

# New Results on Optimal Fixed Length Physical Random Number Post-Processing Functions

Markus Dichtl

Siemens Corporate Technology

**Abstract.** When functions with  $n$  input bits and  $m$  output bits are used for post-processing the output of a physical random number generator, which is assumed to produce statistically independent bits with a fixed bias  $\epsilon$ , the probabilities of the  $2^m$  outputs can be represented as polynomials in  $\epsilon$ . For a good post-processing function all low powers of  $\epsilon$  should have coefficients of zero. [4] showed for most cases with  $1 \leq m \leq n \leq 16$  how many low powers of  $\epsilon$  can be maximally eliminated, but there were 13 cases remaining open. This paper solves these 13 cases.

**Key words:** Physical random numbers, post-processing, bias.

## 1 Introduction

The output of physical random number generators is never perfect. There are always deviations from the mathematical ideal of uniformly distributed, statistically independent random bits. Therefore, algorithmic post-processing is often used to improve the statistical quality of bits produced by physical random number generators. The von Neumann algorithm [3] and the exclusive-or operation are well known methods for post-processing physical random numbers. The von Neumann post-processing results in statistically perfect output bits if the assumption that the input bits are statistically independent is fulfilled. Its drawback is that its latency is unbounded, that is, it can require an arbitrarily large number of input bits until it produces a single bit of output. As this is not desirable for real time applications, which have to produce output always within a defined time, it is useful to study fixed length post-processing functions  $f$  which map  $n$  input bits to  $m$  output bits. We assume that  $n$  and  $m$  are positive integers and  $m \leq n$ . In this paper, we only consider the case of statistically independent input bits from a stationary source, that is the only deficiency of the input bits considered is bias. Bias is the deviation of the probability of a bit to have the value 1 from the ideal value  $1/2$ . If  $p$  is the probability of a bit to be one, the bias  $\epsilon$  is defined as  $\epsilon = 1/2 - p$ . We assume that the bias  $\epsilon$  is constant, but that its numerical value is unknown.

### 1.1 Previous Work

Conventionally, to produce  $m$  output bits from  $km$  random input bits, the input is arranged in  $m$  sets of cardinality  $k$ , and the  $k$  bits in each of the sets are

XOR-ed together to determine an output bit. As there are  $m$  sets, we get  $m$  output bits as desired.

In [1] we described our discovery that for mapping 16 biased output bits of a physical random number generator to 8 result bits of the post-processing, one can do much better than pairwise XOR-ing of the input bits. In order to make this claim precise, one has to define the meaning of the term “better”. The approach introduced in [1] is to describe the probabilities of the output values as functions of the bias  $\epsilon$ . These functions are polynomials in  $\epsilon$ . The minimal positive exponent of  $\epsilon$  with non-zero coefficient in these polynomials is considered as a measure of quality of the post-processing function. We only consider post-processing functions  $f$  for which the constant term in all probabilities is  $2^{-m}$ , that is functions which deliver perfect unbiased output for unbiased input.

In [1] it was shown that there is a post-processing function with 16 input bits and 8 output bits whose output probabilities contain only 6-th and higher powers of  $\epsilon$ . For comparison, pairwise XOR-ing of the input bits eliminates the linear terms in  $\epsilon$ , but not the squares. [1] also shows that the new post-processing function succeeds to squeeze out much more entropy from the biased input bits than XOR.

As a reaction on [1], two follow-up papers by other authors appeared. In [4], K. Suzuki and T. Iwata widened the scope considered from functions with 16 input bits and 8 output bits to the general question of  $n$  input bits and  $m$  output bits, and provided formal definitions of some concepts. Most notably, they define  $\text{deg}(n, m)$ . Above, we considered for a given post-processing function  $f$  the minimal positive exponent  $g$  of  $\epsilon$  with non-zero-coefficient in the probabilities of the output values as a function of the bias  $\epsilon$ .  $\text{deg}(n, m)$  is the maximum of these exponents  $g$  over all admissible post-processing functions with  $n$  input bits and  $m$  output bits. Again, only functions are accepted as admissible post-processing function for which all outputs have the probability  $2^{m-n}$  for unbiased input bits. [4] proves general properties of  $\text{deg}(n, m)$  and provides numerical bounds of it for all values  $1 \leq m \leq n \leq 16$ . In most of these cases, the bounds are sharp enough to determine the exact value of  $\text{deg}(n, m)$ . In 13 cases, however,  $\text{deg}(n, m)$  is not completely determined, but there remain two possible values whose difference is 1.

In [2], P. Lacharme considered the problem from a coding theoretic viewpoint. He identified some of the constructions described in [1] as linear code. He shows generally that, if an  $[n, m, d]$  linear code exists, then there is a post-processing function with  $n$  input bits and  $m$  output bits, for which in the output probabilities the coefficients of  $\epsilon, \epsilon^2, \dots, \epsilon^{d-1}$  are all zero. The existence of  $(n, m, t)$  resilient functions also implies the existence of post-processing function for which the coefficients of  $\epsilon, \epsilon^2, \dots, \epsilon^t$  are all zero. The results of this paper are very valuable for constructing concrete post-processing functions, but it is not the final answer to all questions in this field, as the implications are not reversible. There can be better constructions than those based on resilient functions. So, [2] can be used to determine some values of  $\text{deg}(n, m)$ , but not all of them.

## 1.2 Achievements of this paper

This paper fills in the 13 values of  $\deg(n, m)$  for  $1 \leq m \leq n \leq 16$  which were left open by [4].

## 2 Optimality Results for Fixed Length Post-Processing Functions

### 2.1 Achieving the Upper Bounds

In all cases except one of the open cases from [4], the numerical upper bounds given for  $\deg(x, y)$  are achievable. We just give a post-processing function which achieves the upper bound. To verify this, one just has to sum up the output probabilities (hopefully with the help of a computer algebra system) and to check that the upper bound is indeed achieved.

$\deg(12, 3) = 7$	
Number of outputs	Hamming weight of input
	0 1 2 3 4 5 6 7 8 9 10 11 12
1	0 0 14 35 0 212 28 114 80 20 6 3 0
1	0 1 16 3 86 128 14 212 2 39 10 1 0
1	0 1 16 4 80 142 0 212 16 25 16 0 0
1	0 2 2 58 3 134 154 30 92 32 4 0 1
1	0 2 12 0 134 2 182 86 50 36 6 2 0
2	0 3 1 44 50 72 196 2 126 5 11 2 0
1	1 0 4 32 92 30 154 134 3 58 2 2 0

How to read this table? We have 7 different types of output of the function; each type is described by a row of the table. Let's consider the first row of the table above. It says that the first output type combines 0 inputs of Hamming weight 0, 0 inputs of Hamming weight 1, 14 inputs of Hamming weight 2, 35 inputs of Hamming weight 3,  $\dots$ . The number 1 in front of the first row indicates that there is one 3 bit output which is of this type. In this case, only the 6-th output type occurs for more than a single output value. Which inputs of a given weight are used for the different output, and which 3 bit output patterns are assigned to the different output types is not important, but evidently a fixed assignment is needed for practical implementations.

deg(12, 5) = 5													
Number of outputs	Hamming weight of input												
	0	1	2	3	4	5	6	7	8	9	10	11	12
1	0	0	0	10	28	10	2	72	0	2	2	2	0
8	0	0	0	12	25	0	29	52	3	0	7	0	0
2	0	0	0	18	8	0	74	2	12	12	2	0	0
1	0	0	1	4	40	2	1	71	6	0	0	3	0
2	0	0	1	15	10	2	69	11	0	20	0	0	0
2	0	0	2	12	12	2	74	0	8	18	0	0	0
8	0	0	7	0	3	52	29	0	25	12	0	0	0
1	0	1	0	0	44	6	3	56	15	1	1	0	1
2	0	1	0	0	46	1	1	72	0	4	1	2	0
3	0	2	1	4	0	72	1	1	46	0	0	1	0
1	0	3	0	0	7	67	6	2	35	8	0	0	0
1	1	0	0	8	0	65	9	2	37	5	1	0	0

deg(13, 4) = 7														
Number of outputs	Hamming weight of input													
	0	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0	0	8	20	25	96	148	8	174	0	28	4	1	0
1	0	0	8	24	0	160	64	64	160	0	24	8	0	0
2	0	0	10	7	60	47	183	1	167	5	27	4	1	0
1	0	1	0	35	43	6	231	57	33	92	13	0	0	1
2	0	1	2	27	52	4	246	8	82	83	0	5	2	0
3	0	1	3	31	1	165	1	197	33	54	24	0	2	0
3	0	1	7	9	47	125	1	225	5	78	4	10	0	0
2	0	2	3	8	85	13	183	29	151	5	26	7	0	0
1	1	0	2	3	109	27	43	245	6	41	31	4	0	0

deg(13, 6) = 5														
Number of outputs	Hamming weight of input													
	0	1	2	3	4	5	6	7	8	9	10	11	12	13
5	0	0	0	1	35	0	3	58	25	0	1	5	0	0
11	0	0	0	11	0	36	0	58	0	20	0	3	0	0
3	0	0	0	14	0	1	73	8	2	25	5	0	0	0
7	0	0	1	1	30	2	3	77	0	0	14	0	0	0
7	0	0	1	11	2	1	77	2	0	33	0	1	0	0
1	0	0	2	1	10	45	3	1	64	0	1	0	0	1
9	0	0	3	0	20	0	58	0	36	0	11	0	0	0
7	0	0	3	1	18	0	53	25	0	22	6	0	0	0
13	0	1	1	2	0	59	1	1	59	0	2	1	1	0
1	1	0	1	0	0	55	13	22	1	35	0	0	0	0

deg(14,3) = 9															
Number of outputs	Hamming weight of input														
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	0	1	19	28	126	304	306	396	552	37	259	0	18	2	0
1	0	1	21	8	214	82	654	60	720	49	181	52	2	4	0
1	0	1	22	2	225	88	600	144	678	13	250	6	17	2	0
1	0	2	8	72	53	300	525	24	810	34	138	80	1	0	1
2	0	3	6	61	83	361	144	774	18	505	2	85	3	3	0
1	0	3	6	64	59	442	0	900	18	379	146	4	27	0	0
1	1	0	3	68	158	64	630	360	189	480	23	52	20	0	0

deg(14,4) = 8															
Number of outputs	Hamming weight of input														
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	0	0	12	11	52	212	0	450	0	212	52	11	12	0	0
2	0	1	1	51	3	154	266	86	230	181	5	39	7	0	0
2	0	1	5	22	92	6	406	16	244	177	13	34	8	0	0
5	0	1	6	21	73	76	320	2	398	1	98	25	1	2	0
3	0	1	6	25	37	218	0	450	6	197	66	5	13	0	0
2	0	1	8	13	65	188	12	450	6	209	36	33	1	2	0
1	1	0	3	1	153	60	35	518	35	60	153	1	3	0	1

deg(14,6) = 6															
Number of outputs	Hamming weight of input														
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
2	0	0	0	14	0	56	0	116	0	56	0	14	0	0	0
2	0	0	0	14	0	56	2	104	30	16	30	2	2	0	0
3	0	0	0	15	0	35	71	5	99	1	21	8	1	0	0
2	0	0	0	16	5	1	120	20	2	86	0	4	1	1	0
20	0	0	1	2	41	0	22	124	22	0	41	2	1	0	0
12	0	0	1	13	6	5	118	18	0	90	1	1	2	1	0
1	0	0	3	2	25	0	108	0	52	56	1	6	3	0	0
7	0	0	5	1	1	60	58	6	58	60	1	1	5	0	0
2	0	1	0	7	2	67	32	1	122	4	4	16	0	0	0
3	0	1	1	3	6	71	22	5	126	0	5	16	0	0	0
9	0	1	2	0	5	85	1	19	119	6	1	17	0	0	0
1	1	0	0	3	0	79	37	11	51	65	6	2	0	0	1

deg(14, 7) = 5	
Number of	Hamming weight of input
outputs	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
1	0 0 0 0 24 4 0 66 2 24 4 2 2 0 0
1	0 0 0 1 9 37 2 3 51 23 1 0 0 0 1
4	0 0 0 1 21 2 23 18 51 2 1 9 0 0 0
25	0 0 0 1 23 1 1 72 6 5 17 1 1 0 0
1	0 0 0 2 21 0 4 73 5 2 18 3 0 0 0
32	0 0 0 4 3 38 2 1 73 0 0 5 2 0 0
18	0 0 0 7 0 28 0 58 0 28 0 7 0 0 0
4	0 0 0 9 1 0 60 3 33 2 18 2 0 0 0
4	0 0 1 4 8 8 22 52 8 0 25 0 0 0 0
1	0 0 3 0 8 0 62 0 32 0 23 0 0 0 0
4	0 0 3 3 0 0 76 0 4 28 13 1 0 0 0
18	0 0 4 0 2 2 72 4 2 26 16 0 0 0 0
14	0 1 0 1 8 8 56 3 0 50 0 0 0 1 0
1	1 0 0 0 0 28 42 8 7 28 14 0 0 0 0

deg(15, 2) = 11	
Number of	Hamming weight of input
outputs	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
1	0 2 52 11 522 682 946 2211 1188 1166 1144 17 242 6 2 1
1	0 4 41 9 699 11 2222 858 1782 1562 341 589 31 39 4 0
1	0 7 4 215 17 1496 11 3102 330 1991 506 347 149 10 7 0
1	1 2 8 220 127 814 1826 264 3135 286 1012 412 33 50 2 0

deg(15, 8) = 5	
Number of	Hamming weight of input
outputs	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
1	0 0 0 0 0 42 0 15 0 70 0 0 0 0 0 1
2	0 0 0 0 7 28 0 8 50 28 0 0 7 0 0 0
56	0 0 0 0 15 0 30 0 60 0 18 0 5 0 0 0
16	0 0 0 1 14 2 4 64 3 11 26 1 1 1 0 0
15	0 0 0 4 3 0 56 0 1 56 3 4 0 0 1 0
45	0 0 0 5 0 18 0 60 0 30 0 15 0 0 0 0
18	0 0 1 1 1 27 0 37 23 14 21 0 2 1 0 0
24	0 0 1 1 1 28 3 14 59 4 0 17 0 0 0 0
14	0 0 1 1 10 2 7 69 4 0 26 8 0 0 0 0
22	0 0 1 2 0 12 44 3 1 60 1 1 1 2 0 0
27	0 0 1 2 0 21 14 23 37 0 27 1 1 1 0 0
15	0 1 0 0 4 3 56 1 0 56 0 3 4 0 0 0
1	1 0 0 0 0 1 65 10 5 5 41 0 0 0 0 0

deg(16, 4) = 9																	
Number of	Hamming weight of input																
outputs	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	0	0	0	99	2	176	1021	211	633	1352	146	201	244	8	1	1	1
3	0	0	12	47	2	579	44	1166	396	1166	44	579	2	47	12	0	0
3	0	0	14	21	141	159	868	11	1662	2	938	55	205	7	12	1	0
1	0	0	15	9	201	1	1076	2	1194	890	47	597	13	37	14	0	0
1	0	0	19	2	114	445	170	848	1152	158	803	278	38	61	8	0	0
1	0	1	0	73	97	15	1204	2	846	1283	4	393	153	21	0	4	0
2	0	2	2	61	6	566	45	1016	1008	98	1080	1	162	46	1	2	0
2	0	3	2	14	248	8	646	1049	0	1547	14	424	104	26	10	1	0
1	0	4	0	20	161	366	52	1241	846	44	1156	42	89	74	0	1	0
1	1	1	0	3	308	3	367	1475	9	919	718	105	130	53	3	1	0

deg(16, 6) = 7																	
Number of	Hamming weight of input																
outputs	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
3	0	0	0	0	90	12	0	403	180	60	112	162	2	0	0	3	0
5	0	0	0	6	74	11	4	507	15	25	351	3	3	24	1	0	0
1	0	0	0	6	75	4	25	472	50	4	358	2	3	24	1	0	0
1	0	0	0	10	55	40	1	473	43	25	347	1	2	27	0	0	0
8	0	0	0	19	4	131	13	228	330	4	156	129	2	1	7	0	0
3	0	0	0	21	0	128	1	318	204	4	294	21	12	20	1	0	0
14	0	0	0	21	20	17	224	186	8	410	0	113	4	21	0	0	0
1	0	0	1	3	68	7	181	24	491	16	25	205	0	1	1	0	1
11	0	0	4	0	56	0	252	0	400	0	252	0	56	0	4	0	0
9	0	0	8	0	0	122	152	144	8	500	0	0	88	2	0	0	0
2	0	2	0	0	16	176	26	4	578	4	14	194	6	2	0	2	0
3	0	2	1	0	1	211	0	8	565	31	0	195	6	0	3	1	0
2	0	3	0	0	3	154	140	4	250	347	28	4	91	0	0	0	0
1	1	0	0	2	13	165	66	9	484	11	201	1	66	4	1	0	0

Now some curious readers may not be satisfied by just verifying that the given functions achieve the upper bounds, but might want to know how these functions were found. We just used a small variation of the method suggested in [4]. There, the first step for finding an optimal post-processing function is to determine all solutions of equations (3) and (4) of [4]. Solving these equations just means to find subsets of  $2^{n-m}$  inputs with given Hamming weights such that all the required  $\epsilon$  powers disappear when we sum over the probabilities of the subset. But as a matter of fact, not all solutions of (3) and (4) are needed. We chose suitable subsets which were sufficient to solve equation (5) from [4]. Solving this equation means to assign all possible inputs to one of the subsets identified earlier. Our choice of subsets did not follow clearly defined criteria, but was quite subjective. If the subset turned out to give no solution of (5) in

[4], it was extended. The sizes of the subsets considered varied from about 20 to 70.

## 2.2 The Unachievable Upper Bound

In only one case, namely for  $\text{deg}(16, 7)$ , the upper bound given in [4] can not be achieved. How can we show this? We have to look at the solutions of equation (3) and (4) from [4]. That is, we want to find a subset of  $2^9$  inputs of various Hamming weights such that the coefficients of  $\epsilon, \epsilon^2, \dots, \epsilon^6$  eliminate in the sum of the probabilities. For a moment, we give up the condition that the coefficients  $q_w$ , that is the number of inputs with Hamming weight  $w$ , must be integers. However, we observe the constraints that the number of inputs with a given Hamming weight must be non-negative, that there is only one input with Hamming weight 0, and also only one with Hamming weight 16. Then we use linear programming to determine the maximum solution for  $q_0$ . It is  $592/891$ . But now we remember again that we need integer solutions, so the value is really 0. For the input with Hamming weight 0 the  $\epsilon$ -powers up to 6 can not be eliminated. This means  $\text{deg}(16, 7) \leq 6$ . Combining this with the lower bound  $\text{deg}(16, 7) \geq 6$  from [4] we get the exact value  $\text{deg}(16, 7) = 6$ .

## 3 Conclusion

The results of [4] together with the results of this paper give the exact values of  $\text{deg}(n, m)$  for  $1 \leq m \leq n \leq 16$ .

## References

1. M. Dichtl, "Bad and Good Ways of Post-processing Biased Physical Random Numbers", Fast Software Encryption 2007, Springer, Lecture Notes in Computer Science 4593, pp. 137-152, 2007
2. P. Lacharme, "Post-Processing Functions for a Biased Physical Random Number-Generator", Fast Software Encryption 2008, Springer, Lecture Notes in Computer Science 5086, pp. 334-342, 2008
3. J. von Neumann, "Various techniques for use in connection with random digits", von Neumann's Collected Works, Volume 5, pp. 768-770, Pergamon, 1963
4. K. Suzuki and T. Iwata, "Bounds on Fixed Input/Output Length Post-Processing Functions for Biased Physical Random Number Generators", SAC 2008 (15th Annual Workshop on Selected Areas in Cryptography), preprint, 2008