# A Comprehensive Set of Schemes for PUF Response Generation

Bilal Habib and Kris Gaj

Electrical and Computer Engineering Department
George Mason University,
Fairfax VA , USA
{bhabib, kris}@gmu.edu

**Abstract.** In this work we present a comprehensive set of schemes to generate Physical Unclonable Functions (PUF) responses. Software scripts have been developed to generate the PUF IDs using five different schemes. We also propose a new set of PUF metrics that are based on the Worst Case (WC) PUF performance. Values of these metrics, for the Ring Oscillator (RO) PUF and SR-Latch PUF, have been presented and analyzed in the paper.

**Keywords:** FPGA, PUF, Analysis

## 1 Introduction

Extensive research has been going on to develop new hardware primitives for security. PUF is one of such primitives that can efficiently solve several problems in hardware security. These problems range from reverse engineering and counterfeiting to detection of pirated devices. For this purpose the two important applications where PUF can be used are: key generation and device authentication. The input to PUF is called a challenge and the output is called a response. In the past researchers used only one scheme to generate the PUF responses. For more in-depth analysis we need to generate responses using more than one scheme. This approach gives us the flexibility to control the response bit size. We present five most popular schemes in this work. In Section 2, we describe the previous work. Section 3 explains the PUF schemes; methodology is covered in section 4. In Section 5, we discuss the results. The conclusion and future study are described in Section 6.

## 2 Previous Work

The previously reported schemes for PUF response generation from raw data included: comparing the counts of neighboring ROs [1], Lehmer-Gray encoding method [3], Identity-mapping [4] and S-ArbRO method [5]. Quantitative and statistical performance evaluations of Arbiter PUF and RO-PUF were presented in [2] and [6], respectively. In this work we analyze PUF responses generated using five different schemes. Each scheme is described in detail using a pseudocode. For evaluation of responses, PUF metrics have been developed.

## 3 PUF Schemes for ID generation

PUF responses are generated from the raw PUF data as shown below in Fig. 1,

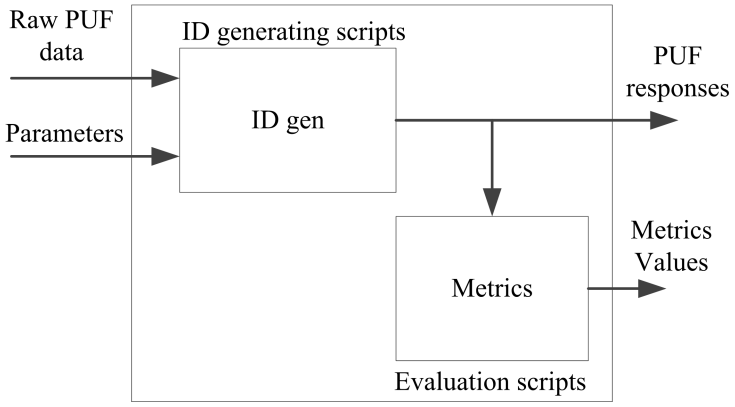


**Fig. 1.** PUF ID generation and evaluation

As shown above in Fig. 1, the raw PUF data is the input to our PUF generation module. In case of RO-PUF raw input consists of the number of ring-oscillator periods within a predefined fixed time interval. Similarly in case of SR-Latch PUF, raw input consists of the number of oscillations of a latch during metastable state. In this work we used three data sets; Spartan-3 data-set for ROs [7], Spartan-6 data-set for SR-Latches [8] and Zynq data-set for SR-Latches. Devices used for Zynq data belongs to (XC7Z010) family. Five schemes are explained below.

**i) Comparing the neighboring components (CNC)** In this scheme raw data from the neighboring components of FPGA are compared. Fig. 2 shows this scheme on the left side. In this figure $C_0$, $C_1$,...,$C_{M-1}$ shows the physical location of components on the FPGA fabric. The comparison of ($C_0$,$C_1$), ($C_1$, $C_2$), ($C_2$, $C_3$),...,($C_{M-2}$,$C_{M-1}$) is carried out. Therefore for M components the total number of PUF response bits will be equal to M-1. Response array stores the PUF response of M components.

**ii) Pairwise Comparison (PC)** In this comparison each component is compared only once with its neighbor. The comparison of ($C_0$, $C_1$), ($C_2$,$C_3$), ($C_4$, $C_5$),...,($C_{M-2}$, $C_{M-1}$) is carried out. Therefore for M components the total number of PUF response bits will be equal to $\lfloor M/2 \rfloor$. Fig. 2 shows this scheme on the right.

**Algorithm 1** Comparing the Neighboring Components

```
CNC( F[], M):     // F[] is the input array of M components.
for(i = 0; i< M–1; i++)
        if(F[i] > F[i+1])
                Response[i] = 1
        else
                Response[i] = 0
        end if
end for
```

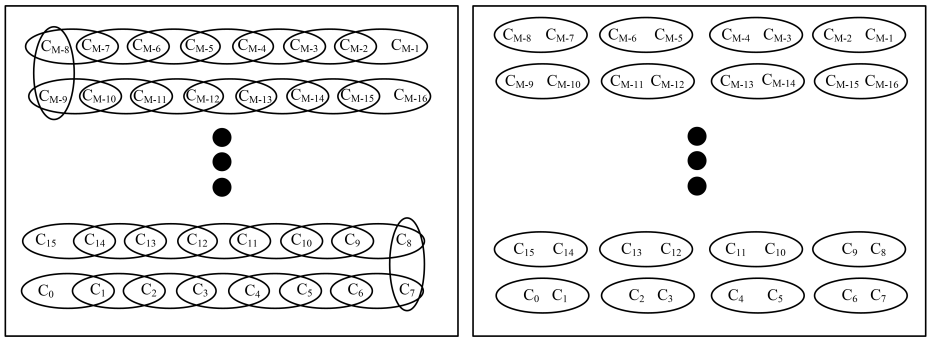


**Fig. 2.** Comparison of neighboring Components (Left). Pairwise comparison with neighbors (Right).

**Algorithm 2** Pairwise Comparison

```
PC(F[], M):    // F[] is the input array of M components
for(i=0; i < 2 * ⌊M/2⌋ ; i+=2)
        if(F[i] > F[i+1])
                Response[i] = 1
        else
                Response[i] = 0
        end if
end for
```

**iii) Binary Lehmar-Gray (BLG) encoding** In LG encoding, all components are divided into sets of size S. Encoding the ordering of S component measurements $F^s$= ($F_0$, . . . , $F_{s-1}$ ) results into an L-bit response. It represents the sorted ordering of F components as a coefficient vector $L^{s-1} = (L_1, \ldots, L_{s-1})$ with $L_i \in [0, 1, ..., i]$. It is clear that $L^{s-1}$ can take $2 * 3 * \ldots * S = S!$ possible values which is exactly the number of possible orderings. The Lehmer coefficients are calculated from F as $L_j = \sum_{i=0}^{j-1} gt(Fj, Fi)$ with gt (x, y) = 1 if x > y and 0 otherwise. The total number of bits generated for each set is: $\sum_{i=2}^{S} \lceil log_2\ i \rceil$.

Below is a pseudocode for converting counts of M components into an $\lfloor M/S \rfloor$ *L -bit response denoted by Response.

---

**Algorithm 3** Binary Lehmer Gray Encoding

---

```
BLG (F[], M, S):   // S=Set size.
for(t=0; t < ⌊M/S⌋  ; t++)
Response= Response || Lehmer(F[t*S:(t+1)*S-1], S)
end for

Lehmer(array[], S):
for(j=1; j< S ; j++)
  sum = 0
  for(i=0; i<j ; i++)
          if (array [j] > array [i])
          sum= sum + 1
          end if
  end for
  L_Response = L_Response || (Gray(bin(sum, ceil(log(j+1)))))
end for
return L_Response

Gray(bin_bits): //array of binary bits is passed to Gray().
Len_bits=len(bin_bits)          //Length of an array
G[0] = bin_bits[0]
for(i=1; i< Len_bits; i++)
        G[i] = XOR(bin_bits [i], bin_bits [i-1])
end for
```

---

Above a function named bin(p,t), converts a decimal number p to t binary bits. In [3], set size S, is 16. Similarly, a function named Gray() is used for

encoding the Lehmer co-efficients. Where bin_bits is an array of binary bits. G is an array of corresponding Gray encoded bits.

**iv) S-ArbRO-2** This scheme is described in [5]. In this design the number of CRPs have been increased. Components are divided into elements. Each element has a pair of components associated with it as shown in Fig. 3 on the left side,

The difference between the counts of components in each element is the respective count associated with that element (r1-r2 or r2-r1). The next step is to select a group size for elements. This is done by selecting a value for parameter K. Inside this group, elements are added with each other. The range of K is $2 \leq K \leq N$.

**Algorithm 4** S-ArbRO-2

```
S-ArbRO-2(E[], K):      //K is the parameter passed.
combo=[]              //It will hold all the possible combinations
                                //of groups of Elements.
sums=[]           //It will hold the result of all the additions.
Kp= 2^(K-1)                 // power(2, K-1).
i = 0
for x in combinations(E,K): // Generate all combinations
                                // of K out of N elements.
        combo[i]= x
        i = i + 1
end for
for(i=0; i< len(combo); i++):
 for(j = 0; j< Kp; j++):
  temp=[]
  p=bin(j,K)                  //convert j into K binary bits.
   for(s=0; s<len(p); s++):
    if(p[s] =='0')
         temp = temp || (combo[i][s][0]- combo[i][s][1]) //r1-r2
    else
         temp = temp || (combo[i][s][1]- combo[i][s][0]) //r2-r1
    end if
   end for
   sums= sums || (sum(temp))
 end for
end for
for(i=0; i< len(sums); i++):
 if(sums[i]> 0)
        Response[i]=1
 else
        Response[i]=0
```

```
 end if
end for
```

---

If the result of adding elements is positive, the response is 1, otherwise it is 0. The total Challenge Response (CR) space is,

$$\text{Total CR space} = \frac{N!}{K! * (N-K)!} * 2^{K-1} \tag{1}$$

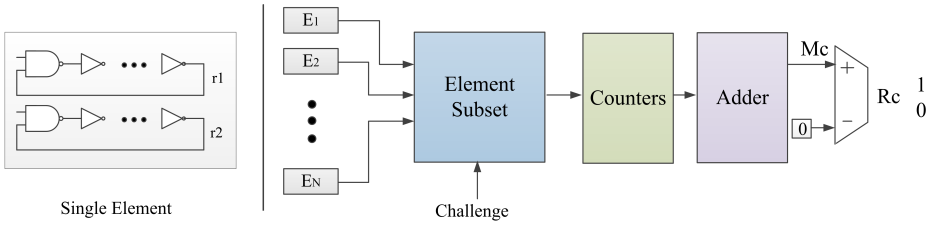


**Fig. 3.** Single element having two components (Left). S-ArbRO-2 showing the relationship between Challenge Response Pairs (Right)

As evident from the Fig. 3, the total number of elements is N. In the pseudocode, E[ ] is an input array of N elements.

Each Element has 2 components. K is the subset size. Combo holds all the possible combinations of $\binom{N}{K}$ elements. Sums will hold all the sums for K elements. Combination(N,K) calculates the $\binom{N}{K}$, sum(array) adds all the elements of an array. The response of S-ArbRO-2 is returned by an array Response[ ]. The pseudocode will generate all the possible CRPs. Assume that the component raw data is [10,5,6,4,17,11]. Therefore the three elements formed are $E_1$=[10,5], $E_2$=[6,4] and $E_3$=[17,11]. The possible number of combinations for K = 2 is $\binom{3}{2}$=3. Hence three groups of elements formed will be {$E_1$, $E_2$}, {$E_1$, $E_3$} and {$E_2$, $E_3$}. Combo will contain [{$E_1$, $E_2$}, {$E_1$, $E_3$},{$E_2$, $E_3$}] or [{[10,5], [6,4]}, {[10,5], [17,11]},{[6,4], [17,11]}]. If the group challenge is (01), it will select group [$E_1$,$E_3$]. Similarly inside each group if the challenge is (00), It will result in 0 $\implies$ $E_1$[r1-r2]= 10-5= 5 and 0 $\implies$ $E_3$[r1-r2]= 17-11= 6. Sums will hold 5+6=11. Since 11>0, therefore the final PUF response will be 1.

**v) Identity Mapping (Id-Map)** This scheme is described in [4]. In identity mapping M components can generate $2^M$ - M - 1 response bits. In this method, t component counts are selected from M component counts where $2 \leq t \leq M$. Initially all the pairs and triplets of component counts are determined by $| S_2 | = \binom{M}{2}$ and $| S_3 | = \binom{M}{3}$ respectively.

Likewise,

$$\mid S_t \mid = \binom{M}{t} \tag{2}$$

Then a random variable $Q_t$ is defined that assigns a real number X to each outcome of $S_t$, $Q_t : S_t \longrightarrow$ X such that

$$Q_t(f_{x1}, f_{x2}, ..f_{xt}) = \sum_{u=1}^{t-1} \sum_{v=u+1}^{t} w_{(xu)(xv)}. \parallel f_{xu} - f_{xv} \parallel^{e} \tag{3}$$

Where 1 ≤ x1, x2, x3,..., xt ≤ m.
And, x1 ≠ x2 ≠ x3 ≠ .. ≠ xt and 2 ≤ t ≤ M. The weight factor $w_{(xu)(xv)}$ can depend on a particular design. However, in our script it is equivalent to 1. Response R from Q is generated by using the following equation:

$$R = mod(Q[i]/q, 2)\ for\ i\ = 0, 1, 2, ... \tag{4}$$

q is the bucket size. The size of array Q depends on the value of t selected. For instance if t =2, the total elements of Q are $\binom{M}{2}$. If t=3, then total length of Q will be $\binom{M}{2} + \binom{M}{3}$, and so on. In addition to the response bits, a set of helper data is also generated. This helper data is used to reduce the effect of noise in the field.

**Algorithm 5** Identity Mapping

```
Identity_map(components[], q, t, e):
                    //q, t and e are parameters.
S=[]                //It will hold the (M t) component counts.
Q=[]                //It will hold the Qs.
i = 0
for x in combinations(components,t)
        S[i]= x
        i = i + 1
end for
j = 0
for a,b in combinations(S,2)
        Q[j]= (|a − b|)^e
        j = j + 1
end for
for(i=0; i< len(Q); i++)
        Response_enrollment[i]= (Q[i]/q) mod 2
        Wt[i] = (0.5 * q)       (Q[i]−       (Q[i]/q)       * q)
end for
for (i=0; i< len( Q   ); i++)     // Q    is generated in the field.
        temp[i]= ( Wt[i] + Q   [i])      //error correction
```

```
        Response_field[i]= (temp[i]/q) mod 2
end for
```

---

Helper data $W_t$ is calculated using the following equation,

$$W_t = \binom{q}{2} - (Q[i] - q * \lfloor \frac{Q[i]}{q} \rfloor) \; for \; i \; = 0, 1, 2, 3... \tag{5}$$

In (5), q is the bucket size. In the pseudocode, components[] is an input array that contains the counts of components, parameter e is any real number $\neq 1$. Parameter t can hold values , such that $2 \leq t \leq m$. PUF Response during enrolment is stored in an array Response_enrollment. While $W_t$ is the array that contains the helper data. In the field, each response bit is recalculated using $W_t$ and noisy Q values. Response_field[] contains the PUF response generated at the field.

## 4 Methodology

We use three properties of PUF to rank the schemes. These properties are Average uniformity, WC uniqueness and WC reliability. These properties are explained here, Uniformity of a PUF estimates how uniform the proportion of 0s and 1s are in the PUF response of a device. It is calculated using (6),

$$Uniformity(i) = \frac{1}{L} \sum_{l=1}^{L} r_{i,l} * 100\% \tag{6}$$

Where $r_{i,l}$ is the lth binary bit in the response of a chip i. Response of each device is L bits. Average uniformity of N devices is shown in (7). The optimal value is 50% for a set of N devices.

$$Avg \; Uniformity = \frac{1}{N} \sum_{i=1}^{N} Uniformity(i) \tag{7}$$

Worst case uniqueness is equal to minimum relative Hamming Distance between the response R(i) and response R(j), as well as between R(i) and complement of R(j). It is determined by looking at all pairs of responses from devices i and j. Its optimal value is equal to 50%. It is calculated at the room temperature and nominal voltage. It is defined as:

$$WC \; Uniqueness = \min_{i=1,j=i+1}^{i=N-1,j=N} \Big( \frac{min(HD(R_i, R_j), L - HD(R_i, R_j))}{L} \Big) * 100\% \tag{8}$$

$R_i$ and $R_j$ are the responses of two FPGA devices. To determine the best scheme we chose the one that has the highest WC uniqueness. The worst case reliability

is calculated using the following equation,

$$WC\ Reliability = \min_{i=1}^{N} \left(1 - \left(\frac{\max_{c=0}^{C} HD(R_i, R_{i,c})}{L}\right)\right) * 100\% \quad (9)$$

It describes how close to the 100% reproduction of the PUF bits a given scheme is getting in the worst case. Optimal value is equal to 100%. In (9), HD is the hamming distance. $R_i$ is the response of device i at the nominal condition. $R_{i,c}$ is the response of device i in the field. L is response size in bits and C is the total number of conditions applied in the field. The best scheme in terms of WC reliability is the one that has the highest value for (9). Overall, we used the following equation to determine the optimum parameters:

$$Z = Max\{50\% - Avg\ Uniformity, 50\% - WC\ Uniqueness, 100\% - WC\ Reliability\} \quad (10)$$

We choose the parameter for any dataset which gives the smallest value for Z according to (10). These parameters are shown in Tables 4 to 9. The input to PUF-ID generating scripts is in the chip-row format. In this format the rows contain the data for a particular device while the columns contain the M components.

## 5 Results

In order to generate same number of PUF response bits using different schemes, we chose the PUF response size of 128 bits. It is due to the fact that Pair-wise Comparison (PC) generates only 128 bits using all the 256 components of SR_latch data.

**Table 1.** Details of dataset.

| | CNC | PC | BLG | S-ArbRO-2 | Id-Map |
|---|---|---|---|---|---|
| Parameters | | | Set size = 16 | K=2 | t=2 |
| PUF Response Length (L) | 128 | 128 | 128 | 128 | 128 |
| Min Components Required (M) | 129 | 256 | 48 | 24 | 17 |
| PUF Response (L) | M-1 | $\lfloor M/2 \rfloor$ | (M/S)*L | $\binom{M}{K} * 2^{k-1}$ | $\binom{M}{t}$ |

**Zynq Data set**: Total devices = 10, Components per device = 256
**Spartan-6 Data set**: Total devices = 25, Components per device = 256
**Spartan-3 Data set**: Total devices = 193, Components per device = 512

From Table 2, 3 and 4, it is evident that the best scheme in terms of WC uniqueness is PC for both Zynq and Spartan-3 datasets. However for Spartan-6 data it is the CNC scheme that generates the best results. Similarly for Average uniformity the best results are offered by BLG in Zynq dataset and S-ArbRO-2

**Table 2.** Zynq data.

| | CNC | PC | BLG | S-ArbRO-2 | Id-Map |
|---|---|---|---|---|---|
| Parameters | | | Set size = 16 | K = 2 | e=0.5, q=2, t=2 |
| Inter-chip HD Mean | 49.02% | 49.87% | 46.64% | 50.46% | 50.14% |
| Inter-chip HD Min | 38.28% | 42.19% | 36.72% | 29.69% | 39.84% |
| Inter-chip HD Max | 57.03% | 58.59% | 53.91% | 71.88% | 58.59% |
| Inter-chip HD Std Dev | 4.48% | 3.36% | 3.8% | 10.28% | 4.37% |
| WC Uniqueness | 38.28% | **41.41%** | 36.72% | 28.12% | 39.84% |
| Avg Uniformity | 46.87% | 46.79% | **47.89%** | 54.14% | 47.34% |
| Std Dev Uniformity | 1.39% | 4.49% | 4.58% | 10.19% | 12.44% |

**Table 3.** Spartan-6 Data

| | CNC | PC | BLG | S-ArbRO-2 | Id-Map |
|---|---|---|---|---|---|
| Parameters | | | Set size = 16 | K = 2 | e=0.5, q=1, t=2 |
| Inter-chip HD Mean | 49.36% | 49.25% | 46.12% | 49.77% | 46.37% |
| Inter-chip HD Min | 35.16% | 33.59% | 32.81% | 25.78% | 34.38% |
| Inter-chip HD Max | 63.28% | 60.16% | 59.38% | 83.59% | 62.5% |
| Inter-chip HD Std Dev | 5.2% | 4.45% | 4.34% | 10.41% | 4.82% |
| WC Uniqueness | **35.16%** | 33.59% | 32.81% | 16.41% | 34.38% |
| Avg Uniformity | 44.03% | 44.71% | 44.65% | **52.96%** | 63.68% |
| Std Dev Uniformity | 1.95% | 4.52% | 4.24% | 8.63% | 6.41% |

**Table 4.** Spartan-3 Data

| | CNC | PC | BLG | S-ArbRO-2 | Id-Map |
|---|---|---|---|---|---|
| Parameters | | | Set size = 16 | K = 2 | e=0.5, q=30, t=2 |
| Inter-chip HD Mean | 46.83% | 46.61% | 45.85% | 46.13% | 48.76% |
| Inter-chip HD Min | 28.91% | 30.47% | 26.56% | 13.28% | 24.22% |
| Inter-chip HD Max | 65.63% | 62.5% | 66.41% | 83.59% | 64.84% |
| Inter-chip HD Std Dev | 4.86% | 4.45% | 4.76% | 10.2% | 4.47% |
| WC Uniqueness | 28.91% | **30.47%** | 26.56% | 13.28% | 24.22% |
| Avg Uniformity | 49.86% | 52.25% | 47.70% | **50.24%** | 57.80% |
| Std Dev Uniformity | 2.53% | 4.58% | 6.70% | 9.39% | 4.96% |

in both Spartan-6 and Spartan-3 datasets. Tables 5-9 shows the worse case reliability as defined in (9). Spartan-3 data set contains voltage and temperature data for only five devices. Similarly Zynq data set contains voltage and temperature data for ten devices. Additionally the Spartan-6 data set contains voltage and temperature data for fifteen devices. The nominal voltage of Zynq devices is 1V on the other hand it is 1.2V for both Spartan-6 and Spartan-3 devices.

**Table 5.** CNC and PC results

| | CNC | | | PC | | |
|---|---|---|---|---|---|---|
| | Zynq | Spartan-6 | Spartan-3 | Zynq | Spartan-6 | Spartan-3 |
| PUF Type | SR-Latch | SR-Latch | RO-PUF | SR-Latch | SR-Latch | RO-PUF |
| Rel @ +5%V | 93.75% | 99.21% | **87.50%** (+10%V) | 93.75% | 97.65% | **91.40%** (+10%V) |
| Rel @ -5%V | 93.75% | 95.31% | 91.40% (-10%V) | 93.75% | **96.09%** | 92.96% (-10%V) |
| Rel @ 85C | **91.40%** | **94.53%** | 95.31% (+65C) | **89.84%** | 96.09% | 92.96% (+65C) |
| Rel @ 0C | 94.53% | 96.87% | N/A | 94.53% | 96.09% | N/A |

**Table 6.** BLG and S-ArbRO-2 results

| | BLG | | | S-ArbRO-2 | | |
|---|---|---|---|---|---|---|
| | Set size = 16 | | | K=2 | | |
| | Zynq | Spartan-6 | Spartan-3 | Zynq | Spartan-6 | Spartan-3 |
| Rel @ +5%V | 86.71% | 94.57% | **83.59%** (+10%V) | 92.18% | 96.87% | 48.14% (+10%V) |
| Rel @ -5%V | 85.93% | 90.62% | 84.37% (-10%V) | 92.96% | 96.09% | **38.2%** (-10%V) |
| Rel @ 85C | **78.9%** | **85.93%** | 90.62% (+65C) | **83.59%** | **89.06%** | 39.06% (+65C) |
| Rel @ 0C | 89.84% | 92.18% | N/A | 92.96% | 93.75% | N/A |

From Table 5-9, the best scheme in terms of worst case reliability is CNC for Zynq, PC for Spartan-6 and Id-Map for Spartan-3 datasets. The bold values shown in each column shows the minimum value. For any dataset the best scheme is chosen that results in the highest bold value. The worst result is offered by S-ArbRO-2 for Spartan-3 dataset. It might be due to the fact that certain ROs are affected more by voltage and temperature variation than others. In S-ArbRO-2 if the sign of the elements change in the field. Then it results in a bit flip, hence low reliability.

## 6 Conclusion and Future Work

From this work we conclude that PUF responses should be generated using multiple schemes to determine the uniformity and worst cases of uniqueness and

**Table 7.** Identity Mapping scheme

| | Zynq | Spartan-6 | Spartan-3 |
|---|---|---|---|
| Parameters | q=2 | q=1 | q=30 |
| Rel @ +5%V | **75.81%** | 89.84% | 94.53% (+10%V) |
| Rel @ -5%V | 91.4% | **85.93%** | **92.96%** (-10%V) |
| Rel @ 85C | 64.84% | 89.84% | 97.65%(+65C) |
| Rel @ 0C | 92.18% | 90.62% | N/A |

reliability. S-ArbRO-2, Lehmer-Gray and Identity mapping offers the ability to use less number of components for PUF design; however the CRPs are no longer independent. BLG scheme offers the best results in terms of uniformity in Zynq, however it is the S-ArbRO-2 scheme that generates the best uniformity results for both Spartan-3 and Spartan-6 datasets. In case of uniqueness PC scheme offers best results for both Zynq and Spartan-3 datasets. In case of Reliability we have no winner as three different schemes offer the best results for three data sets. In the future we intend to enhance the scope of our method by including the Arbiter PUF data set [9] in our analysis. We plan to post our scripts at [10].

## References


1. Maiti, A., Schaumont, P.: Improved Ring oscillator PUF: An FPGA Friendly Secure Primitive. Journal of Cryptology, vol. 24, No. 2, pp. 375–397, Oct. 2010
2. Hori, Y., Yoshida, T., Katashita, T., Satoh, A .: Quantitative and statistical performance evaluation of arbiter physical unclonable functions on FPGAs. In Proc. International conference on Recongurable computing and FPGAs (ReConFig) 2010, pp 298-303, Dec 2010
3. Maes, R., Herrewege, A., Verbauwhede, I.: PUFKY: A Fully Functional PUF-based Cryptographic Key Generator. In Proc. Workshop on Cryptographic Hardware and Embedded Systems (CHES), 2012, LNCS vol. 7428, pp. 302–319
4. Maiti, A., Kim, I., Schaumont, P.: A Robust Physical Unclonable Function With En-hanced Challenge-Response Set. IEEE Transactions on Information Forensics and Security, Vol. 7, No. 1, February 2012
5. Ganta, D., Nazhandali, L.: Easy-to-Build Arbiter Physical Unclonable Function with Enhanced Challenge/Response Set. In Proc. Quality Electronic Design (ISQED), 2013 14th International Symposium on Quality Electronic Design
6. Maiti, A., Gunreddy, V., Schaumont, P.:. A Systematic Method to Evaluate and Compare the Performance of Physical Unclonable Functions. in Embedded Systems Design with FPGAs, Springer, 2013, pp. 245–267
7. http://rijndael.ece.vt.edu/puf/detailed.php
8. Habib, B., Kaps, J., Gaj, K.: Efficient SR-Latch PUF. In Proc. 11th International Symposium on Applied Reconfigurable Computing, 2015, Bochum, Germany, April 15-17. 2015
9. https://staff.aist.go.jp/hori.y/en/puf/index.html
10. https://cryptography.gmu.edu/puf