

An Analysis of Component Functions: Replicating Kuznyechik's S-Box Properties Through Heuristic Methods

**XIV Workshop on Current Trends in Cryptology
CTCrypt 2025**

June 5, 2025

S-Box design rationale

- S-Boxes are fundamental components in the design of symmetric cryptographic primitives.
- Essential cryptographic properties include low linearity, low differential uniformity and a high algebraic degree.
- Desirable characteristics include good implementation properties, such as efficient hardware and bit-slice implementations, as well as compactness to minimize memory footprint.

S-Box design rationale

- How were the S-Boxes generated?
- Do they adhere to a specific algebraic structure, such as finite field inversion?
- Are they constructed using a Feistel Network, SPN, or Lai-Massey structure?
- Was random generation employed? If so, which seed was used?
- What specific cryptographic properties were optimized, and how was this optimization performed?

As reported by Saarinen and Brumley¹ following discussions at CTRcrypt 2014, the designers described their principle as iteratively generating “*randomized*” S-Boxes using various “*building blocks*” until a permutation satisfied baseline differential, linear, and algebraic criteria. This approach aimed to provide resilience against potential future attacks and known algebraic techniques.

¹Saarinen, M.J.O. and Brumley, B.B. “*WHIRLBOB, the Whirlpool Based Variant of STRIBOB*”, Proceedings of Secure IT Systems: 20th Nordic Conference, 2015

Kuznyechik S-box decomposition

- Biryukov et al. conducted a reverse-engineering analysis which resulted in the proposal of the TU-decomposition, a structural representation of the S-Box and its components.
- Subsequent work by Perrin and Udovenko investigated the S-Box's potential origins, suggesting it derives from a variant of exponential substitution, further generalized as pseudo-exponential substitution.
- Finally, the TKLog construction, introduced by Perrin, attempted to unify the structural findings presented in the previous works.

Arguments against the randomized generation of the S-Box

- (1) Its differential and linear properties deviate from those expected of a uniformly random permutation.
- (2) The existence of structural decompositions replicating its input/output behavior.

Arguments against the randomized generation of the S-Box

- (1) Its differential and linear properties deviate from those expected of a uniformly random permutation.

Achievable through heuristic randomized methods that optimize these very characteristics

- (2) The existence of structural decompositions replicating its input/output behavior.

Arguments against the randomized generation of the S-Box

- (1) Its differential and linear properties deviate from those expected of a uniformly random permutation.

Achievable through heuristic randomized methods that optimize these very characteristics

- (2) The existence of structural decompositions replicating its input/output behavior.

The existence of such structural representations does not preclude a randomized generation process, nor does it confirm intentional design based on that structure.

No further reverse-engineering, ask the right questions

- What underlying mathematical properties of Boolean functions or S-Boxes explain the observed patterns in the Kuznyechik S-Box's LAT, and are these properties unique to this S-Box?

No further reverse-engineering, ask the right questions

- What underlying mathematical properties of Boolean functions or S-Boxes explain the observed patterns in the Kuznyechik S-Box's LAT, and are these properties unique to this S-Box?
- Could these patterns stem from a mathematical construction or structure distinct from those previously proposed?

No further reverse-engineering, ask the right questions

- What underlying mathematical properties of Boolean functions or S-Boxes explain the observed patterns in the Kuznyechik S-Box's LAT, and are these properties unique to this S-Box?
- Could these patterns stem from a mathematical construction or structure distinct from those previously proposed?
- Is it plausible that an S-Box generation method, consistent with the designers' arguments involving iterated randomized "building blocks", could inherently result in the TU-decomposition?

The pattern identified in Kuznyechik's S-Box

According to Biryukov et. al., the stripes observed in the Linear Approximation Table (LAT) of the affine S-Box π_{KUZ} correspond to linear approximations of the form $(a_\ell || a_r \rightsquigarrow b_\ell || 0)$ where $b_\ell > 0$. These functions correspond to the output masks γ in the set $\mathcal{H}$:

$$\mathcal{H} = \left\{ \begin{array}{ccccc} 0x1a, & 0x20, & 0x3a, & 0x44, & 0x5e, \\ 0x64, & 0x7e, & 0x8a, & 0x90, & 0xaa, \\ 0xb0, & 0xce, & 0xd4, & 0xee, & 0xf4 \end{array} \right\}.$$

Notably, the authors include 0x00 in this set, making $\mathcal{H}$ a vector subspace of $\mathbb{F}_2^8$ which is denoted in what follows by $\mathcal{H}'$, i.e., $\mathcal{H}' = \mathcal{H} \cup \{0x00\}$.

What these functions have in common?

Since our analysis focuses on the linear spectrum of the S-Box, a natural approach is to examine the extreme values within the LAT, which determine the nonlinearity of the corresponding component functions f_γ .

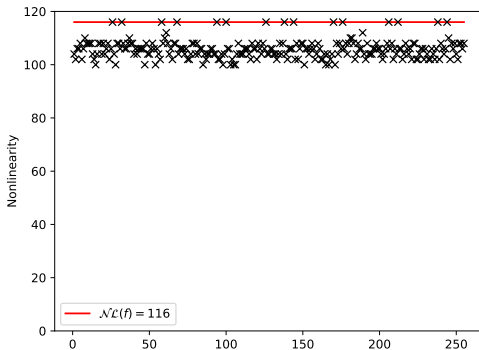


Figure: Nonlinearity of component functions of π_{KUZ} .

Simulating high nonlinearity component functions

STEP 1: Select target component Boolean functions.

STEP 2: Apply an optimization method to a random bijective S-Box, optimizing only the nonlinearity of the targeted functions.

STEP 3: Generate the Jackson Pollock representation of the resulting permutation and examine for vertical stripes in columns corresponding to optimized components.

Simulating high nonlinearity component functions

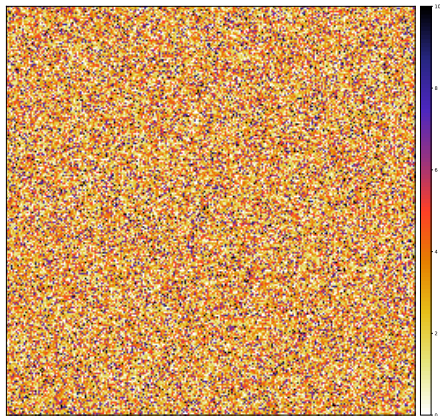


Figure: Jackson Pollock representation of the LAT of one randomly generated S-Box where some of its component functions were optimized for high nonlinearity.

Nonlinearity can't cause the stripes by itself

All targeted functions achieved nonlinearity 116. However, the LAT representation reveals no discernible patterns, contradicting the initial hypothesis. This leads to the following observation:

Observation

High nonlinearity of component Boolean functions does not guarantee stripe patterns in the LAT visualization. The stripes in π_{KUZ} might arise from different or combined factors.

What else do these functions have in common?

Besides nonlinearity, component Boolean functions f_γ , also share the same curvature value as shown by de la Cruz ²:

$$\text{curv}(f_\gamma) = 2^{\frac{3n}{2}} - 2^n = 3840, \quad n = 8.$$

Following our previous observation, we investigated curvature's influence. We modified the experiment to optimize for high curvature the component functions identified by $\{0x1f, 0x20\}$ of a randomly generated S-Box.

²de la Cruz Jiménez, R. A., "On some properties of the curvature and nondegeneracy of Boolean functions", Mat. Vopr. Kriptogr., Vol. 13(2), 2022.

Simulating high curvature component functions

After the optimization process:

- The targeted component functions had nonlinearity 110 and curvatures 3824 and 3784 respectively.

Simulating high curvature component functions

After the optimization process:

- The targeted component functions had nonlinearity 110 and curvatures 3824 and 3784 respectively.
- While nonlinearities were lower than for f_γ or the S-Box from the first figure, curvatures were high (close to these of f_γ) and significantly higher than the S-Box optimized for nonlinearity.

Simulating high curvature component functions

After the optimization process:

- The targeted component functions had nonlinearity 110 and curvatures 3824 and 3784 respectively.
- While nonlinearities were lower than for f_γ or the S-Box from the first figure, curvatures were high (close to these of f_γ) and significantly higher than the S-Box optimized for nonlinearity.
- The Jackson Pollock representation of the LAT of the S-Box shows evidence of the stripe pattern.

Simulating high curvature component functions

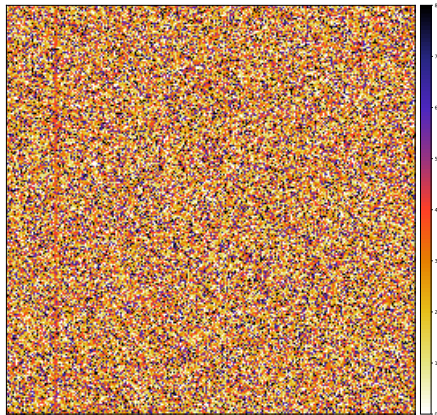


Figure: Jackson Pollock representation of the LAT of one randomly generated S-Box where some of its component functions were optimized for high curvature.

Replicating our results in Kuznyechik's S-Box

To test this on π_{KUZ} , we used its TU-decomposition and replaced the component ν_0 by the identity permutation $\mathcal{ID}(x) = x$, $\forall x \in \mathbb{F}_2^4$. For the resulting S-Box (π'_{KUZ}), functions $f'_\gamma(x)$ exhibited nonlinearity equal 112 and $\text{curv}(f) = 3840$.

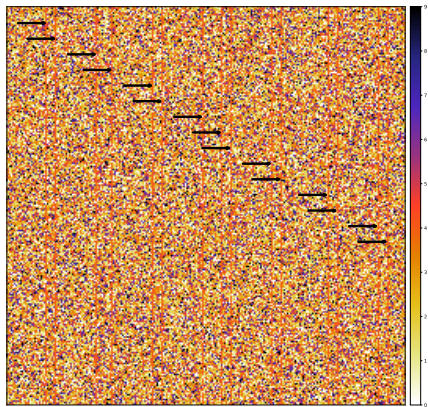


Figure: LAT of π'_{KUZ} (π_{KUZ} with ν_0 replaced by the identity permutation).

Replicating our results in Kuznyechik's S-Box

As the previous figure confirms, the stripe patterns persist in π'_{KUZ} . This supports two claims:

- The nonlinearity of ν_0 in the TU-decomposition is not critical for π_{KUZ} 's LAT stripes.
- High curvature values correlate with vertical stripes in the LAT visualization.

Replicating our results in Kuznyechik's S-Box

The visibility of the vertical stripe pattern is related to the distribution of coefficients in the Walsh spectrum of the Boolean functions with values close to magnitude 16.

S-Box / Coeff. (Abs. Val.)	0	4	8	12	16	20	24	28	32
π_{KUZ}	16	0	16	64	80	64	16	0	0
π'_{KUZ}	24	0	0	0	224	0	0	0	8

Table: Frequencies of absolute Walsh coefficient values for $f_\gamma(x)$, $f'_\gamma(x)$.

Replicating our results in Kuznyechik's S-Box

The visibility of the vertical stripe pattern is related to the distribution of coefficients in the Walsh spectrum of the Boolean functions with values close to magnitude 16.

S-Box / Coeff. (Abs. Val.)	0	4	8	12	16	20	24	28	32
π_{KUZ}	16	0	16	64	80	64	16	0	0
π'_{KUZ}	24	0	0	0	224	0	0	0	8

Table: Frequencies of absolute Walsh coefficient values for $f_{\gamma}(x)$, $f'_{\gamma}(x)$.

Is there any family of Boolean functions whose nonlinearity and curvature behave like the functions we've analyzed from Kuznyechik's S-Box?

A class of Boolean functions related to π_{KUZ} components

Define $n = 2k$ and consider an arbitrary k -variables Boolean function g and a normal $2k$ -variables bent function $\varphi(x, y)$. Then, we construct $f_{\varphi, g}$ as follows

$$f_{\varphi, g} = \begin{cases} \varphi(x, y), & \text{if } x \neq x_0; \\ g(y), & \text{if } x = x_0. \end{cases} \quad (1)$$

This construction, studied by Kamlovskiy ³, generalizes the particular case investigated by de la Cruz ⁴.

³Kamlovskiy, O.V. "The sum of modules of Walsh coefficients of some balanced Boolean functions", Mat. Vopr. Kriptogr., Vol. 8(4), 2017

⁴de la Cruz Jiménez, R. A., "On some properties of the curvature and nondegeneracy of Boolean functions", Mat. Vopr. Kriptogr., Vol. 13(2), 2022. 

A class of Boolean functions related to π_{KUZ} components

- If g is a balanced function, the curvature behaves as $\text{curv}(f_{\varphi,g}) \sim 2^{3n/2}$ when $n \rightarrow \infty$.
- The functions $f_{\varphi,g}$ exhibit high nonlinearity, approaching the theoretical maximum bounds.
- This construction provides the potential to achieve maximal algebraic degree.

A class of Boolean functions related to π_{KUZ} components

- If g is a balanced function, the curvature behaves as $\text{curv}(f_{\varphi,g}) \sim 2^{3n/2}$ when $n \rightarrow \infty$.
- The functions $f_{\varphi,g}$ exhibit high nonlinearity, approaching the theoretical maximum bounds.
- This construction provides the potential to achieve maximal algebraic degree.

These combined properties naturally lead to the question of whether the targeted component Boolean functions of Kuznyechik's S-Box can themselves be obtained from a Dobbertins' construction.

Proposition

For any $f_\gamma(x) = \langle \gamma, \pi_{\text{KUZ}} \rangle, \gamma \in \mathcal{H}$ there exist a Boolean function h obtained from construction (1) such that f_γ and h are linear-equivalent.

A class of Boolean functions related to π_{KUZ} components

Proposition

For any $f_\gamma(x) = \langle \gamma, \pi_{\text{KUZ}} \rangle, \gamma \in \mathcal{H}$ there exist a Boolean function h obtained from construction (1) such that f_γ and h are linear-equivalent.

What family of normal bent functions is used within the Dobbertins' construction?

An equivalent S-Box having coordinate functions with a well-defined mathematical structure

Let $\mathcal{A}$ be a linear mapping defined by:

$$\begin{array}{llll} \mathcal{A}(01) = \gamma_1 & \mathcal{A}(02) = \gamma_2 & \mathcal{A}(04) = \gamma_3 & \mathcal{A}(08) = \gamma_4 \\ \mathcal{A}(10) = 10 & \mathcal{A}(20) = 20 & \mathcal{A}(40) = 40 & \mathcal{A}(80) = 80 \end{array}$$

with $\mathcal{B} = \{\gamma_i \in \mathcal{H} \mid i = 1, 2, 3, 4\}$ being a basis for the subspace $\mathcal{H}'$

An equivalent S-Box having coordinate functions with a well-defined mathematical structure

Let $\mathcal{A}$ be a linear mapping defined by:

$$\begin{array}{llll} \mathcal{A}(01) = \gamma_1 & \mathcal{A}(02) = \gamma_2 & \mathcal{A}(04) = \gamma_3 & \mathcal{A}(08) = \gamma_4 \\ \mathcal{A}(10) = 10 & \mathcal{A}(20) = 20 & \mathcal{A}(40) = 40 & \mathcal{A}(80) = 80 \end{array}$$

with $\mathcal{B} = \{\gamma_i \in \mathcal{H} \mid i = 1, 2, 3, 4\}$ being a basis for the subspace $\mathcal{H}'$

The permutation $\tilde{\pi}_{\text{KUZ}} = \mathcal{A} \circ \pi_{\text{KUZ}}$ has its four least significant coordinate functions given by $f_i(x) = \langle \gamma_i, \pi_{\text{KUZ}}(x) \rangle$, where $\gamma_i \in \mathcal{B}$ and these functions are linearly equivalent to functions derived from the Dobbertins construction.

On the form of bent functions in the Dobbertin construction

Proposition

Let $\mathcal{B} = \{0x20, 0x5e, 0x8a, 0xd4\}$ be a basis for the vector subspace $\mathcal{H}'$. Any bent function φ_γ , defined as

$$\varphi_\gamma(x) = \begin{cases} h_\gamma(x), & \text{if } x \notin W \\ \epsilon, & \text{if } x \in W \end{cases}$$

where $h_\gamma(x) = \langle \gamma, \pi_{\text{KUZ}}(x) \rangle$, $\gamma \in \mathcal{B}$, $\epsilon \in \mathbb{F}_2$ and $W = \{(\ell, 0) : \ell \in \mathbb{F}_2^4\}$, is linearly equivalent to a function g_γ that satisfies one of the following conditions:

- $g_\gamma \in \mathcal{PS}_{ap}$;
- $\tilde{g}_\gamma = g_\gamma \oplus 1 \in \mathcal{PS}_{ap}$.

Similarities to the TU-Decomposition of π_{KUZ}

The four most significant coordinates of the nonlinear core of π_{KUZ} can be represented as follows:

$$\ell_{out} = \begin{cases} \nu_1(\frac{\ell}{r}), & \text{if } r \neq \mathbf{0}; \\ \nu_0(\ell), & \text{if } r = \mathbf{0}. \end{cases}$$

Setting $\ell_{out} = (l_0, l_1, l_2, l_3)$, each coordinate function l_i can be expressed as:

$$l_i = f_i(\ell, r) = \begin{cases} \nu_{1,i}(\frac{\ell}{r}), & \text{if } r \neq \mathbf{0}; \\ \nu_{0,i}(\ell), & \text{if } r = \mathbf{0}. \end{cases}$$

where $\nu_{0,i}$ and $\nu_{1,i}$ represent the i -th coordinate functions of the permutations ν_0 and ν_1 , respectively. This representation is closely related to the Dobbertin construction.

Consider the function φ_i defined as:

$$\varphi_i(\ell, \phi(r)) = \begin{cases} \nu_{1,i}(\frac{\ell + \psi(r)}{r}), & \text{if } r \neq \mathbf{0}; \\ 0, & \text{if } r = \mathbf{0}. \end{cases}$$

This function φ_i , where ψ is the zero constant function ($\psi(r) = 0$ for all r) and ϕ is the identity permutation ($\phi(r) = r$), is a normal bent function. This follows from Dobbertins' triple construction, given that $\nu_{0,i}$ is balanced.

Similarities to the TU-Decomposition of π_{KUZ}

Observe that if $\nu_{1,i}(\mathbf{0}) = 0$ (i.e., $\nu_{1,i}$ vanishes at $\mathbf{0}$), then the function φ_i constructed above belongs to the $\mathcal{PS}_{ap}$ class.

Conversely, if $\nu_{1,i}(0) = 1$, then φ_i can be expressed as:

$$\varphi_i(\ell, r) = \begin{cases} \nu_{1,i}(\frac{\ell}{r}) \oplus 1, & \text{if } r \neq \mathbf{0}; \\ 1, & \text{if } r = \mathbf{0}. \end{cases}$$

In this case, φ_i is the complement of a function in the $\mathcal{PS}_{ap}$ class, and therefore belongs to the $\mathcal{PS}^+$ class.

A quick summary

The curvature of Boolean functions is related to the appearance of vertical stripe patterns in the visual representation of the LAT of S-Boxes.

A quick summary

The curvature of Boolean functions is related to the appearance of vertical stripe patterns in the visual representation of the LAT of S-Boxes.

The set of component functions of the Kuznyechik S-Box involved in the stripe patterns are the result of a Dobbertins' construction.

A quick summary

The curvature of Boolean functions is related to the appearance of vertical stripe patterns in the visual representation of the LAT of S-Boxes.

The set of component functions of the Kuznyechik S-Box involved in the stripe patterns are the result of a Dobbertins' construction.

Some of these functions are obtained using the partial spread class as normal bent functions within the Dobbertins' construction.

What is next?

Can randomized building blocks yield the TU-decomposition?

What is next?

Can randomized building blocks yield the TU-decomposition?

Would it yield S-Boxes emulating the cryptographic properties of Kuznyechik? If so, would be possible to obtain such S-Boxes using knowledge from literature survey prior to the publication of the algorithm?

A three phase randomized algorithm

Phase 1: Construct a balanced mapping $F_4 : \mathbb{F}_2^8 \rightarrow \mathbb{F}_2^4$,
 $F = (f_0, f_1, f_2, f_3)$, where the coordinate functions $f_i : \mathbb{F}_2^8 \rightarrow \mathbb{F}_2$
($i = 0, 1, 2, 3$) derive from Dobbertin's construction.

Phase 2: Extend F_4 to a bijection $G : \mathbb{F}_2^8 \rightarrow \mathbb{F}_2^8$ using random
generated permutations on $\mathbb{F}_2^4$. Evaluate the differential
characteristic of the resulting permutation.

Phase 3: Employ a modified Fisher-Yates shuffle as a heuristic
optimization algorithm to improve the linear properties of G .
We then evaluate the remaining relevant cryptographic
parameters (linearity, the minimum degree and the graph
algebraic immunity) of the resulting permutation.

Constructing the balanced mapping $F_4 : \mathbb{F}_2^8 \rightarrow \mathbb{F}_2^4$

- Within each main iteration, it seeks a coordinate function that maintains or progresses towards the desired balanced property for the partially constructed mapping (related to $\mathbb{F}_2^{i+1}$, $i = 0, 1, 2, 3$).

Constructing the balanced mapping $F_4 : \mathbb{F}_2^8 \rightarrow \mathbb{F}_2^4$

- Within each main iteration, it seeks a coordinate function that maintains or progresses towards the desired balanced property for the partially constructed mapping (related to $\mathbb{F}_2^{i+1}$, $i = 0, 1, 2, 3$).
- To avoid excessive computation time and potential bias from suboptimal earlier choices, a restart condition is triggered if the inner loop fails to find a suitable function within one million iterations.

Constructing the balanced mapping $F_4 : \mathbb{F}_2^8 \rightarrow \mathbb{F}_2^4$

- Within each main iteration, it seeks a coordinate function that maintains or progresses towards the desired balanced property for the partially constructed mapping (related to $\mathbb{F}_2^{i+1}$, $i = 0, 1, 2, 3$).
- To avoid excessive computation time and potential bias from suboptimal earlier choices, a restart condition is triggered if the inner loop fails to find a suitable function within one million iterations.
- Upon triggering, the entire construction restarts, discarding all previously selected coordinate functions.

Constructing the balanced mapping $F_4 : \mathbb{F}_2^8 \rightarrow \mathbb{F}_2^4$

- Within each main iteration, it seeks a coordinate function that maintains or progresses towards the desired balanced property for the partially constructed mapping (related to $\mathbb{F}_2^{i+1}$, $i = 0, 1, 2, 3$).
- To avoid excessive computation time and potential bias from suboptimal earlier choices, a restart condition is triggered if the inner loop fails to find a suitable function within one million iterations.
- Upon triggering, the entire construction restarts, discarding all previously selected coordinate functions.
- If a valid mapping is found, the process advances to the second constructive phase of our proposal.

Generating a bijective mapping from $\mathbb{F}_2^8$ to itself

Given a balanced $(8, 4)$ -mapping G obtained in the previous stage this phase ensures the output is a bijective mapping iterating through all values of $\mathbb{F}_2^4$ as the most significant bits for each unique value of the least significant bits (defined by G), thus defining a complete output set for the desired S-Box F .

Generating a bijective mapping from $\mathbb{F}_2^8$ to itself

Given a balanced $(8, 4)$ -mapping G obtained in the previous stage this phase ensures the output is a bijective mapping iterating through all values of $\mathbb{F}_2^4$ as the most significant bits for each unique value of the least significant bits (defined by G), thus defining a complete output set for the desired S-Box F .

Proposition

S-Boxes generated in Phase 2 have TU-decomposition.

Generating a bijective mapping from $\mathbb{F}_2^8$ to itself

Given a balanced $(8, 4)$ -mapping G obtained in the previous stage this phase ensures the output is a bijective mapping iterating through all values of $\mathbb{F}_2^4$ as the most significant bits for each unique value of the least significant bits (defined by G), thus defining a complete output set for the desired S-Box F .

Proposition

S-Boxes generated in Phase 2 have TU-decomposition.

One may tweak this phase to check for good differential characteristics of the resulting permutation.

Optimizing cryptographic the properties of the permutations

While iterating through stages 1 and 2 ensures that the resulting permutations can exhibit differential properties comparable to π_{KUZ} and possess a TU-decomposition, their linear and algebraic properties remain unassessed.



Modified Fisher-Yates shuffle

Optimizing cryptographic the properties of the permutation

- Analogous to the standard Fisher-Yates method, our algorithm iteratively swaps pairs of S-Box output values.
- A key distinction is that following each swap, the linear and algebraic characteristics of the modified S-Box are evaluated against the target design criteria.

Optimizing cryptographic the properties of the permutation

- Analogous to the standard Fisher-Yates method, our algorithm iteratively swaps pairs of S-Box output values.
- A key distinction is that following each swap, the linear and algebraic characteristics of the modified S-Box are evaluated against the target design criteria.
- Importantly, the selection of values to swap is not arbitrary; uncontrolled swapping could invalidate the S-Box's TU-decomposition.

Optimizing cryptographic the properties of the permutation

- Analogous to the standard Fisher-Yates method, our algorithm iteratively swaps pairs of S-Box output values.
- A key distinction is that following each swap, the linear and algebraic characteristics of the modified S-Box are evaluated against the target design criteria.
- **Importantly, the selection of values to swap is not arbitrary; uncontrolled swapping could invalidate the S-Box's TU-decomposition.**

Proposition (TU-preserving transposition)

Let $F : \mathbb{F}_2^8 \rightarrow \mathbb{F}_2^8$ be a bijective S-Box generated by Algorithm 2. Let (x_1, x_2) denote the transposition swapping two distinct elements $x_1, x_2 \in \mathbb{F}_2^8$. If $\text{LSB}_4(F(x_1)) = \text{LSB}_4(F(x_2))$, then the permutation $F' = (x_1, x_2) \cdot F$ also possesses a TU-decomposition.

Modified Fisher-Yates method

- The method receives an initial permutation $F : \mathbb{F}_2^8 \rightarrow \mathbb{F}_2^8$ from Phase 2 and maintains a record of the 'best' S-Box found so far based on its characteristics, initially setting this best candidate to F .
- In addition to the nonlinearity parameter, we compute a metric describing the quality of the linear spectrum ⁵.
- For each potential input x starting from the largest value down to 1 (using the integer representation of vectors in $\mathbb{F}_2^8$), an inner loop considers all preceding inputs y (from $x - 1$ down to 0) to identify pairs (x, y) that can form a TU-preserving transposition with respect to the current state of the S-Box

⁵This metric is obtained using the cost function employed by Clark et al. in 2004, adhering to the original parameter selection $R = 3, X = 4$.

Modified Fisher-Yates method

- For each such identified pair (x, y) , the algorithm generates a new candidate S-Box F' by swapping the outputs $F(x)$ and $F(y)$.
- Then it checks if F' satisfies the desired cryptographic criteria. If the criteria are met, the algorithm successfully terminates and returns F' .
- Otherwise, it checks if F' possesses better characteristics (according to our metrics) than the current best S-Box recorded. If so, F' replaces the current best.
- The algorithm then proceeds to the next potential TU-preserving transposition pair.

Modified Fisher-Yates method

An important aspect is that within the processing for a single outer loop value x , the changes made to the S-Box via the inner loop are cumulative. That is, the S-Box used to check the TU-preserving condition for a pair (x, y_j) might already incorporate swaps from previous pairs (x, y_k) where $k > j$.

Cryptographic Properties of Generated S-Boxes			
$\delta/\mathcal{L}$	$\frac{30}{128}$	$\frac{28}{128}$	$\frac{26}{128}$
$\frac{8}{256}$	236	4760	4
$AI_{gr}/d_{\min}$	5	6	7
3	164	3413	1423
Passed all criteria: 1373			

Table: Distribution of cryptographic properties for 5000 S-Boxes generated by our proposal using partial-spread-based Dobbertins' construction.

Further extending our investigation, we've examined the performance of our algorithm when replacing the initial partial spread class with the alternative class of normal Boolean functions introduced by de la Cruz. ⁶

- We modify Phase 1 of our proposal to construct the balanced $(8, 4)$ -mapping using this class of functions instead of partial spreads.
- Remove any constraint to differential uniformity to the output of Phase 2.
- Relaxing the conditions on differential uniformity to enter the update stage in Phase 3 from requiring $\delta(F') \leq 8/256$ to $\delta(F') \leq \delta(F)$.

⁶de la Cruz Jiménez, R. A., "On some properties of the curvature and nondegeneracy of Boolean functions", Mat. Vopr. Kriptogr., Vol. 13(2), 2022.

Experimental results

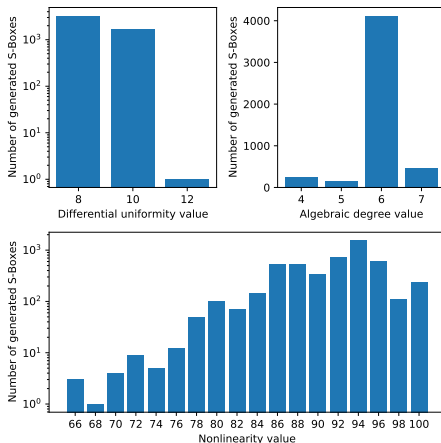


Figure: Frequency distribution of S-Boxes generated using the bent function from de la Cruz as base of the Dobbertin's construction. Here, 45 of the generated S-Boxes fulfilled all cryptographic requirements.

Based on the modifications made to Phase 2 and 3 we implemented an enhanced search strategy focused on optimizing S-Box properties. This involved two key changes:

- Replacing the initial cost function from Clark et. al. with the more effective function from Freyre et. al.^a for (non)linearity optimization;
- Setting a more demanding target linearity: 26/128.

^aFreyre-Echevarría A., et. al. "*An External Parameter Independent Novel Cost Function for Evolving Bijective Substitution-Boxes*", Symmetry, Vol. 12(11), 2020.

Employing this strategy with the partial spread-based Phase 1 resulted in the generation of several S-Boxes that satisfy the desired TU-decomposition property. These S-Boxes demonstrate improved nonlinearity compared to the Kuznyechik S-Box, while maintaining similar differential uniformity and algebraic degree.

CONCLUSIONS

High Curvature as the Determinant of LAT Patterns: We establish conclusively that the prominent vertical stripe patterns observed in the Jackson Pollock representation of the LAT of π_{KUZ} are a direct consequence of the high *curvature* exhibited by a specific subset of its component Boolean functions.

Identification of Component Functions via Dobbertin

Construction and Partial Spreads: We formally prove that the component functions responsible for the LAT patterns are linearly equivalent to functions derived from Dobbertin's construction. Furthermore, the analysis pinpoints the underlying normal bent functions as belonging to the completed affine partial spread class ($\mathcal{PS}_{ap}$) or their complements.

Randomized Generation Yielding TU-Decomposition: A novel three-phase algorithm is presented, which constructs 8-bit S-Boxes using randomized building blocks explicitly derived from Dobbertin functions based on $\mathcal{PS}_{ap}$.

- The algorithm's design inherently guarantees that all generated S-Boxes possess a TU-decomposition.
- Experimental validation show that this method successfully produces S-Boxes replicating Kuznyechik's essential cryptographic parameters ($\mathcal{L}$, δ , d_{min} , Al_{gr}) and exhibiting the characteristic LAT stripe patterns.

This serves as a valuable study case which show that a randomized generation process, guided by specific structural building blocks, can indeed result in the TU-decomposition as an emergent property, lending credence to the plausibility of the designers' arguments given the cryptographic tools available prior to 2014.

THANKS FOR YOUR ATTENTION