G_1-G_4-G_5-G_6 are sensitized and forwarded to RRE; nominal delays are shown on top of gates

In this paper, we present a new PUF design called *sensitized-path PUF* (SP-PUF). SP-PUF follows the *embedded PUF* paradigm [11], [12]: it reuses the disorder already present in timing variability of sufficiently complex circuit blocks, thus inseparably intertwining PUF behavior with circuit functionality. Embedded PUFs have several advantages with respect to conventional, standalone PUF modules. First, they can be cost-efficient because they leverage disorder that is already present in the system and do not have to generate entropy themselves. Second, they are more difficult to attack using invasive methods, as the PUF functionality is spread over functional blocks. For instance, it is possible to read-out an Arbiter PUF by photonic-emission analysis [13], but applying the same preparation technique to a large circuit block will likely make it dysfunctional and therefore the attempted attack will be detected. Third, it was proposed to integrate fingerprints of different components on a printed circuit board into a *fusion PUF* [14], and SP-PUF is a natural candidate for complex digital circuits in this scenario.

The principle of SP-PUF is illustrated in Fig. 1. A small circuit is enhanced by two multiplexers MuxA and MuxB and a *race-resolution element RRE* (similar to arbiters used in Arbiter PUFs). Four circuit lines (its primary outputs) are *tapped* and forwarded to the RRE through MuxA and MuxB. A challenge C_i of the SP-PUF consists of three parts: a *test pair* (v_i, w_i) that sensitizes (at least) two paths, and two control signals sel_i^A and sel_i^B which select two taps from which the transitions are forwarded to the RRE. To use the SP-PUF with challenge C_i, sel_i^A and sel_i^B are set, and v_i followed by w_i is applied to the original circuit's inputs. The test pair is generated such as to *simultaneously sensitize* two paths p^A and p^B with roughly identical lengths, i.e., nominal cumulative delays. In a manufactured instance of the circuit, the cumulative delays of both paths p^A and p^B will be determined by fabrication variability. The 1-bit response R_i to challenge C_i is the value produced by the RRE. It will be 1 if p^A is slower than p^B and 0 otherwise.

Table I shows four possible challenges for the SP-PUF of Fig. 1. Applying $C_1, \ldots, C_4$ to a specific circuit instance will result in a four-bit response string $R_1, \ldots, R_4$ which will differ among different circuit instances. For practical application, a larger number of response bits is desired (e.g., at least 128 bits for key generation). Moreover, not all bits may have sufficient quality; for example, C_2 sensitizes path pairs with a rather large nominal delay difference (12 vs. 16), and response bit R_2 will have an undesired bias towards 0. Furthermore, delays are affected by random *environmental fluctuations* which can make the PUF response unstable. This problem also occurs for other types of PUFs and is addressed by *fuzzy extraction* [15].

In this paper, we focus on the applicability of SP-PUF in the key-generation context [16]. In this scenario, the number of CRPs should be sufficient to obtain a useful number of key bits (our experiments target 256 bits after fuzzy extraction). These bits should have very low bias and very low correlation among each other, because otherwise an attacker could predict parts of the secret key. They should have perfect *reliability* (i.e., key generation should be repeatable under environmental fluctuations) and *uniqueness* (average Hamming distance between the response strings of different PUF instances) close to 0.5. The physical realization should withstand invasive attacks (read-out). Note that the conventional usage of a PUF for authentication [17] demands a very large (exponential) number of CRPs and resistance against model-building attacks [4]; we do not investigate these aspects of SP-PUF in this paper.

The specific contributions of this paper are as follows.

- We devise a sophisticated SAT-based method to identify pairs of simultaneously sensitizable paths in a given circuit that are suitable as sources of entropy.
- We introduce a formal mathematical analysis to filter path pairs such as to achieve uniqueness and reliability targets.
- We develop a technique to automatically generate the additional SP-PUF circuitry while minimizing its size.
- We demonstrate the versatility of SP-PUF by applying it to a large number of standard benchmark circuits.

The remainder of the paper is organized as follows. In the next section, we discuss relevant related approaches and the benefits provided by SP-PUF. Section III provides information on SP-PUF construction, that is, turning an arbitrary, sufficiently complex circuit into an SP-PUF. It presents the individual construction steps and describes their role within a larger design flow. Experimental results are reported in Section IV. Section V concludes the paper and provides an outlook.

II. Related Work

There are two existing embedded PUFs based on circuit timing: HELP (hardware-embedded delay PUF) [11] and Glitch PUF [12]. HELP extracts entropy by measuring path delays using an auxiliary on-chip test structure called REBEL and discretized in quanta called PN (for example, 0 PN stands for 5 ns and 128 PN stands for 15 ns in [18]). *Stable paths* (paths with no glitches and nearly identical length when measured multiple times) are identified and this information is stored in on-chip non-volatile memory (NVM) as public helper data.

HELP supports several modes (*UMD, UNMD, DPNC*) to generate responses based on pairs of measured path delays [18], [19]. In UMD, two thresholds with a pre-defined difference ("UMD margin") are used; path lengths above the upper (below the lower) threshold are defined as "strong 1" ("strong 0"). A response bit is generated from two strong paths by XORing their values. In UNMD, the difference of two path delays is computed and compared with two thresholds called +Tr and -Tr. In DPNC, path delays undergo a modulus operation before use to make long and short paths comparable.

TABLE I: Four challenges for SP-PUF in Fig. 1

C_i	(v_i, w_i)	sel_i^A	MuxA	sel_i^B	MuxB
C_1	(001110, 011110)	0	G_1-G_2-G_3	1	G_1-G_4-G_5-G_6
C_2	(001010, 011010)	0	G_1-G_2-G_3	0	G_7-G_8-G_9
C_3	(001110, 011110)	1	G_7-G_8-G_{10}	1	G_1-G_4-G_5-G_6
C_4	(001010, 011010)	1	G_7-G_8-G_{10}	0	G_7-G_8-G_9

The key differences between HELP and SP-PUF are summarized below.

- HELP responses are constructed from comparisons of *sufficiently long* with *sufficiently short* paths. The minimum difference between path delays is enforced in order to obtain reproducible responses. In contrast, SP-PUF compares delays of paths of *similar, ideally identical lengths*, and reproducibility is achieved by fuzzy extraction.
- HELP uses a pseudorandom source (an LFSR) to launch transitions. As a consequence, paths may not be robustly sensitized and invalidations are possible; HELP must identify unstable behavior ("jumps" and "hazards") and suppress the affected paths. In contrast, SP-PUF employs deterministic test pairs which simultaneously sensitize two paths while avoiding hazards and other invalidations.
- HELP stores the information which paths are used for response bit generation in an on-chip NVM during enrollment. An attacker who has access to these helper data and to the nominal gate delays can reproduce the generated key by simulation. The authors of [18] report this vulnerability for the UNMD version; they write that it does not exist for DPNC, but no security analysis is provided. SP-PUF extracts entropy from small variations of nearly equally long paths which cannot be simulated even if the attacker knows the nominal gate delays. It requires only standard helper data for fuzzy extraction and uses them in the same way as SRAM PUFs and other weak PUFs. If new helper data attacks are discovered, all countermeasures will be readily applicable to SP-PUF.
- HELP requires extensive assist circuitry that occupies more than 100% area of the underlying circuit, and it produces 167 key bits per minute [18]. SP-PUF produces one raw response bit per clock cycle (if scan chains are used for challenge application, the number of required cycles is the length of the longest chain).

A further timing-based and hardware-embedded solution is the Glitch PUF [12], which uses the number of glitches in a waveform as the entropy source. It is based on a completely different principle than SP-PUF and is not directly comparable.

III. SP-PUF Construction

As had been indicated in Fig. 1, an existing circuit is turned into an SP-PUF by adding a race-resolution element RRE and two multiplexers MuxA and MuxB. A number of circuit lines are *tapped* and connected to the multiplexers. In this paper, we restrict tap points to circuit outputs, but in general it is also possible to tap internal circuit lines. The number of tap points connected to MuxA (MuxB) is denoted by n_A (n_B). We aim at minimizing n_A and n_B because they determine the size of multiplexers and their logarithm defines the number of multiplexer select inputs sel^A and sel^B; for this reason, we will prefer paths that terminate at a small subset of all circuit outputs.

Fig. 2 shows a typical circuit design flow with SP-PUF-related steps shown in gray. The ultimate target of SP-PUF is to deliver secret key bits, and their number n_{key} is defined on system (or register-transfer) level by the key size used by

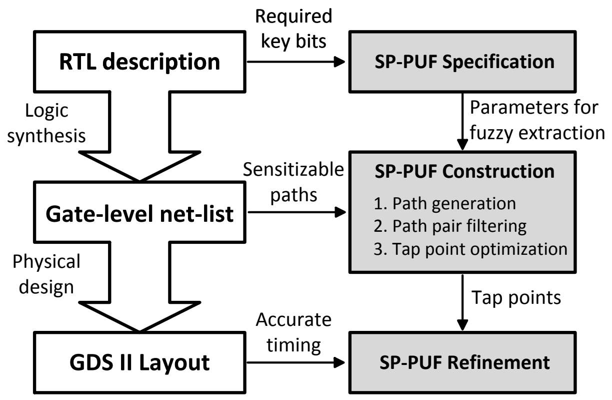

Fig. 2: Steps of SP-PUF construction in design flow

the employed cryptographic algorithm(s) and by the number of different required keys (e.g., for different encryptions or for re-keying). Recall that SP-PUF responses might be noisy and therefore require fuzzy extraction and cryptographic hashing to obtain satisfactory key bits. The parameters of the fuzzy extractor (FE) are set based on the extent of variability (quantified by coefficients of variation c_P and c_E explained in Section III-B) such as to ensure reliable error correction. We employ a *Reed Muller FE* [20], but any other extractor could be used as well. The FE together with the number of required key bits n_{key} determines the required number of SP-PUF's raw response bits n_{target}, i.e., the number of different path pairs that need to be sensitized to obtain n_{key} key bits after fuzzy extraction.

The actual SP-PUF construction is done in three steps that are explained in more detail further below. First, a pool of candidate paths is selected using certain criteria (Section III-A). Out of this pool, pairs of paths (p^A, p^B) are selected such that (1) p^A and p^B are sensitizable by the same test pair and (2) they can provide sufficient uniqueness and reliability. (Section III-B). Finally, the number of required tap points (and therefore the overhead of the scheme) is minimized by choosing path pairs which terminate at a restricted set of circuit outputs (Section III-C).

It is advisable to construct the SP-PUF before physical design, since the required taps can then be considered during routing. This means that only rough gate timing can be used during construction and no effects of parasitic capacitances or low-level optimizations such as gate upsizing are incorporated. Therefore, it is necessary to run a *refinement* step after physical design based on accurate extracted delays of all gates and interconnects. The accurate cumulative delays of all paths in pairs (p^A, p^B) must be calculated, and if they do not match anymore, repair via post-layout engineering change orders (slight speed-up or slow-down of problematic gates by, e.g., up- or downsizing) can be attempted. If this fails, new test pairs can be generated using accurate timing while not modifying the tap points, since they have been routed already. Only if the number of found test pairs is insufficient (less than n_{target}), the complete construction flow has to be repeated and new taps have to be routed.

In the following, we provide more details on the individual steps of SP-PUF construction.

A. Path generation

Given a circuit, we aim at identification of path pairs (p^A, p^B) that are suitable for SP-PUF operation. For this purpose, we first select a pool P of longest paths through the circuit that are *sensitizable*. A path from circuit input i to circuit output o is sensitized by a test pair (v, w) if the application of v followed by w launches a transition on input i and propagates it through all gates on the path, such that it can be observed on o. We use the SAT-based engine PHAETON [21] to determine longest paths that are sensitizable according to the strongest known condition, namely *hazard-free robust sensitization*, to rule out the possibility of invalidations. For this purpose, PHAETON searches for longest paths such that all side-inputs of on-path gates can be simultaneously set to stable non-controlling values, i.e., 0 for (N)OR and 1 for (N)AND gates.

B. Path pair filtering

To use a *pair* of paths (p^A, p^B) in the SP-PUF context, it is necessary to find (v, w) that sensitizes both paths, i.e., sets all side-inputs of gates from p^A and p^B to stable non-controlling values [22]. After the pool of paths P has been constructed, we identify all pairs of paths (p^A, p^B) from P that satisfy the following four conditions. First, p^A and p^B must be simultaneously sensitizable by the same test pair. Second, the lengths of p^A and p^B must be "roughly the same", i.e., they are allowed to differ by at most a user-defined amount Δ_L. This condition keeps undesired bias in check. Finally, the overlap between p^A and p^B must be restricted; for this purpose, at least Δ_G gates in p^A must not be part of p^B, and vice versa (Δ_G is also user-defined). The rationale behind this condition is to avoid pairs (p^A, p^B) which are affected by process variations in largely the same way because they share almost all gates.

The three mentioned conditions have been expressed using Boolean formulae and computed using PHAETON. The fourth condition used for filtering focuses on the quality of a path pair in terms of PUF-specific metrics *uniqueness* (i.e. inter-chip Hamming distance and bit-aliasing) and *reliability* (*rel*). We compute, for each considered path pair, its bit-aliasing and its reliability, calculate a weighted *score* $Rel - |0.5 - BA|$ based on these estimates and retain only path pairs with score exceeding a threshold (determined heuristically as the median score of 5,000 randomly chosen path pairs from P having similar lengths).

Note that a good bit-aliasing was shown to also imply a good inter-chip Hamming distance [23] and hence we refrain from employing an additional uniqueness filtering step.

The set of path pairs retained after filtering is called PP_{filt}. In the remainder of this section, we discuss the underlying delay model and outline the filtering procedures. We had to omit formal derivation of Eqs. 5 and 6 due to lack of space.

We assume that each gate g in the circuit has a *nominal delay* $\delta_{\text{nom}}(g)$. All our procedures support pin-to-pin delays and can distinguish between rising and falling delays. In this way, it is straightforward to incorporate interconnect delays including all applicable parasitics into gate delays. We assume two variability mechanisms: *process variations*, which modify gate delays in each manufactured instance of a circuit due to, e.g., line-edge roughness or dopant level fluctuations, and *environmental fluctuations* (random noise) due to, e.g., unpredictable voltage drops or local clock jitter, which affect the gate delays every time the circuit is used. We model process variations and environmental fluctuations by Gaussian distributions with standard deviations σ_P and σ_E defined by *coefficients of variation* c_P and c_E, respectively.

Let Z be the circuit model where each gate g has nominal delay $\delta_{\text{nom}}(g)$ (actually, a set of rising and falling pin-to-pin delays, which we omit for brevity). If m circuit instances $Z_1, \ldots, Z_m$ are manufactured, each gate g in instance

Z_i assumes *process-variation affected delay* $\delta_P(g)$, which is drawn from the normal distribution $\mathcal{N}(\delta_{\text{nom}}(g), \sigma_P^2)$ with $\sigma_P = c_P \cdot \delta_{\text{nom}}(g)$. Suppose that circuit Z_i is run q times. Random environmental fluctuation will slightly affect its gate delays during each run. We model this by defining q "virtual circuit instances" $Z_{i,1}, \ldots, Z_{i,q}$. The *environmentally influenced delay* δ_E of gate g in $Z_{i,j}$ is drawn from $\mathcal{N}(\delta_{\text{P}}(g), \sigma_E^2)$ with $\sigma_E = c_E \cdot \delta_{\text{nom}}(g)$. Note that all considered distributions are uncorrelated because the effects of correlated path delay changes (e.g., due to circuit-wide temperature increase or decrease) on SP-PUF cancel each other out. In practice, variations may have correlated and uncorrelated components, and the considered σ_P and σ_E capture the uncorrelated portion.

Recall that an SP-PUF's challenges are n pairs of paths $(p_1^A, p_1^B), \ldots, (p_n^A, p_n^B)$. For each of the m instances Z_i, the *k-th SP-PUF response bit* $R_k^{i,j}$ is 0 if the cumulative delay of path p_k^A is lower than that of p_k^B ($\delta_E(p_k^A) < \delta_E(p_k^B)$) in repeat j, and 1 otherwise.

We evaluate the *uniqueness* of the SP-PUF with the inter-chip Hamming distance metric [24] that is defined as the average Hamming distance between responses of different instances Z_i:

$$Uniq = \frac{2}{m \cdot (m-1)} \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} \frac{HD(R^{i,1}, R^{j,1})}{n}. \tag{1}$$

The best possible uniqueness is 0.5 which indicates a high diversity of responses. Note that we are not using the inter-chip Hamming distance metric in the process of path filtering but report this metric in the analysis of our empirical simulations.

For path filtering, the highly correlated (cf. [23]) *bit aliasing* metric is used. Bit aliasing $BA(p_k^A, p_k^B)$ of bit k is the proportion of 1's among all responses:

$$BA(p_k^A, p_k^B) \coloneqq |\{R_k^{i,1} = 1 \mid 1 \le i \le m\}| / m \tag{2}$$

The best value of $BA(p_k^A, p_k^B)$ is 0.5; it means that the bit was sampled as 0 just as often it was sampled as 1 and indicates absence of any unwanted bias. The bias $Bias(p_k^A, p_k^B)$ of bit k is defined as

$$Bias(p_k^A, p_k^B) \coloneqq |0.5 - BA(p_k^A, p_k^B)| \tag{3}$$

$Bias(p_k^A, p_k^B)$ captures the deviation of $BA(p_k^A, p_k^B)$ from its optimal value. We are therefore targeting 0.0 for this metric which indicates the absence of a bias towards 1 or 0.

The *reliability* $Rel(p_k^A, p_k^B)$ of bit k quantifies the average likelihood over m instances Z_i that the same response will be generated when an instance is run multiple times under environmental fluctuations. It is formalized as the average proportion of consistent bits among "virtual instances" $Z_{i,1}, \ldots, Z_{i,q}$:

$$\begin{aligned} Rel(p_k^A, p_k^B, Z_i) \coloneqq \max\{&|\{R_k^{i,j} = 1 \mid 1 \le j \le q\}| \,/\, q, \\ &|\{R_k^{i,j} = 0 \mid 1 \le j \le q\}| \,/\, q\}. \end{aligned} \tag{4}$$

The best reliability value is 1.0; it describes the case when no fluctuation-related response bit-flips occur. Note that intra-chip HD is also used in the literature to quantify PUF stability [18]; the best value of that metric would be 0.0.

To filter path pairs from PP_{init}, we calculate, for each $(p_k^A, p_k^B) \in PP_{\text{init}}$, BA and Rel assuming the number m of circuit instances and the number q of runs go to ∞:

$$BA(p_k^A, p_k^B) = \Pr(\delta_E(p_k^A) > \delta_E(p_k^B)) \tag{5}$$

$$\begin{aligned} Rel(p_k^A, p_k^B) = \frac{1}{2} \cdot \int_{-\infty}^{\infty} f(t) + f(t) \cdot |1 - 2 \cdot \\ \Pr\big(\delta_E(p_k^A) > \delta_E(p_k^B) \mid \delta_P^-(p_k^A) - \delta_P^-(p_k^B) = t\big)| \,\mathrm{d}t \end{aligned} \tag{6}$$

where f denotes the density function of $\delta_P^-(p_k^A) - \delta_P^-(p_k^B)$ and $\delta_P^- = \delta_P - \delta_{\text{nom}}$.

C. Tap point optimization

If the number of path pairs in PP_{filt} exceeds the required number n_{target} of SP-PUF's raw response bits, we apply *tap point optimization* (TPO) to reduce the size of multiplexers MuxA and MuxB and the number of bits in sel^A and sel^B that are part of SP-PUF's challenge. TPO selects path pairs out of PP_{filt} such that the numbers n_A and n_B of tap points connected to MuxA and MuxB, respectively, are minimized. Note that TPO can significantly reduce the number of usable path pairs, and we are employing it in the key-generation scenario because the overall number of extractable response bits is fixed to n_{target}.

TPO constructs the set PP_{TPO} of selected path pairs and the sets T_A and T_B of tap points for MuxA and MuxB. Let $e(p)$ denote the endpoint of path p. A *valid tap point assignment of size* ν is a combination of PP_{TPO}, T_A and T_B which fulfills the following conditions:

$$\begin{aligned} \forall (p^A, p^B) \in PP_{\text{TPO}} \quad &: \quad \big(e(p^A) \in T_A \wedge e(p^B) \in T_B\big) \vee \\ & \qquad \big(e(p^B) \in E_A \wedge e(p^A) \in E_B\big) \\ |PP_{\text{TPO}}| \quad &\ge \quad n_{\text{target}} \\ n_A = |T_A| \quad &\le \quad \nu \\ n_B = |T_B| \quad &\le \quad \nu \end{aligned}$$

We map the conditions to MaxSAT formulae by constructing a series of problem instances, starting with $\nu = n_{\text{target}}$ (for which the problem is guaranteed to have a straightforward solution). We solve them using software *antom* [25], gradually decreasing ν until the equations become unsatisfiable. The last ν for which the problem was solvable is the minimal possible number of multiplexer inputs; smaller n_A or n_B cannot accommodate n_{target} suitable path pairs.

After the optimization, the multiplexers MuxA and MuxB with n_A and n_B inputs, respectively, are added to circuit Z and connected to the determined taps.

D. Security aspects

SP-PUF's security is defined as the inability of an adversarial attacker to reconstruct (read-out) the generated secret key or replace it by a different key. Knowing the underlying circuits' structure, the tap positions, the nominal gate delays and the challenges (test pairs) is not sufficient to recover the key by simulation, because the compared paths are of nearly identical lengths and the outcome is determined by process variations. Similarly, breaking an Arbiter PUF requires extracting variation-affected delays of its elements by invasive techniques [13] or model building [4].

The security of SP-PUF would be compromised if the attacker knew the variation-affected gate delays in a specific circuit instance. Invasive attacks are much more difficult for the large and complex circuitry used by SP-PUF as entropy source compared to compact Arbiter PUF with regular structure. However, the circuit should still implement state-of-the-art defenses against invasive attacks (e.g., light sensors for tamper-detection). Moreover, the adversary should not be able to reuse infrastructure for manufacturing test to measure input-to-output delays in the circuit, because otherwise gate-level characterization [26] might reveal variation-affected gate delays. This is best achieved by secure test-access solutions [27], [28]. Note that we focus on the key-generation scenario where the SP-PUF

TABLE II: SP-PUF construction to obtain $n_{\text{target}} = 4096$ raw bits for $c_P = 20\%, c_E = 3\%$: # of path pairs after filtering ($|PP_{\text{filt}}|$), # of MuxA/MuxB inputs ($|T|$), # of path pairs after TPO ($|PP_{\text{TPO}}|$), average $Uniq$, BA, $Bias$, Rel as well as the run times

Circuit	Gates	Paths	$\lvert PP_{\text{filt}}\rvert$	Tap point optim.		Validation on random instances				Run time [s]		
				$\lvert T\rvert$	$\lvert PP_{\text{TPO}}\rvert$	Avg. $Uniq$	Avg. BA	Avg. $Bias$	Avg. Rel	Path gen.	Filtering	Valid.
s05378	2156	3883	12288	24	4096	0.5000	0.5003	0.0251	1.0000	0.80	51.46	561.82
s09234	1696	2836	12288	10	4096	0.4991	0.4983	0.0243	1.0000	0.83	54.10	503.19
s13207	5344	2678	10123	130	4096	0.5000	0.5012	0.0247	1.0000	2.60	47.19	842.70
s15850	5614	7140	12288	84	4096	0.5000	0.4996	0.0252	1.0000	48.67	51.37	733.44
s35932	18058	8193	12288	152	4096	0.5000	0.4987	0.0244	1.0000	1.96	36.75	739.64
s38417	16065	12288	12288	25	4096	0.5000	0.5004	0.0260	1.0000	34.28	82.63	1007.22
s38584	16377	9328	12288	218	4096	0.5000	0.5006	0.0254	1.0000	4.55	98.73	1028.92
b14	6895	858	3966	276	3966	0.4999	0.4998	0.0250	1.0000	13.86	78.58	517.53
b15	10538	3679	11755	85	4096	0.5000	0.5001	0.0255	1.0000	1455.10	1104.31	526.91
b17	34069	12288	12288	91	4096	0.5000	0.5004	0.0247	1.0000	3370.83	541.02	1341.46
b18	95692	12288	12288	88	4096	0.5000	0.4995	0.0254	1.0000	8682.70	601.06	2481.92
b20	15607	2773	12288	118	4096	0.5000	0.4996	0.0239	1.0000	59.24	233.35	723.99
b21	15251	2665	12288	128	4096	0.5000	0.5008	0.0253	1.0000	55.84	209.83	908.12
b22	21728	3674	12288	198	4096	0.5001	0.5009	0.0256	1.0000	90.97	239.67	927.03
p35k_s	33273	12288	12288	32	4096	0.5000	0.4986	0.0250	1.0000	4994.52	26872.68	1937.68
p45k_s	37663	12288	12288	26	4096	0.4999	0.5013	0.0246	1.0000	2527.22	231.41	1326.25
p78k_s	86153	12288	12288	98	4096	0.5000	0.4980	0.0248	1.0000	454.38	223.87	6862.17
p81k_s	99927	12288	12288	16	4096	0.4997	0.4979	0.0254	1.0000	1550.06	4792.17	1459.04
p100k_s	82575	12288	12288	32	4096	0.5000	0.4996	0.0245	1.0000	4719.63	331.57	2046.16

output is not observable outside the circuit and cannot be used for model building.

The challenges (test pairs) are identical for all circuits and not considered "helper data", i.e., circuit-specific bits generated during enrollment. They can be stored (publicly) on-chip in a ROM; since the test pairs can include *don't cares*, it is possible to compress them using standard techniques [29]. Note that an adversary who knows the challenges cannot reconstruct the key, so neither the compressed data nor the decompression block need obfuscation. However, the challenges must be protected against being overwritten, because otherwise an adversary could copy a single challenge n_{target} times and replace the SP response by an all-0 or all-1 string. It is possible to store the challenges outside the circuit but only if their integrity is guaranteed. This can be achieved by using secure access mechanism for transmitting the challenges to the circuit, or by authenticating the data either using a digital signature or a cryptographic hash (the public key of the signer or the expected hash value can be stored on-chip in a ROM).

Regardless whether the challenges are stored on-chip, decompressed or transmitted from outside, they are applied either via the inputs of the basic circuit or via its internal scan chains using launch-on-shift or launch-on-capture techniques. If any circuitry for challenge application is connected with infrastructure used for manufacturing test, its adversarial usage must be ruled out by, e.g., requiring authentication [28].

IV. Experimental Results

We applied SP-PUF construction and tap point optimization to sufficiently large (at least 1,000 gates) ISCAS-89, ITC-99 and industrial circuits provided by NXP (the names of these circuits start with "s", "b" and "p", respectively). All circuits have been resynthesized using the 45nm Nangate OpenCell library. We employed a Reed Muller fuzzy extractor (FE) [20] with parameters $R = 1$ and $M = 7$. These settings deliver reliable 256 key bits out of 4096 raw SP-PUF response bits.

Table II summarizes the results for coefficients of variation $c_P = 20\%$ and $c_E = 3\%$. Recall that c_P defines the standard deviation of the delay distribution due to manufacturing process variability that provides SP-PUF's entropy ($\sigma_P = c_P \cdot \delta_{\text{nom}}(g)$) while c_E is its counterpart for modeling environmental fluctuations or random noise that limits SP-PUF's reliability and necessitates fuzzy extraction ($\sigma_E = c_E \cdot \delta_P(g)$).

The first three columns of Table II contain the circuit name, the number of gates and the number of paths (longest paths through each output in the circuit, excluding duplicates and bounded by $3 \cdot n_{\text{target}}$) selected by the method from Section III-A. The number of path pairs that remained after filtering (Section III-B) with parameters $\Delta_L = 2.5\%$ and $\Delta_G = 5$ and bounded by $3 \cdot n_{\text{target}}$ is reported in column $|PP_{\text{filt}}|$. Column $|T|$ shows the minimal number of taps required to retain $n_{\text{target}} = 4096$ path pairs. It can be seen that procedure from Section III-C is successful in bringing the number of required taps far below n_{target}.

We validated the SP-PUF concept by Monte-Carlo simulation on $(1,000 \cdot 250)$ circuit instances $Z_{i,j}$ that were randomly created following the construction principle from Section III-B. First, $m = 1,000$ *process-variation affected instances* Z_i were obtained from the considered circuit by drawing the delay $\delta_P(g)$ of every gate g from the normal distribution $\mathcal{N}(\delta_{\text{nom}}(g), \sigma_P^2)$. Then $q = 250$ runs of circuit Z_i affected by environmental fluctuations were modeled by $Z_{i,1}, \ldots, Z_{i,250}$ with gate delays drawn from $\mathcal{N}(\delta_{\text{P}}(g), \sigma_E^2)$.

For each of the $(1,000 \cdot 250)$ circuits $Z_{i,j}$, we obtained raw SP-PUF bits, applied fuzzy extraction to these bits and extracted the 256-bit key using the cryptographic hash function SHA-3. We repeated generation of raw bits (using identical sets of challenges) three times and applied majority voting before FE, in order to counter rare cases when FE did not fully eliminate bit flips due to environmental noise. In this way, we consistently achieved perfect reliability on all considered instances. We report average uniqueness (Eq. 1), bit-aliasing (Eq. 2), bias (Eq. 3) and reliability (Eq. 4) of the generated keys in columns "Avg. $Uniq$", "Avg. BA", "Avg. $Bias$" and "Avg. Rel", respectively. It can be seen that the uniqueness and the bit-aliasing are both very close to their best possible values of 0.5. The average bias is around 0.025 and hence very low. Note that some small bias over the $1,000$ validation instances is statistically expected. The reliability is 1.0 for each circuit indicating a perfect reliability after FE for each validation instance and repeat.

The average of these values over all circuits in Table II *before* FE is $Rel = 0.9537$, $Uniq = 0.4949$, $BA = 0.4985$ and $Bias = 0.1365$.

We validated the suitability of the generated keys for cryptographic applications by applying statistical tests from the suite *dieharder*, version 3.31.1. We had to concatenate keys generated for 250,000 circuit instances $Z_{1,1}, \dots, Z_{250000,1}$ of some reference circuits to obtain strings of 64,000,000 bits, and all tests applicable to this number of bits were passed.

The final columns of Table II show run-times for path generation, filtering including TPO and validation on $(1,000 \cdot 250)$ instances. The slowest path generation time of around 2 hours is for the 100K gate circuit b18, which is remarkably fast given the complexity to find and simultaneously sensitize pairs of paths using hazard-free robust conditions. Filtering run-times are determined by the numbers of available suitable path pairs and the complexity to simultaneously sensitize two paths.

Table III shows the influence of different settings of process variations c_P and environmental fluctuations c_E on the quality of SP-PUFs based on the circuits used in the previous experiment. The table displays the expected *Bias* and the expected reliability *before* FE and key generation (as opposed to Table II) for $c_P \in \{15\%, 20\%, 25\%\}$ and $c_E \in \{1\%, 3\%, 5\%\}$.

It can be seen that the expected reliability ranges between 0.90 to 0.98 and is highly influenced by the "signal-to-noise" ratio between c_P and c_E. The results suggest that the highest expected noise c_E should be assumed during SP-PUF construction because otherwise the reliability deteriorates substantially (and must be addressed by a more powerful error-correction scheme to achieve perfect reliability after key generation). In contrast, the bias mainly depends on c_P and decreases for larger process variations. Note that after key generation the bias is reduced due to the additional confusion introduced by the cryptographic hash function. Hence, the considered process variability levels introduce sufficient randomness for SP-PUF construction.

V. Conclusions and Future Work

We presented a new approach for constructing PUFs from (almost) arbitrary, sufficiently complex digital circuits, which is based on the inherent timing variability of simultaneously sensitized paths. Our method closely intertwines PUFs and circuits, allowing a new class of PUF designs. It offers security benefits, can be integrated into a system-wide "fusion PUF" and at the same time it is associated with limited area overhead and provides high bit generation speed. We showed that SP-PUF is suitable for reliable key generation when used in combination with a state-of-the-art fuzzy extractor. Its helper data are restricted to the standard information used for error correction and constitute no known security threats.

We also introduced a design flow that incorporates advanced multipath sensitization methods, statistical filtering techniques and MaxSAT optimization. This flow can turn an arbitrary circuit into an SP-PUF with a high degree of automation and minimal interference with the regular circuit design process. Using this flow, we demonstrated that a 256-bit key can be reliably generated for a variety of circuits.

In the future, we plan to investigate the suitability of SP-PUF for challenge-response authentication. For this purpose, a large (exponential) number of challenge-response pairs is needed, but a lower reliability is acceptable, which relaxes the requirements on path sensitization. Moreover, the vulnerability of SP-PUF to modeling attacks is of interest in this context.

Acknowledgements

This research was partially financed by the Baden-Württemberg Stiftung gGmbH Stuttgart within the scope of its IT security research programme. Ulrich Rührmair is supported by ERC Project ERCC (FP7/615074).

TABLE III: Influence of variability on the avg. *Bias* and *Rel*

	$c_E = 1\%$		$c_E = 3\%$		$c_E = 5\%$	
c_P	*Bias*	*Rel*	*Bias*	*Rel*	*Bias*	*Rel*
$c_P = 15\%$	0.1127	0.9800	0.1107	0.9406	0.1069	0.9029
$c_P = 20\%$	0.0853	0.9847	0.0845	0.9543	0.0827	0.9248
$c_P = 25\%$	0.0686	0.9876	0.0682	0.9630	0.0673	0.9389

References

[1] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *DAC*, 2007, pp. 9–14.

[2] R. Maes and I. Verbauwhede, "Physically unclonable functions: A study on the state of the art and future research directions," in *Towards Hardware-Intrinsic Security*. Springer, 2010, pp. 3–37.

[3] U. Rührmair, S. Devadas, and F. Koushanfar, "Security based on physical unclonability and disorder," in *Introduction to Hardware Security and Trust*, M. Tehranipoor and C. Wang, Eds. Springer, 2012, pp. 65–102.

[4] U. Rührmair *et al.*, "Modeling attacks on physical unclonable functions," in *ACM Conf. on Computer and Comm. Security*, 2010, pp. 237–249.

[5] U. Rhrmair and D. E. Holcomb, "Pufs at a glance," in *DATE*, March 2014, pp. 1–6.

[6] R. Maes, *Physically Unclonable Functions - Constructions, Properties and Applications*. Springer, 2013.

[7] J. Guajardo, S. S. Kumar, G. J. Schrijen, and P. Tuyls, "FPGA intrinsic PUFs and their use for IP protection," in *CHES*, 2007, pp. 63–80.

[8] D. E. Holcomb, W. P. Burleson, and K. Fu, "Power-up SRAM state as an identifying fingerprint and source of true random numbers," *IEEE Trans. Computers*, vol. 58, no. 9, pp. 1198–1210, 2009.

[9] J. Li and J. Lach, "At-speed delay characterization for IC authentication and trojan horse detection," in *HOST*, 2008, pp. 8–14.

[10] A. Vijayakumar and S. Kundu, "A novel modeling attack resistant PUF design based on non-linear voltage transfer characteristics," in *Design, Automation & Test in Europe Conf.*, 2015, pp. 653–658.

[11] J. Aarestad, P. Ortiz, D. Acharyya, and J. Plusquellic, "HELP: A hardware-embedded delay PUF," *IEEE Design & Test*, vol. 30, no. 2, pp. 17–25, 2013.

[12] K. Shimizu, D. Suzuki, and T. Kasuya, "Glitch PUF: extracting information from usually unwanted glitches," *IEICE Transactions*, vol. 95-A, no. 1, pp. 223–233, 2012.

[13] S. Tajik *et al.*, "Physical characterization of arbiter PUFs," in *CHES*, 2014, pp. 493–509.

[14] A. Aysu *et al.*, "A design method for remote integrity checking of complex PCBs," in *DATE*, 2016, pp. 1517–1522.

[15] R. Maes, A. V. Herrewege, and I. Verbauwhede, "PUFKY: A fully functional PUF-based cryptographic key generator," in *CHES*, 2012, pp. 302–319.

[16] D. Lim *et al.*, "Extracting secret keys from integrated circuits," *IEEE Trans. VLSI Syst.*, vol. 13, no. 10, pp. 1200–1205, 2005.

[17] R. Pappu, B. Recht, J. Taylor, and N. Gershenfeld, "Physical one-way functions," *Science*, vol. 297, no. 5589, pp. 2026–2030, 2002.

[18] J. Aarestad, J. Plusquellic, and D. Acharyya, "Error-tolerant bit generation techniques for use with a hardware-embedded path delay PUF," in *HOST*, 2013, pp. 151–158.

[19] F. Saqib, M. Areno, J. Aarestad, and J. F. Plusquellic, "ASIC implementation of a hardware-embedded physical unclonable function," *IET Computers & Digital Techniques*, vol. 8, no. 6, pp. 288–299, 2014.

[20] C. Bösch, J. Guajardo, A.-R. Sadeghi, J. Shokrollahi, and P. Tuyls, *Efficient Helper Data Key Extractor on FPGAs*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 181–197.

[21] M. Sauer, B. Becker, and I. Polian, "PHAETON: A SAT-based framework for timing-aware path sensitization," *IEEE Trans. Computers*, vol. 65, no. 6, pp. 1869–1881, 2016.

[22] M. Sauer *et al.*, "Efficient SAT-based dynamic compaction and relaxation for longest sensitizable paths," in *DATE*, 2013, pp. 448–453.

[23] L. Feiten, M. Sauer, and B. Becker, "On metrics to quantify the inter-device uniqueness of PUFs," 2016, https://eprint.iacr.org/2016/320.

[24] A. Maiti, J. Casarona, L. McHale, and P. Schaumont, "A large scale characterization of RO-PUF," in *HOST*, 2010, pp. 94–99.

[25] S. Reimer, M. Sauer, T. Schubert, and B. Becker, "Incremental encoding and solving of cardinality constraints," in *ATVA*, 2014, pp. 297–313.

[26] S. Wei, S. Meguerdichian, and M. Potkonjak, "Gate-level characterization: foundations and hardware security applications," in *DAC*, 2010, pp. 222–227.

[27] B. Yang, K. Wu, and R. Karri, "Secure scan: A design-for-test architecture for crypto chips," *IEEE Trans. on CAD*, vol. 25, no. 10, pp. 2287–2293, 2006.

[28] R. Baranowski, M. A. Kochte, and H. Wunderlich, "Fine-grained access management in reconfigurable scan networks," *IEEE Trans. on CAD*, vol. 34, no. 6, pp. 937–946, 2015.

[29] J. Rajski *et al.*, "Embedded deterministic test," *IEEE Trans. on CAD*, vol. 23, no. 5, pp. 776–792, 2004.