Reverse-Engineering the S-Box of Kuznyechik – The Consequences

Ole André Hauge
Department of Information Security and Communication Technology
Norwegian University of Science and Technology (NTNU)
Gjøvik, Norway
April 25, 2022

***Abstract*—The Russian Federation published the standard for the Kuznyechik 128-bit block cipher in March 2016. Its S-box, π, was reverse-engineered by Alex Biryukov, Léo Perrin, and Aleksei Udovenko and presented in 2016. This accomplishment was possible due to the previous work in cryptography conducted by the researchers, as they heavily relied on it during the process. By applying Pollock's Pattern Recognition method they were able to use the visual patterns in the LAT of π to exploit the linear whitening layers to recover the hidden structures and decompose the S-box. We found that this decomposition has mentionable consequences for π related to hardware implementation, a suspicion of the structure having a backdoor, as well as further analysis of the S-box and development of more attacks. The authors are not convinced that the S-box uses the best cryptographic properties. To that extent, they discouraged the standardization of Kuznyechik until the designers clearly explain why these findings are beneficial.**

***Index Terms*—Reverse-engineering, S-box, consequences, Kuznyechik, Linear Approximation Table, Feistel Network, Pollock's Pattern Recognition**

I. Introduction

The Russian Federation published a standard for GOST R34.12-2015, commonly known as Kuznyechik in March 2016. Kuznyechik is a 128-bits block cipher with a 256-bits key. It is meant to be used as a cryptographic technique for information processing and protection in computer-aided systems, including the provision of confidentiality, authenticity, and integrity of information during transmission, processing, and storage [1]. Despite the published standard, little is known about the architectural logic of the cipher's S-box. What we can learn from the standard is that it runs on 8 bits, is a permutation, and should have good differential linear characteristics. The designers only provide the minimum requirement for implementing the S-box, the lookup table. Testing shows that they are slightly better than what you would anticipate from an 8-bit random permutation, but we do not know how they were created [2].

When addressed by other researchers, the designers frequently provided evasive answers and reasons for the design. At one point, the designers claimed that they had lost the generation method, but that they did not worry because, at the time they built it, people did not care where the parameters originated from. However, to have the standard accepted as an ISO standard the designers released additional information claiming that the S-box was chosen from known classes as it should provide close to optimal values of some cryptographic parameters such as differentiality and linearity, an obvious analytic structure, and could use finite field inversion. This is comparable to how they are produced in the AES, and it is a perfectly valid method of selecting S-boxes. Other sources say that they employ a random search with a set parameter limit to choose the S-box with the best possible linear and differential qualities. Such S-boxes do not have clear analytic structures, which is why designers frequently employ them. There is nothing wrong with it, and based on the information provided, it appears that the latter choice was utilized for Kuznyechik [2].

As a consequence of the designers' inconsistent information and unclear remarks, researchers from the University of Luxembourg, Alex Biryukov, Léo Perrin[1], and Aleksei Udovenko[2], were entrusted with attempting to reverse-engineer Kuznyechik's S-box to disclose its underlying structure. Because of their research, they were able to reverse-engineer the S-box and uncover its hidden structure. They discovered that the S-box is based on a two-round Feistel network with finite field multiplication instead of exclusive-or. This structure is hidden by two different linear layers applied before and after. In a field of size 2^4, the S-box is calculated using five distinct 4-bit S-boxes, a multiplexer, two 8-bit linear permutations, and two finite field multiplications. The part that is a bit humorous, according to Léo Perrin, is that the designers picked the second option above the first because it did not have an obvious analytical structure; nonetheless, the researchers discovered that it had a highly analytical structure.

This paper looks at one of the papers, "Reverse-Engineering the S-Box of Streebog, Kuznyechik, and STRIBOBr1" [3], resulting from their research. We first explain their methodology before identifying the consequences their findings have on the Kuznyechik block cipher. The paper is structured as follows: Section II describes definitions and notations as well as important aspects of cryptanalysis that are needed to understand the following sections. Section III explains the methodology of the paper. Section IV presents the findings of

[1]The work of Léo Perrin was supported by the CORE ACRYPT project (IDC12-15-4009992) funded by the Fonds National de la Recherche (Luxembourg)

[2]The work of Aleksei Udovenko was supported by the Fonds National de la Recherche, Luxembourg (project reference 9037104)

the literature review. Section V discusses the consequences of the decomposition of the S-box and section VI concludes the paper.

II. Background

We begin by listing the operations and notations that will be used throughout the paper, followed by a description of a few cryptographic attacks and analysis methods that will be mentioned later in the text.

Definitions and Notations

The below notations are adopted from the authors' paper [3] to keep the synergy between our papers.
$\boldsymbol{\pi}$ – the notation for the S-box of Kuznyechik.
XOR – denoted exclusive-or, XOR, or by $\oplus$.
DDT – the Difference Distribution Table of a function mapping n bits to m is a $2^n \times 2^m$ matrix.
LAT – the Linear Approximation Table of a function mapping n bits to m is a $2^n \times 2^m$ matrix.
Cardinal – of the differential is the coefficient of a DDT.
Bias – of the approximation is the coefficient of a LAT.
Affine-Equivalence – two vectorial Boolean functions f and g are affine-equivalent if there exists two affine mappings μ and η such that $f = \eta \circ g \circ \mu$.
Whitening layer – a linear transformation of a vector of random variables with a known covariance matrix into a set of new variables whose covariance is the identity matrix, meaning that they are uncorrelated and each has a variance of 1.

Multiset Properties

Multiset properties allow us to characterize intermediate values deep within an encryption structure even if you do not know what the actual functions are. A multiset can be represented by a list of tuples (value, multiplicity). The multiset's size is the total of its multiplicities. Multisets contain the following properties as proved in [4]:

1) A multiset M of m-bit values has property C (constant) if it contains an arbitrary number of repetitions of a single value.
2) A multiset M of m-bit values has property P (permutation) if it contains exactly once each one of the $2m$ possible values.
3) A multiset M of m-bit values has property E (even) if each value occurs an even number of times (including no occurrences at all).
4) A multiset M of m-bit values has property B (balanced) if the XOR of all the values (taken with their multiplicities) is the zero vector.
5) A multiset M of m-bit values has property D (dual) if it has either property P or property E.

SASAS Attack

The SASAS attack employs structural cryptanalysis, which is the study of the security of cryptosystems defined using generic block diagrams. In [4], an attack is detailed in which the attacker's sole knowledge is that the block cipher has this general structure and the variables k and m. It focuses on the five-layer scheme "S3A2S2A1S1", in which each S layer has k invertible S-boxes that map m-bits to m-bits and each A layer contains an invertible affine mapping of $n = k \times m$-bit-vectors over GF(2), as denoted by:

$$A_i(x) = L_i x \oplus B_i$$

As we will see later in this paper, the approach may be adapted for smaller structures.

Distinguishing Attack

Patarin, J explains in his work [5] a method that may be applied to Feistel ciphers to distinguish between random Feistel ciphers and truly random permutations.

Satisfiability Checking (SAT) Attack

An algorithm is used in an SAT-based attack to determine a key value that is compatible with a collection of input and output observations. While it is guaranteed to deliver a key value that gives the right output for the input patterns, this key value may generate incorrect output for other input patterns. Because of this, as well as the problem's computational complexity, multiple attack algorithms for various SAT assaults have been developed [6].

Differential and Linear Cryptanalysis

Differential cryptanalysis is a strong tool for discovering flaws in block ciphers. The first step in assessing whether a given block cipher is vulnerable to differential cryptanalysis is to compute the differential distribution table (DDT) of all S-boxes utilized in the construction of the block cipher. If the DDT of an S-box has a big integer number, it reveals possible vulnerabilities. This indicates that it might be vulnerable to a differential cryptanalysis attack. Therefore, if the DDT of the S-box only contains small integers, it is less likely to be subject to a differential cryptanalysis attack.

The DDT of the S-box, S, is a matrix with 2^n-rows and 2^m-columns. It is denoted by $\mathcal{D}$. The rows are indexed by the elements $\alpha \in \mathbb{F}_2^n = \{0, ..., 2^n - 1\}$. The columns are indexed by the elements $\beta \in \mathbb{F}_2^m = \{0, ..., 2^n - 1\}$. In other words, the values are integers from 0 to the value of $2^n - 1$ and $2^m - 1$ respectively. The entry at row index α and column index β is given by $d_{\alpha,\beta} = |\Delta_{\alpha,\beta}|$, which is the same as the cardinality of the difference set associated with the S-box at the values α and β. Hence the DDT of S is a matrix that contains every possible integer value $d_{\alpha,\beta}$ as α and β range over the row and columns. This is denoted by $\mathcal{D} = (d_{\alpha,\beta})$ (see Figure 1 for an example. NB! This shows an LAT.)

When designing and analyzing attacks by differential cryptanalysis, we often want to translate the integer values from the DDT into probabilities. In probability theory, we have a fixed finite set Ω called "universe" where subsets E of Ω are called "events". The probability p that E occurs is defined to be the quotient of the cardinalities of sets E and Ω. Denoted by $p = \frac{|E|}{|\Omega|}$, which returns a value between 0 to 1.

Input Sum	Output Sum 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	+8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	–2	–2	0	0	–2	+6	+2	+2	0	0	+2	+2	0	0
2	0	0	–2	–2	0	0	–2	–2	0	0	+2	+2	0	0	–6	+2
3	0	0	0	0	0	0	0	0	+2	–6	–2	–2	+2	+2	–2	–2
4	0	+2	0	–2	–2	–4	–2	0	0	–2	0	+2	+2	–4	+2	0
5	0	–2	–2	0	–2	0	+4	+2	–2	0	–4	+2	0	–2	–2	0
6	0	+2	–2	+4	+2	0	0	+2	0	–2	+2	+4	–2	0	0	–2
7	0	–2	0	+2	+2	–4	+2	0	–2	0	+2	0	+4	+2	0	+2
8	0	0	0	0	0	0	0	0	–2	+2	+2	–2	+2	–2	–2	–6
9	0	0	–2	–2	0	0	–2	–2	–4	0	–2	+2	0	+4	+2	–2
A	0	+4	–2	+2	–4	0	+2	–2	+2	+2	0	0	+2	+2	0	0
B	0	+4	0	–4	+4	0	+4	0	0	0	0	0	0	0	0	0
C	0	–2	+4	–2	–2	0	+2	0	+2	0	+2	+4	0	+2	0	–2
D	0	+2	+2	0	–2	+4	0	+2	–4	–2	+2	0	+2	0	0	+2
E	0	+2	+2	0	–2	–4	0	+2	–2	0	0	–2	–4	+2	–2	0
F	0	–2	–4	–2	–2	0	+2	0	0	–2	+4	–2	–2	0	+2	0

Figure 1. Linear Approximation Table
Source: Adapted from [7]

In regards to the DDT, the universe is $\Omega = \Delta_\alpha$, where Δ_α is the difference set, and the event is $E = \Delta_{\alpha,\beta}$. This means that the cardinality of $|\Omega| = |\Delta_\alpha| = 2^n$ and $|E| = |\Delta_{\alpha,\beta}| = d_{\alpha,\beta}$. If $p_{\alpha,\beta}$ is the associated probability then, $p_{\alpha,\beta} = \frac{|\Delta_{\alpha,\beta}|}{|\Delta_\alpha|} = \frac{d_{\alpha,\beta}}{2^n}$. We can regard universe as $\Omega = \mathbb{F}_2^n$ and the event as the set $E = \{x \in \mathbb{F}_2^n : S(x) \oplus S(x \oplus \alpha) = \beta\}$.

With the DDT of an XOR S-box, if we let $k \in \mathbb{F}_2^n$ be fixed we can define the bijective S-box $S : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$, $S(x) = x \oplus k$. Here k represents the key for a key-mixing layer. Note that $S(x_1 \oplus x_2) \neq S(x_1) \oplus S(x_2)$ when $k \neq 0_n$. Therefore when $k \neq 0$ the S-box S is non-linear. The DDT of this S-box is found by considering the equation for S: $S(x) \oplus S(x \oplus \alpha) = \beta$. By the definition of S this is equivalent to: $x \oplus k \oplus x \oplus \alpha \oplus k = \beta$, which is equivalent to $\alpha = \beta$. This gives the following properties of the XOR S-box: $\Delta_{\alpha,\beta} = \begin{cases} \Delta_\alpha \text{ if } \alpha = \beta \\ \emptyset \text{ if } \alpha \neq \beta \end{cases}$ and $d_{\alpha,\beta} = \begin{cases} 2^n \text{ if } \alpha = \beta \\ 0 \text{ if } \alpha \neq \beta \end{cases}$ When $\alpha = \beta$ it means that the entries are located on the main diagonal, so the DDT of this S-box will consist of 2^n on the main diagonal and zeros everywhere else. Since the key-mixing layer is separate from the non-linear layer, we should not be concerned about large values in the DDT of the XOR S-box. The presence of the large 2^n value indicates a cipher that uses this S-box for its confusion layer may be highly vulnerable to an attack by differential cryptanalysis. The XOR key-mixing layer is a critical part of a cipher, but we would not use it in the confusion layer.

An alternative to the DDT is linear cryptanalysis which uses the linear approximation table (LAT) of the S-box. To develop a linear cryptanalysis attack, we first examine the LAT of each S-box utilized in the cipher's design. The LAT is a type of matrix, often denoted as $\mathcal{L}$, similar to that found in DDTs, however, the range of entries is different. The linear approximation of an S-box demonstrates linear relationships between the input and output bits, as seen in Figure **??**. This is accomplished by comparing the various combinations of input and output bits to the linear expression. This tells us the number of agreements that exist between the left and right sides of each equation. The linear probability bias is then computed to determine the attack's success. This is determined by the absolute magnitude of the linear probability bias, which is computed by subtracting 0.5 from the probability p, which is given by the number of agreements. In the context of probability, bias is denoted as $\varepsilon = p - \frac{1}{2}$. If $\varepsilon = 0$, our approximation is as good as random guessing since it is accurate half of the time and the wrong half of the time. If $\varepsilon > 0$, it suggests that it is true more frequently than not, indicating that it is a good inner approximation. If $\varepsilon < 0$, i.e. negative, it is a weak linear approximation; but, if it is XORed with one, it can still be useful.

The LAT is an enumeration of all linear S-box approximations, with each element in the matrix representing the number of matches between the linear equation in the input sum row and the sum of the output bits in the output sum column. A general characteristic of the LAT is that dividing a table entry by 2^n yields the linear probability bias for the specific linear approximation. The LAT specifies the quality of each potential mask: The number of solutions minus all possible values is given by $\text{LAT}[\alpha, \beta] = s - 2^{b-1} = 2^b\varepsilon$. A zero in the matrix indicates that the approximation is valid in half of the situations, and hence we cannot learn anything from it; this is similar to the DDT. Because it is extremely frequently or very seldom right, the greater the entry, the more the linear approximation informs us about the function. These are the entries that will be used in the analysis.

III. Methodology

This paper adheres to the IMRAD [8] structure and is based on qualitative literature reviews in the fields of information technology and security, computer science, and cryptography. The foundation of discussion is based on relevant academic sources found using acknowledged online academic search engines and literary databases, like Google Scholar.

The selection of relevant publications about cryptographic algorithms, ciphers, attacks, and backdoors was made to keep within the scope of this paper, since the length, technical level, and time constraints necessitated that not all elements of the issue could be addressed.

IV. Results

In this section we explain the methods the authors used to reverse-engineer π.

The S-box was decomposed in phases and based on a methodology known as "Pollock's Pattern Recognition." This is a technique, created in [9] by Alex Biryukov and Léo Perrin, that can be used to recover hidden structures in S-boxes. Pollock's Pattern Recognition creates pictures of DDTs and LATs to use the human eye's innate pattern-finding ability to discern non-random qualities. The approach examines the complete DDT/LAT at once by assigning different colors to the coefficient values. The colors may be used to emphasize patterns, as seen in Figure 2, where the Jackson Pollock

depiction of $\mathcal{L}'_\pi$ from their data shows distinct patterns that can be identified by the human eye (here: the vertical lines to the left and white box in the top left corner).

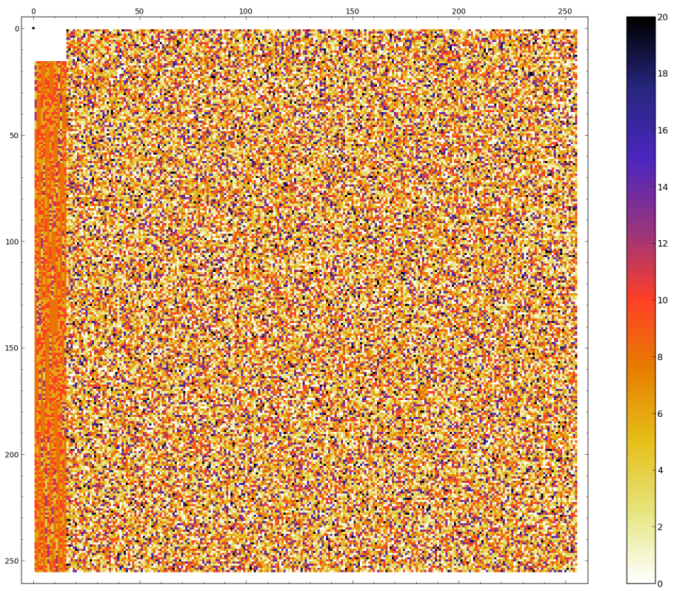

Figure 2. The Jackson Pollock representation of $\mathcal{L}'_\pi$, where $\mathcal{L}'_\pi[i,j] = \mathcal{L}_\pi[L(i), L(j)]$
Source: Adapted from [9]

In the paper [9] they suggest a decomposition strategy to be used with the Pollock's Pattern Recognition method:

1) Draw the "Pollock" visual representation of the LAT and DDT of a generic S-box S.
2) Check whether the linear and differential properties of S are compatible with a random function/permutation.
3) Compute the signature $\sigma(S)$ of S.
4) If $\sigma(S)$ is even, you may:
   a) Try an attack with SASAS.
   b) Try to distinguish S from a Feistel Network with XOR, using the distinguishers in [5].
   c) If one of the Feistel Network distinguishers worked, run $DecomposeFeistel(S, R, \oplus)$ for an appropriate R.
5) Regardless of $\sigma(S)$, run $DecomposeFeistel(S, R, \boxplus)$ for $R \in [2, 5]$.
6) Regardless of $\sigma(S)$, run $BreakArithmetic(S)$.

This is roughly the method employed by the authors, which seems reasonable given that two of the authors created the method in the paper [9]. The following sections describe the steps the authors used to deconstruct the S-box of Kuznyechik.

Recovering Structures

The researchers applied attacks to π in an attempt to recover possible hidden structures. The only thing they were able to uncover was that it has an even signature $\sigma(S)$, which could indicate that the structure is based on a substitution permutation network with simple bit permutations, or it could be based on a Feistel network. With this knowledge they applied attacks known to be effective against such structures, as seen in 4a) and 4b), without success.

The authors rejected the hypothesis that π was an affine-equivalent to a monomial of F_2^8 based on the fact that functions that are affine-equivalent to a monomial have a DDT with all lines corresponding to a non-zero input-difference containing the same coefficients. This resulted directly from the definition of the differential spectrum of monomials (f.ex. the DDT of AES). This is also evident from looking at the Jackson Pollock representation of the DDT of π.

Highly Structured Affine-Equivalent S-Box

The "Jackson Pollock" representations of the DDT and LAT of the S-box were used to visually discern patterns that indicated whether π had good or bad cryptographic properties. The Jackson Pollock depiction, as shown in Figure 2, clearly reveals traces of a highly structured LAT. The authors used linear operations to map the data in the LAT without chaining its attributes to better show the visual patterns, as shown in Figure 2. To do this, they first used an $\oplus$-texture as an auto-correlation approach to highlight the patterns in the LAT and make them more obvious for examination. They then utilized a linear mapping to group the columns of interest to generate a linear permutation of F_2^8, as shown in the figure. A 16×16 white square may be seen in the top left corner (except for the $0, 0$ corner, where the bias is equal to the maximum value). The white hue denotes that the coefficients are all equal to zero. A pattern may also be seen in the 15 leftmost columns, which consist of the lines that were previously spread vertically in the LAT. The deeper color is because all of the values are in the $[4, 12]$ range.

Building a New S-Box

Based on these observations, they constructed another S-box from π with a LAT equal to L'_π. The rationale is that if they can create an S-box with the same qualities as π they can gain an understanding of how the actual S-box was produced. To accomplish this, they find support in the Ph.D. project of Bart Preneel [10] and his Proposition 8.3, where he discusses the impact of transformations on truth tables or the Walsh Fourier transform. They can establish an S-box with $L^t \circ \pi \circ (L^t)^{-1}$ has L'_π as its LAT by adapting his thesis to the LAT table and studying the permutations of the structure. The mapping L^t consists of a linear Feistel round followed by a permutation of the left and right 4-bit nibbles. They simplify the notation by replacing the nibble permutation with the L^* notation: $\pi' = L^* \circ \pi \circ L^*$.

The First Decomposition

The S-box π' is highly structured and has multiset properties that may be used, as discussed in [4] because it has the same affine-equivalent properties as π. The multiset properties allow us to characterize intermediate values deep within an encryption structure even if we do not know what the real functions are. They used SAT attacks on π' and π'^{-1} since

they assumed they had a structure from a lower degree Feistel (less than 5). Despite the failure of the attacks, they proceeded to investigate the multiset properties of the inverse of π'. They observed that the multiset property was even more powerful than π'. By replacing the set's unique element with $(0||0)$, it may be seen as a vector space for any constant, with the right nibble being a linear function of the left nibble. Because the left nibble accepted all possible values, excluding outputs of the form $(?||0)$ let them write π'^{-1} as:

$$\pi'^{-1}(\ell||r) = T_\ell(r)||V_\ell(T_\ell(r))$$

In this case, T is a 4-bit block cipher with a 4-bit key, with the left input of π'^{-1} acting as the key. V is a keyed linear function, and V_ℓ is a linear function translating 4 bits to 4 bits for all ℓ. They then devised a method to prevent $V_\ell(0) = 0$ by substituting $V_\ell(0)$ with the left side of $\pi'^{-1}(\ell||T_\ell^{-1}(0))$. As a result, they were able to discover a high-level decomposition of π'^{-1}.

Finally, they defined a new keyed function $U_r(\ell) = V_\ell(r)$, where U_r is a permutation for every r. This resulted in the π'^{-1} decomposition presented in Figure 3, where the mini-block ciphers T and U are deconstructed later in the their paper.

$$\pi'^{-1}(\ell||r) = T_\ell(r)||U_{T_\ell}(r)||U_{T_\ell(r)}(\ell)$$

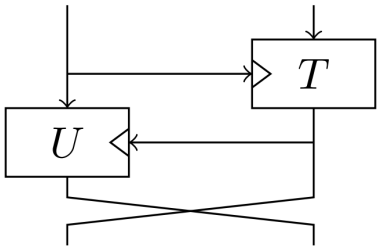

Figure 3. The high level structure of π'^{-1}.
Source: Adapted from [9]

Reverse-Engineering T

The reverse-engineering of T is accomplished by creating a modified version of T such that all lines of T'_k can be produced by a linear combination of $T'_6, T'_7, T'_8,$ and T'_9. They discovered that $T'_6, T'_7, T'_8,$ and T'_9 are all affine equivalents. They could see a distinct LFSR structure if they swapped the two least significant bits (swap2lsb) before and after performing a linear mapping, as shown in Figure 4.

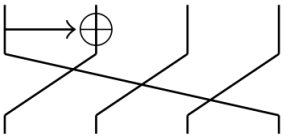

Figure 4. Swapping the least significant bits.
Source: Adapted from [9]

They proceed to derive that the LFSR polynomial is $X^4 + X^3 + 1$, indicating a finite field multiplication. With this understanding they can compute T as:

$T_k(k) = \text{swap2lsb}(f(k) \odot t(x \oplus t_{in}(k) \oplus 0xC))$

Reverse-Engineering U

The reverse-engineering of U was accomplished using the knowledge that $U_k(x) = V_x(k)$ is a linear function when $x \neq 0$, and by re-deriving the permutations $U_2, U_4,$ and U_8 from U_1 using an affine function.

The Structure of π

The authors uncover the high-level structure consisting of T and U in the first decomposition. After reverse-engineering both T and U, they were able to combine the two to build a structure for π'^{-1}. T and U are mini-block ciphers that employ four non-linear 4-bit functions: f, t, u_0, u_1, two finite field multiplications, a "trick" to overcome the non-invertibility of multiplication by 0, with basic linear functions. They generated the S-box of π by applying these expressions to the high-level structure, as shown in Figure 5.

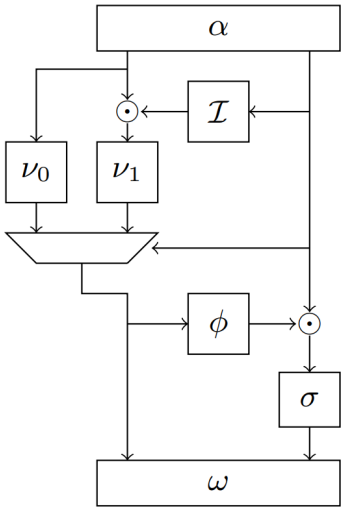

Figure 5. The decomposition of π.
Source: Adapted from [9]

In the diagram, α represents L^*, a 2-bit swap of the lsb in the key U and the ciphertext T, as well as the addition of u_{out} and the inverse of u_f. T_{in} and L^* are combined to form ω. The inverse of U is represented by v_0 and v_1, where $\mathcal{I} : x \to x^{14}$ is the multiplicative inverse in $\mathbb{F}_2^4$. The inverse of T is represented by σ and ϕ.

Summary

They managed to reverse-engineer the hidden structure of the S-box by exploiting patterns revealed by applying the "Pollock's Pattern Recognition" method on the linear approximation table of π. The reverse-engineering entailed constructing a new S-box with similar properties to π by creating a LAT L'_π equal to that of π based on the Jackson Pollock representation. They then exploited the multiset properties

of the inverse of π' to create a high-level decomposition structure of π'^{-1} consisting of two mini-block ciphers T and U. Reverse-engineering T and U let them construct the S-box decomposition of π, however, they had to find the inverse of T and U as they were found using $\pi'^{-}1$.

They determined that it is based on two rounds, similar to a Feistel structure, in which the output of the Feistel function is coupled with the other branch using finite field multiplication rather than exclusive-or. A different heuristic is employed in each round to avoid problems caused by multiplication by 0. This structure is masked by two whitening linear layers that are applied before and after it. To calculate the S-box, five separate 4-bit S-boxes, a multiplexer, two 8-bit linear permutations, and two finite field multiplications in $\mathbb{F}_2^4$ are required.

The attack against π worked by identifying patterns in a visual representation of its LAT and exploiting them to recover parts of the whitening linear layers surrounding the core of the permutation.

V. Discussion

In this section we focus on what the consequences of the reverse-engineering of π are for the Kuznyechik block cipher. We categorize three different types of consequences: hardware, backdoor, and possible attacks.

Hardware Implementation

One of the advantages designers of ciphers have is the knowledge of how to best implement it into hardware. This enables those with the knowledge to develop superior hardware solutions. As we have seen from the reverse-engineering of the S-box, it has a clear structure and one of the reasons for having a highly structured S-box is to allow for efficient implementation. This could be a valid reason for the structure as the designers claimed hardware implementation to be one of the design criteria. The authors also consider two other possible explanations. The first is that it could be to prevent attacks that exploit the inverse in $\mathbb{F}_2^8$. This would be similar to attacks on the AES block cipher. The second is that the designers could have used the structure to implement special properties, like a backdoor, which we will return to later in this section.

The authors tested the hypothesis about the hardware implementation by implementing four different definitions of π to see if there were any noticeable differences. They discovered that the knowledge of the decomposition allows for a more efficient hardware implementation by dividing the area and delay by the factors of 2.5 and 8 respectively. These findings further support the belief that knowing the structure improves hardware implementation.

Backdoor

Cryptanalysis is another benefit of having a structure (backdoor). The authors found support in Bannier's Ph.D. thesis [11], in which he proposed a method for implementing a backdoor in a block cipher. Bannier proved that an S-box mapping additive cosets of a subspace to additive cosets of a subspace is required to have a partition-preserving backdoor. The backdoor would not be affected by the key schedule and can be defined as:

$$x \in \mathcal{V}_i \leftrightarrow E_k(x) \in \mathcal{W}_i$$

This type of attribute must be implemented at the S-box level if you want these partitions to be retained such that two elements in the same $\mathcal{W}_i$ always end up in the same $\mathcal{W}_i$.

What we see in π is that the designer's self-claimed random generating process produces an S-box mapping multiplicative cosets to additive cosets. This structure of π (see decomposition of π in Figure 5) is reminiscent of a known backdoor structure (see Figure 6). As a result, Léo Perrin stated in one of his talks [2] that he was suspicious of the structure of π and further discouraged the ISO representatives to not standardizing the block cipher in 2019.

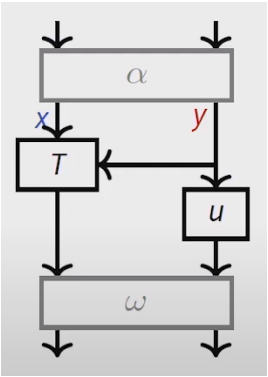

Figure 6. Partition-based backdoor depicted by Léo Perrin.
Source: Adapted from [2]

Possible Attacks

The authors used the decomposition of π to study its cryptographic properties further. Based on the unique resilience of each of its components, they hypothesized that π may be robust against differential and linear attacks. However, only the inverse function has a differential uniformity of four, according to their findings. They also observed that the structure of π strongly suggests the existence of a truncated differential. They discovered that if the left branch value is 0 for two different inputs, the output difference on the left branch would remain 0 with a probability of 1. Moreover, except for the inverse function which is utilized just once, none of the decomposition's components have very good cryptographic qualities.

They looked at the decomposition to see if it is possible to attack the corresponding algorithms in other ways. They managed to find another LAT-based attack against the linear whitening which targeted a specific pattern in the LAT of a 4-round Feistel Network using bijective Feistel functions. An attack methodology was developed:

1) Identify patterns in the LAT
2) Deduce partial whitening linear layers

3) Recover the core of the permutation with an ad hoc attack

Using affine layers, the patterns in the Jackson Pollock representation of the LAT may be exploited to attack a whitened (patterns of zeroes) 4-round Feistel Network. When the affine layers are applied before and after a 4-round Feistel Network, the white segments in the LAT are scrambled linearly, and each segment becomes an affine subspace. The attack's main premise is to calculate the target's LAT and then attempt to reconstruct both the horizontal and vertical portions. They retrieve portions of the linear permutations applied to the rows and columns of the inner Feistel Network's LAT, as well as sections of the real linear layers, using Theorem 1 in their paper [3].

The initial phase in the attack is to locate the Feistel Networks' decomposition with whitening linear layers. They used $(n \times n)$-bit matrices to create three structures and assumed they had the whole codebook for Feistel's composition (see Figure 7 for a conceptual example of a codebook). As a result, they were able to calculate its LAT. They then search for linear subspaces with specific properties that allow for the creation of a vector space in the LAT, to produce an affine space. They wanted each vertical segment's row indices to have an identical linear component in the Jackson Pollock depiction. To change the first LAT and build one without a branch on the left side in the input layer, they use an arbitrary bijective linear mapping. The inverse was treated the same way to find a linear mapping that would allow them to create a new permutation.

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
T_0	e	f	2	5	7	b	8	1	3	c	d	a	0	9	4	6
T_1	2	9	a	4	e	6	7	b	1	8	3	d	0	c	f	5
T_2	e	f	2	5	7	b	8	1	3	c	d	a	0	9	4	6
T_3	5	d	4	2	6	7	b	8	c	1	9	f	0	3	a	e
T_4	5	e	6	7	4	3	f	a	0	1	d	2	8	b	c	9
T_5	9	d	f	a	c	6	8	1	0	5	b	3	2	4	e	7
T_6	3	9	d	f	1	e	b	8	0	2	7	c	4	a	5	6
T_7	5	e	6	7	4	3	f	a	0	1	d	2	8	b	c	9
T_8	7	b	8	5	9	d	c	3	2	e	a	f	6	1	0	4
T_9	d	f	a	c	e	6	2	5	1	3	b	7	9	4	0	8
T_a	e	6	7	4	c	3	8	1	a	2	d	9	5	b	0	f
T_b	4	2	5	d	b	8	6	7	9	f	c	1	a	e	0	3
T_c	2	5	a	4	3	9	d	8	c	f	0	7	b	1	6	e
T_d	e	6	2	5	d	f	a	c	9	4	0	8	1	3	b	7
T_e	9	d	c	3	7	b	8	5	6	1	0	4	2	e	a	f
T_f	8	1	7	b	2	5	e	f	4	6	0	9	d	a	3	c

(a) T.

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
U_0	8	f	0	2	d	5	6	9	e	3	1	7	c	b	4	a
U_1	8	c	7	3	d	f	2	0	e	4	1	b	6	5	9	a
U_2	3	4	e	9	d	8	0	5	1	2	c	f	7	b	a	6
U_3	b	8	9	a	0	7	2	5	f	6	d	4	1	e	3	c
U_4	c	2	5	b	e	8	7	1	4	f	d	6	9	3	0	a
U_5	4	e	2	8	3	7	5	1	a	b	c	d	f	6	9	0
U_6	f	6	b	2	3	0	7	4	5	d	1	9	e	8	a	c
U_7	7	a	c	1	e	f	5	4	b	9	0	2	8	d	3	6
U_8	a	f	b	e	c	4	d	5	7	0	6	1	8	3	9	2
U_9	2	3	c	d	1	b	f	5	9	4	7	a	e	6	0	8
U_a	9	b	5	7	1	c	d	0	6	2	a	e	f	8	3	4
U_b	1	7	2	4	c	3	f	0	8	6	b	5	9	d	a	e
U_c	6	d	e	5	2	c	a	4	3	f	b	7	1	0	9	8
U_d	e	1	9	6	f	3	8	4	d	b	a	c	7	5	0	2
U_e	5	9	0	c	f	4	a	1	2	d	7	8	6	b	3	e
U_f	d	5	7	f	2	b	8	1	c	9	6	3	0	e	a	4

(b) U.

Figure 7. This figure shows what a codebook is in the context of cryptography. Source: Adapted from [3]

They begin the second step by decomposing the previous step's outcome, which is a 2n-bit 4-round Feistel Network made up of two n-bit linear permutations. The goal of this phase is to recover these permutations. They make an observation inspired by the yoyo-game to attack the 4-round Feistel Network with the help of [12]. They realize that by deducing the difference in the right output words of the ciphertext pairs acquired by their yoyo-game approach, they may retrieve the two linear permutations. The approach generates a differential, which may be used to recover the linear components of permutations by iterating over them. However, as seen in [12] and [3], they have to swap the roles of encryption and decryption when targeting the two n-bit linear permutations.

In the last phase, they use a guess and determine a strategy to recover all four Feistel functions in the modified form of the original Feistel Network, omitting its linear layers, in time $O(2^{3n/2})$. As a consequence, they have a four-round Feistel Network that they may use to decompose the original Feistel Network.

VI. Conclusion

Alex Biryukov, Léo Perrin, and Aleksei Udovenko reverse-engineered the S-box π utilized by the Russian block cipher Kuznyechik and presented it in 2016. This achievement was made feasible by the researchers' earlier work in cryptography, which they significantly depended on during the research. They were able to leverage the visual patterns in the LAT of π to exploit the linear whitening layers, recover the hidden structures, and decompose the S-box using Pollock's Pattern Recognition approach, which was invented by two of the authors. This decomposition has important consequences for π. The first is that hardware implementation is made more effective and diminishes the designer's advantage. This is bad as the designers state that the structure in the S-box is a result of having an effective hardware implementation. Furthermore, the authors are suspicious of the structure as it resembles a similar structure of S-boxes that enables backdoors in the cipher. Thus, the integrity of the cipher is questioned. Lastly, the decomposition allows for further analysis of the S-box and enables the development of more attacks. This knowledge means that it is likely that there could be found more vulnerabilities and attacks in the future. The best-case scenario is that the cipher is tested by new attacks and found to be good, although the authors remain skeptical that the S-box employs the best cryptographic properties. To that extent, they discourage the standardization of Kuznyechik until the designers clearly explain why these findings are beneficial.

References

[1] V. Dolmatov, "Rfc7801 - gost r 34.12-2015: Block cipher "kuznyechik"," 2020. https://datatracker.ietf.org/doc/rfc7801/ [Accessed on: 29.03.2022].

[2] L. Perrin, "Partitions in the s-box of streebog and kuznyechik," 2019. https://www.youtube.com/watch?v=6DQATqVlO30&t=62s [Accessed on: 06.03.2022].

[3] A. U. Alex Biryukov, Léo Perrin, "Reverse-engineering the s-box of streebog, kuznyechik and stribobr1 (full version)*," 2016. https://eprint.iacr.org/2016/071.pdf#page=30&zoom=100,180,440 [Accessed on: 07.03.2022].

[4] A. Biryukov and A. Shamir, "Structural cryptanalysis of sasas," *Pfitzmann, B., ed.: Advances in Cryptology – EUROCRYPT 2001*, vol. Volume 2045 of Lecture Notes in Computer Science, p. 395–405, 2001. https://link.springer.com/chapter/10.1007/3-540-44987-6_24 [Accessed on: 04.03.2022].

[5] J. Patarin, "Luby-rackoff: 7 rounds are enough for $2^{n(1-\varepsilon)}$ security," *Advances in Cryptology - CRYPTO'03*, pp. 513–529, 2003. https://iacr.org/archive/crypto2003/27290510/27290510.pdf [Accessed on: 25.02.2022].

[6] P. Subramanyan, S. Ray, and S. Malik, "Evaluating the security of logic encryption algorithms," 2015. https://www.eecs.umich.edu/courses/eecs578/eecs578.f15/papers/sub15.pdf [Accessed on: 27.03.2022].

[7] H. M. Heys, “A tutorial on linear and differential cryptanalysis,” 2015. https://ioactive.com/wp-content/uploads/2015/07/ldc_tutorial.pdf [Accessed on: 29.03.2022].
[8] Springer, “Overview of imrad structure,” 2020. https://www.springer.com/gp/authors-editors/journal-author/overview-of-imrad-structure/1408 [Accessed on: 07.03.2022].
[9] A. Biryukov and L. Perrin, “On reverse-engineering s-boxes with hidden design criteria or structure.” https://www.iacr.org/archive/crypto2015/92160208/92160208.pdf, 2015. [Accessed on: 25.02.2022].
[10] B. Preneel, “Analysis and design of cryptographic hash functions,” *PhD thesis*, 2003. https://eprint.iacr.org/2016/071.pdf [Accessed on: 03.03.2022].
[11] A. Bannier, “Combinatorial analysis of block ciphers with trapdoors,” *PhD thesis*, 2017. https://tel.archives-ouvertes.fr/tel-03125786/document [Accessed on: 06.03.2022].
[12] L. P. Alex Biryukov, Gaëtan Leurent, “Cryptanalysis of feistel networks with secret round functions,” 2015. https://hal.inria.fr/hal-01243130/document [Accessed on: 07.03.2022].