Improving differential properties of S-boxes with local changes of DDT

(Extended Abstract)

Pavol Zajac[1*]

[1*]Department of Computer Science and Mathematics, Slovak University of Technology in Bratislava, Ilkovicova 3, Bratislava, 81219, Slovakia.

Corresponding author(s). E-mail(s): pavol.zajac@stuba.sk;

Keywords: differential uniformity, S-box generation algorithm, cryptographic applications of Boolean functions

1 Introduction

In a recent article [1] we have proposed an algorithm to construct S-boxes with prescribed differential properties. The main principle is to produce the function assignment in discrete increments while checking the restrictions on a partially constructed difference distribution table. Although the algorithm can find any S-box with prescribed differential properties, it is impractical due to its high complexity. On the other hand, it can produce S-boxes with better differential properties than a random selection. We can naturally ask whether it is possible to obtain better S-boxes by manipulating the vector of values in a systematic way by taking into account the differential properties of the S-box.

We already know that the answer is positive. There is a large number of results concerning heuristic methods (e.g., evolutionary algorithms) that produce better S-boxes than a random search. However, these methods are typically generic, focusing on improving some objective function with a stochastic search directed by an objective function. The objective function typically includes but is not restricted to, the differential properties of the S-box.

In our current research, we take a different approach to the problem. We start with a random S-box and specify the operation we are allowed to do with the S-box value vector. We are concerned with bijective S-boxes, where the value vector is a permutation. Our permitted operations are either swaps (interchange of values of $y_1 = S(x_1)$ and $y_2 = S(x_2)$) or application of cycles. We select the operation based on the difference distribution table properties in a systematic way. Our experimental results show that it is possible to significantly improve the differential properties of an S-box, but there seems to be a limit to which this strategy converges.

2 Methods

Let $S : \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ be a bijective vectorial Boolean function (an S-box). The difference distribution table DDT of S contains values

$$DDT(dx, dy) = |\{x \in \mathbb{Z}_2^n : S(x) \oplus S(x \oplus dx) = dy\}| .$$

S-box S is δ-differentially uniform, if $\delta = \max_{dx \neq 0} \{DDT(dx, dy)\}$. Let c_0 denote the number of zero elements of DDT of S (for nonzero dx),

$$c_0 = |\{(dx, dy) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n : DDT(dx, dy) = 0 \land dx \neq 0\}| .$$

In ideal case, $\delta = 2$ and $c_0 = 2^n(2^n - 1)/2$ (for APN function S). Randomly selected S-box will have higher δ and c_0 due to collisions of differences: for some dx, randomly selected x_1 and $x_2 \neq x_1$, $x_2 \neq x_1 \oplus dx$ will produce the same difference $dy = S(x_i) \oplus S(x_i \oplus dx)$. Such collisions increase the maximum value in DDT and the number of zero positions in DDT (if there is a colliding pair, there will not be enough remaining x values to produce some dy).

Suppose that there exists a bijective APN S-box (or some S-box of desired quality), and consider it as a permutation. By composing the S-box with some other permutation, we will change the properties of the DDT in some way, typically decreasing the quality of the S-box. On the other hand, with systematic compositions (e.g., by using a swap of two elements, a transposition), it is possible to construct any other permutation from it. This process can work in the opposite direction: starting from a random S-box, we might reach an S-box with high quality. Unfortunately, in each step we can choose from a large number of transpositions, and the complexity of constructing the desired permutation blindly is super-exponential.

Our main research question is as follows: Starting from a random S-box, can we use a greedy selection of transpositions directed by DDT properties to find the best possible S-box? If not, how far can we improve our initial random selection? We do not know the theoretical answers to this question, but we present some experimental results for consideration.

In our experiments, we use the following method to improve S-boxes:

1. Start from a randomly selected S-box S.
2. For each dy compute list $D(dy)$ which contains quadruples (y_1, y_2, y_3, y_4), such that $dy = y_1 \oplus y_2 \oplus y_3 \oplus y_4$, with $y1 = S(x_1)$, $y_2 = S(x_1 \oplus dx)$, $y3 = S(x_2)$, $y_4 = S(x_2 \oplus dx)$, with $dx \neq 0$, $x_2 \neq x_1$ and $x_2 \neq x_1 \oplus dx$.

	Iterations			Changes	Initial $D(0)$			Final $D(0)$		
	min	avg	max	avg	min	avg	max	min	avg	max
Swaps	55	70.46	93	140.92	7734	8208.63	8601	4983	5161.08	5397
3-cycles	34	48.28	64	144.84	7710	8184.72	8571	4887	5082.78	5358
4-cycles	26	36.91	50	147.85	7776	8184.96	8661	4926	5053.53	5271

Table 1 The results of the experiments: number of iterations (and estimated average number of changes in S-box), initial, and final sizes of $D(0)$ sets.

3. List $D(0)$ contains "DDT collissions". In the ideal case, we want this list to be empty. We terminate the algorithm if $D(0)$ is empty, or if we cannot decrease its size by any transposition (see the next step).
4. For each pair (y, z), $y \neq z$, compute the effect of applying transposition $y \leftrightarrow z$ to S on the size of $D(0)$ (see discussion below).
5. Apply the transposition that minimizes the next $D(0)$. Repeat the algorithm with the new S-box.

The crucial step in the algorithm is the estimation of the size of $D(0)$ in the next step. In the case of transpositions, we can compute the value exactly. Each element of type (y, z, y_3, y_4) will remain in the corresponding set $D(dy)$, as swapping y and z will not change the sum dy. Each element of type (y, y_2, y_3, y_4) or (z, y_2, y_3, y_4) will change its position in sets $D(dy)$: If $y \oplus y_2 \oplus y_3 \oplus y_4 = dy$, than $z \oplus y_2 \oplus y_3 \oplus y_4 = dy \oplus (y \oplus z)$. Thus, all elements of this type will move from set $D(0)$ to set $D(y \oplus z)$, decreasing the size of new $D(0)$. On the other hand, such elements from $D(y \oplus z)$ will move to $D(0)$, increasing its size. By summing all such contributions (with a minus sign for elements in $D(0)$ and a plus sign for elements in $D(y \oplus z)$), we can assign a score (expected change of $|D(0)|$) to each pair (y, z).

Using the greedy approach, we first select such pair with the lowest score, if the score is less than 0. If there is no pair (y, z) with a negative score, we end the algorithm to ensure that it stops.

The method can be generalized in multiple ways. One generalization is to apply multiple transpositions at once (e.g., all that produce a negative score). In such a case, however, we cannot predict the change of $D(0)$ exactly because of the interactions between contributions of elements containing values from multiple transposition pairs. The preliminary experiments with multiple transpositions were unsuccessful.

The second generalization is to replace transpositions with more general permutations. We have used small cycles of sizes 3 and 4. In the experiments, we computed the estimated score for all possible cycles of a given size. This means that the complexity quickly grows with the cycle size and becomes impractical for longer cycles. Note that in case of more cycles, we have computed the score just as the unidirectional contributions of individual elements in the cycle (e.g., $a \leftarrow b$, then $b \leftarrow c$, then $c \leftarrow a$). This score is then only an estimate, as it ignores the effects of elements that contain multiple elements from the cycles. Even if the estimated change of $D(0)$ is negative, sometimes the contribution from the shared elements causes the $D(0)$ in the next step to grow. In such a case we again terminate the algorithm to ensure that it stops.

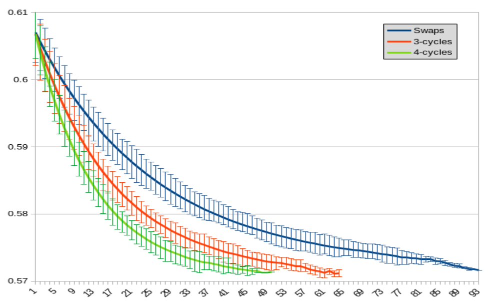

Fig. 1 Average fraction of zeroes (y) in the DDT (of 100 randomly generated S-boxes) after x iterations of the algorithm.

3 Results and Discussion

We have conducted 3 experiments with custom software implementing three methods from Section 2: swap, 3-cycles, and 4-cycles. For each experiment, we generated 100 random bijective 8-bit S-boxes. In the initial set, the S-boxes had differential uniformity between 10 and 16, with 60.7% of zeroes in the DDT. The "smoothing algorithm" improved the DDT of S-boxes gradually (see Figure 1), reaching a minimum of 57.1% of zeroes in the DDT (consistently for the 3 different methods). The final differential uniformity was between 8 and 10, the 3-cycle method produced two S-boxes with $\delta = 6$ (but not as a final step), and the 4-cycle method produced 1 S-box with $\delta = 6$ (in a final step).

While the number of steps depends on the chosen method, the expected number of changes in the S-box table, and the final results seem independent of the method chosen. From the computational perspective, it is better to implement only the "swap" method, which exchanges two values at a time, and each iteration is faster than the other methods.

An interesting observation is that the DDT-smoothing method also improved the non-linearity of the S-boxes. From the initial values between 86 and 96, we have reached S-boxes with non-linearity between 98 and 102. Interestingly, these values are comparable to results of advanced evolutionary techniques (see [2]) while using only a simple algorithm focusing on a completely different S-box characteristic.

Acknowledgments. This research was supported in part by the NATO Science for Peace and Security Programme under Project G5985 and in part by the Slovak Scientific Grant Agency, Grant Number VEGA 1/0105/23.

References

[1] Marochok, S., Zajac, P.: Algorithm for generating s-boxes with prescribed differential properties. Algorithms **16**(3), 157 (2023)

[2] Picek, S., Cupic, M., Rotim, L.: A new cost function for evolution of s-boxes. Evolutionary computation **24**(4), 695–718 (2016)