IACR Transactions on Symmetric Cryptology
ISSN 2519-173X, Vol. 2016, No. 2, pp. 99–124. DOI:10.13154/tosc.v2016.i2.99-124

Exponential S-Boxes: a Link Between the S-Boxes of BelT and Kuznyechik/Streebog

Léo Perrin and Aleksei Udovenko

Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg, Luxembourg, Luxembourg
{leo.perrin,aleksei.udovenko}@uni.lu

Abstract. The block cipher Kuznyechik and the hash function Streebog were recently standardized by the Russian Federation. These primitives use a common 8-bit S-Box, denoted π, which is given only as a look-up table. The rationale behind its design is, for all practical purposes, kept secret by its authors. In a paper presented at Eurocrypt 2016, Biryukov et al. reverse-engineered this S-Box and recovered an unusual Feistel-like structure relying on finite field multiplications.
In this paper, we provide a new decomposition of this S-Box and describe how we obtained it. The first step was the analysis of the 8-bit S-Box of the current standard block cipher of Belarus, BelT. This S-Box is a variant of a so-called exponential substitution, a concept we generalize into pseudo-exponential substitution. We derive distinguishers for such permutations based on properties of their linear approximation tables and notice that π shares some of them. We then show that π indeed has a decomposition based on a pseudo-exponential substitution. More precisely, we obtain a simpler structure based on an 8-bit finite field exponentiation, one 4-bit S-Box, a linear layer and a few modular arithmetic operations.
We also make several observations which may help cryptanalysts attempting to reverse-engineer other S-Boxes. For example, the visual pattern used in the previous work as a starting point to decompose π is mathematically formalized and the use of differential patterns involving operations other than exclusive-or is explored.

Keywords: Reverse-Engineering · S-Box · Streebog · Kuznyechik · STRIBOBr1 · White-Box · Linear Approximation Table · Exponentiation · BelT

1 Introduction

Symmetric cryptographic primitives such as block ciphers and hash functions cannot be linear. Therefore, their design often relies on so-called *S-Boxes*, non-linear functions operating on a small enough space that they can be specified through their look-up table.

The choice of an S-Box is a crucial step of the design process of a primitive as the resilience against linear [TCG92, Mat94] and differential [BS91] cryptanalysis usually hinges on the mathematical properties of this function. For example, the excellent properties of the S-Box of the AES [DR02] allow the designers of this block cipher to show that 4 rounds of the block cipher are sufficient to prevent single-trail differential distinguishers.

Three broad categories can be identified when it comes to S-Box design strategies. The first is the use of mathematical objects, most prominently the finite field multiplicative inversion first presented in [Nyb94]. The second approach is to pick many S-Boxes uniformly at random and select the one with the best properties or, alternatively, to use a hill climbing algorithm to optimize these properties. This is for instance the method used to design the S-Boxes of the last Ukrainian standard, Kalyna [OGK+15]. The last approach relies on using strategies from the design of block ciphers by building S-Boxes using for instance

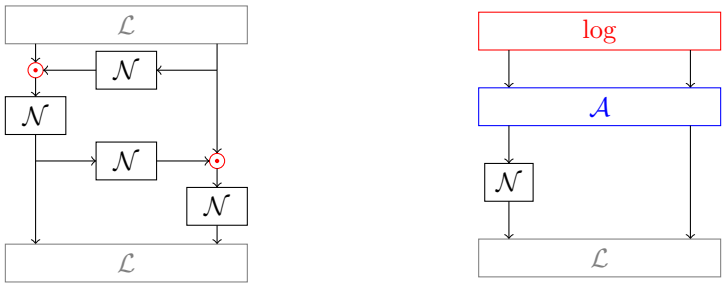
Received: 2016-09-01, Accepted: 2016-11-01, Published: 2017-02-03

Feistel structures [CDL15] or small Substitution-Permutation Networks as has been done e.g. in CLEFIA [SSA+07].

Unfortunately, some institutions do not describe the rationale behind the choice of the S-Boxes they use and merely give their look-up table. Still, as has been shown by a recent line of research starting with [BP15], it is sometimes possible to *reverse-engineer* S-Boxes, which means to figure out its design criteria or, if there is one, recover the structure used.

A prominent target for this type of analysis is the 8-bit S-Box shared[1] by two of the last standards of the Russian Federation, namely the hash function Streebog [Fed12] which is also an IETF standard [DD13] and the block cipher Kuznyechik [Fed15]. While the designers of the algorithm did not disclose the method used to generate this component, Biryukov *et al.* recently managed to recover a first decomposition. A high-level view of this decomposition is provided in Figure 1a. It is based on five 4-bit S-Boxes, a multiplexer (not shown), two finite field multiplications and two linear layers specified using binary matrices. These components are assembled in a fashion reminiscent of a Feistel network where the usual exclusive-OR has been replaced by multiplications. Still, the design process of this S-Box remains a mystery: for instance, how were the 4-bit components chosen?

Another block cipher has been standardized recently by Belarus, BelT [Bel11]. It also uses an 8-bit S-Box specified only by its look-up table. Little information is available about this algorithm in the literature but, as we will see, not only is its S-Box highly structured, its S-Box is in fact related to that of the Russian algorithms.



(a) Previous decomposition.

(b) New decomposition.

Figure 1: A simplied view of previous and our new decompositions of π. Linear (resp. nonlinear) functions are denoted $\mathcal{L}$ (resp. $\mathcal{N}$). $\odot$ denotes finite field multiplication and *log* is a finite field logarithm. $\mathcal{A}$ denotes a few simple integer arithmetic operations. Linear functions are represented in grey, finite field operations in red and integer operations in blue.

Our Contribution Our contribution can be organized along three main axes.

1. We identify some patterns in the linear approximation matrix of the S-Box of BelT and show how those can be explained by the structure of this component which, while not well known, is in fact described in a Belarusian paper. We further describe new distinguishers to identify such permutations.

2. We provide new decompositions of the Russian S-Box relying, like the S-Box of BelT, on finite field exponentiation. While some components of our decompositions remain mysterious, our work illustrates that the decomposition of Biryukov *et al.* is not

[1] This S-Box is also used by StriboBR1, a candidate for the first round of the CAESAR competition designed by Saarinen [Saa15] who is not related to the designers of the Russian standards. The second version of this algorithm [SB15] changed some of the components, including the S-Box, in no small part because of the secrecy surrounding their design.

unique and that the structures used for the Russian and the Belarusian algorithms are related.

3. Finally, we make several observations which may help cryptanalysts trying to reverse-engineer an unknown S-Box, be it based on a finite field exponentiation or not.

Outline This paper is organized as follows. First, we recall in Section 2 the notations and definitions used throughout this work. For the sake of completeness, we provide a brief description of the ciphers BelT and Kuznyechik in Section 3. We then analyse the S-Box of BelT in Section 4: first, we show the patterns we identified in its linear approximation table, then describe the structure its designers used and finally we generalize and provide distinguishers for such permutations. Finally, Section 5 describes our new decompositions of the S-Box of Kuznyechik as well as the process we used to derive them.

2 Notations and Definitions

Throughout the paper, we use the following notations and definitions related to Boolean functions. Let $\mathbb{F}_2 = \{0, 1\}$. The set $\mathbb{F}_2^n$ consisting of elements $x = (x_0, ..., x_{n-1})$ can be given different structure.

- It can be interpreted as a finite field of size 2^n defined as $\mathbb{F}_2[x]/p(x)$ for some irreducible polynomial p of degree n. In this case, we let w be a generator of the multiplicative subgroup and identify $x \in \mathbb{F}_2^n$ with $\hat{x} \in \mathbb{F}_{2^n}$ where $\hat{x} = \sum_{i=0}^{n-1} x_i w^i$. We denote $x \odot y$ the binary representation of the multiplication in the finite field of $\hat{x}$ and $\hat{y}$.
- Alternatively, $\mathbb{F}_2^n$ can be viewed as $\mathbb{Z}/2^n\mathbb{Z}$, in which case $x \in \mathbb{F}_2^n$ is identified with $\overline{x} = \sum_{i=0}^{n-1} x_i 2^i$. We denote $x \boxplus y$ and $x \boxminus y$ the binary representations of $\overline{x} + \overline{y} \mod 2^n$ and $\overline{x} - \overline{y} \mod 2^n$ respectively.

To simplify notations and because there is no ambiguity, if λ is a finite field element then we use λ^x to denote $\lambda^{\overline{x}}$.

We also define the scalar product of two elements x and y of $\mathbb{F}_2^n$ as $x \cdot y = \bigoplus_{i=0}^{n-1} x_i y_i$.

S-Boxes are intended to provide non-linearity in a primitive. As such, they are the components preventing, among others, differential and linear attacks. How good an S-Box is with regards to these attacks can be quantified using the following two tables. We consider a permutation[2] s of $\mathbb{F}_2^n$.

DDT. The Difference Distribution Table of s is the $2^n \times 2^n$ table DDT where the entry $\mathsf{DDT}[i, j]$ is equal to the number of solutions x of $s(x \oplus i) \oplus s(x) = j$.

LAT. The Linear Approximation Table of s is the $2^n \times 2^n$ table LAT where $\mathsf{LAT}[a, b] = (1/2)W_{a,b}$. The coefficients $W_{a,b}$ are the Walsh coefficients of the permutation s and are given by

$$W_{a,b} = \sum_{x \in \mathbb{F}_2^n} (-1)^{a \cdot x \oplus b \cdot s(x)}.$$

The maximum value of $\mathsf{DDT}[i, j]$ for $i \neq 0$ is called differential uniformity of s [Nyb94]. The lower it is, the better an S-Box is at preventing differential attacks. The non-linearity of s is equal to $2^{n-1} - \max(|W_{a,b}|)/2 = 2^{n-1} - \max(|\mathsf{LAT}[a, b]|)$, where the maximum is taken over all non-zero a and b. The higher it is, the better the resilience of the S-Box against linear attacks.

[2] These notions are not tied to s being a permutation but we restrict ourselves to such functions in this paper.

We also recall the following well-known result which relates the LAT of two linear-equivalent permutations.

Theorem 1 (Theorem 1 of [BPU16])**.** *Let f be a permutation of $\mathbb{F}_2^n$ and let $\mathcal{L}$ be its LAT. Let $\mathcal{L}'$ be a table defined by $\mathcal{L}'[u,v] = \mathcal{L}[\mu(u), \eta(v)]$ for some linear permutations μ and η. Then the function f' has LAT $\mathcal{L}'$, where*

$$f' = \eta^t \circ f \circ (\mu^{-1})^t.$$

Finally, we denote ρ^d the rotation left by d steps, that is the permutation of $\mathbb{F}_2^n$ mapping $(x_0, ..., x_{n-1})$ to $(x_d, x_{d+1}, ..., x_{d-1})$.

3 Ciphers Description

3.1 Description of Kuznyechik

Kuznyechik, which means "grasshopper" in Russian, is a 128-bit block cipher developed by the Russian Technical Committee for standardization of "Cryptography and security mechanisms" (TC 26) which is supervised by the Russian Federal Security Service (FSB), i.e. the Russian counterpart of the American National Security Agency (NSA). This block cipher is also known as GOST 34.12-2015 as it is an official Russian standard [Fed15]. It has also been proposed for standardization by the IETF [Dol10b].

This cipher is a Substitution-Permutation Network which uses 9 rounds to encrypt a 128-bit block using a 256-bit key. The linear layer consists in multiplying the internal state by a 16×16 MDS matrix with elements in a finite field of size 2^8. The non-linear layer is an S-Box layer using an 8-bit S-Box π, whose look-up table is given in Table 1. This S-Box is applied in parallel on the full state. There has been few third-party attacks of this cipher. The only one published to the best of our knowledge is a 5-round meet-in-the-middle attack [AY15].

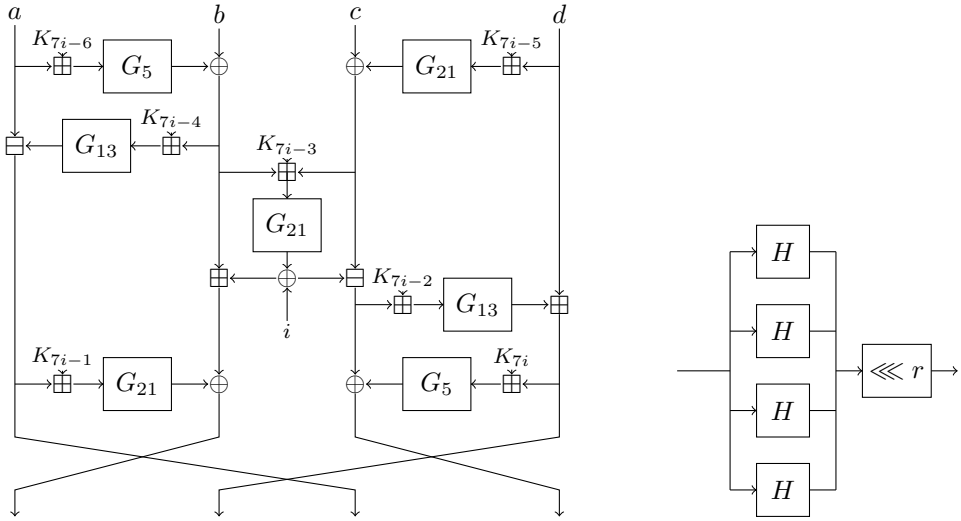
Table 1: The look-up table of π. For example $\pi(\texttt{0x7A}) = \texttt{0xC6}$.

	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F
0.	FC	EE	DD	11	CF	6E	31	16	FB	C4	FA	DA	23	C5	04	4D
1.	E9	77	F0	DB	93	2E	99	BA	17	36	F1	BB	14	CD	5F	C1
2.	F9	18	65	5A	E2	5C	EF	21	81	1C	3C	42	8B	01	8E	4F
3.	05	84	02	AE	E3	6A	8F	A0	06	0B	ED	98	7F	D4	D3	1F
4.	EB	34	2C	51	EA	C8	48	AB	F2	2A	68	A2	FD	3A	CE	CC
5.	B5	70	0E	56	08	0C	76	12	BF	72	13	47	9C	B7	5D	87
6.	15	A1	96	29	10	7B	9A	C7	F3	91	78	6F	9D	9E	B2	B1
7.	32	75	19	3D	FF	35	8A	7E	6D	54	C6	80	C3	BD	0D	57
8.	DF	F5	24	A9	3E	A8	43	C9	D7	79	D6	F6	7C	22	B9	03
9.	E0	0F	EC	DE	7A	94	B0	BC	DC	E8	28	50	4E	33	0A	4A
A.	A7	97	60	73	1E	00	62	44	1A	B8	38	82	64	9F	26	41
B.	AD	45	46	92	27	5E	55	2F	8C	A3	A5	7D	69	D5	95	3B
C.	07	58	B3	40	86	AC	1D	F7	30	37	6B	E4	88	D9	E7	89
D.	E1	1B	83	49	4C	3F	F8	FE	8D	53	AA	90	CA	D8	85	61
E.	20	71	67	A4	2D	2B	09	5B	CB	9B	25	D0	BE	E5	6C	52
F.	59	A6	74	D2	E6	F4	B4	C0	D1	66	AF	C2	39	4B	63	B6

This S-Box is also used by Streebog which is both the standard hash function for Russia [Fed12] and an IETF standard [Dol10a]. It has been designed by the same institution.

3.2 Description of BelT

In 2011, a new block cipher has been standardized by Belarus, called BelT [Bel11]. It encrypts a 128-bit block using a 128-,192- or 256-bit key. The internal state is divided into four 32-bit branches and the round function consists in a mix of Feistel-like operations along with a Lai-Massey-like one. Exclusive-or, addition and subtraction modulo 2^{32} are used to combine the output of the Feistel functions with the different branches. A high-level view of the round function is provided in Figure 2.



(a) The round function of BelT. (b) The 32-bit function G_r.

Figure 2: A high level view of BelT.

The G_r function is a 32-bit permutation. It has a structure very similar to that of the Feistel function of the Magma (previously called GOST) block cipher which is still part of the standard block cipher suite of the Russian Federation [Fed15], except that the layer of 8 parallel 4-bit S-Boxes is replaced by 4 parallel 8-bit ones. Unlike in Magma/GOST, the S-Boxes used in G_r are all identical. This S-Box is denoted H and is specified in Table 2. Our purpose in this paper is not to attack this block cipher but merely to study its S-Box. A complete specification of the block cipher is given in [Bel11].

The S-Box H is differentially 8-uniform and the maximum coefficient of its LAT is 26. Using the approach originally applied to the "F-Table" of Skipjack [BP15], we can show that the probability that a random S-Box has a similar or better pair $(\max(\mathsf{LAT}), \#\{(i,j), \mathsf{LAT}[i,j] = \max(\mathsf{LAT})\})$ is upper bounded by $2^{-122.9}$. Obviously, this S-Box has not been picked uniformly at random and has not been picked as the best element from a feasibly large set of random permutations.

The specification of the algorithm does not explain how this S-Box has been obtained. We therefore set out to reverse-engineer it.

4 Studying the S-Box of BelT

The specification of [Bel11] does not contain any information about the actual design of its S-Box. Indeed, much like in the specification of Kuznyechik [Fed15] or that of the NSA cipher Skipjack [NSA98], it is simply given as a look-up table. We naturally tried to reverse-engineer this S-Box or, in other words, we attempted to recover as much information as possible from its look-up table. However, it later turned out that the

Table 2: The look-up table of H. For example $H(\texttt{0x7A}) = \texttt{0x18}$.

	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F
0.	B1	94	BA	C8	0A	08	F5	3B	36	6D	00	8E	58	4A	5D	E4
1.	85	04	FA	9D	1B	B6	C7	AC	25	2E	72	C2	02	FD	CE	0D
2.	5B	E3	D6	12	17	B9	61	81	FE	67	86	AD	71	6B	89	0B
3.	5C	B0	C0	FF	33	C3	56	B8	35	C4	05	AE	D8	E0	7F	99
4.	E1	2B	DC	1A	E2	82	57	EC	70	3F	CC	F0	95	EE	8D	F1
5.	C1	AB	76	38	9F	E6	78	CA	F7	C6	F8	60	D5	BB	9C	4F
6.	F3	3C	65	7B	63	7C	30	6A	DD	4E	A7	79	9E	B2	3D	31
7.	3E	98	B5	6E	27	D3	BC	CF	59	1E	18	1F	4C	5A	B7	93
8.	E9	DE	E7	2C	8F	0C	0F	A6	2D	DB	49	F4	6F	73	96	47
9.	06	07	53	16	ED	24	7A	37	39	CB	A3	83	03	A9	8B	F6
A.	92	BD	9B	1C	E5	D1	41	01	54	45	FB	C9	5E	4D	0E	F2
B.	68	20	80	AA	22	7D	64	2F	26	87	F9	34	90	40	55	11
C.	BE	32	97	13	43	FC	9A	48	A0	2A	88	5F	19	4B	09	A1
D.	7E	CD	A4	D0	15	44	AF	8C	A5	84	50	BF	66	D2	E8	8A
E.	A2	D7	46	52	42	A8	DF	B3	69	74	C5	51	EB	23	29	21
F.	D4	EF	D9	B4	3A	62	28	75	91	14	10	EA	77	6C	DA	1D

structure of the S-Box was public: the design rationale behind the BelT cipher and the S-Box is explained in a separate note [AGMH40]. Nevertheless, our cryptanalysis attempt leads to new reverse-engineering techniques and to the discovery of new properties of a particular S-Box class which we describe below.

Here, we first show in Section 4.1 how the distributions of coefficients in LAT of an S-Box can be used in a different way from what has been done before e.g. in [BP15]. Then, we will see that tweaking the definition of the DDT may also yield interesting results in Section 4.2, in particular for H. Finally, we will interpret our results in light of the actual structure of H in Section 4.3.

4.1 Looking at Lines/Columns

A key step in reverse-engineering an S-Box is the study of the distribution of the coefficients in its DDT and in its LAT. A paper by Daemen and Rijmen [DR07] specifies the probability distribution of these coefficients under the assumption that each coefficient is a sample from a given distribution and that all sampling are independent. The authors show ample empirical evidence to illustrate the validity of this assumption. This model has the following implication.

Observation 1. *The lines and the columns of the DDT of a random permutation should be independent from one another. The same is true for the LAT.*

In particular, the variance of the absolute value of the coefficients in each column should be independent from the column index. The fact that this is not true for the S-Box of Streebog and Kuznyechik is what allowed Biryukov et. al to decompose it in [BPU16]. Indeed, the visual pattern they observed in the LAT corresponds to 15 columns containing each exactly 16 zeroes and, apart from that, only coefficients with an absolute value v in $\{4, ..., 12\}$. These columns had thus a lower contrast than the others, a motif which the cryptanalyst's eye can efficiently identify. Figure 3 plots the variance of the absolute values of the coefficients in each column[3] of the LAT of Kuznyechik. As expected, we can see 15 column indices for which the variance is significantly lower and which correspond to the "red lines" Biryukov *et al.* saw in the LAT. Furthermore, all of those variances are equal. In fact, all these columns have the exact same distribution of coefficients.

[3]Note that we operate on the *absolute value* of the coefficients. Indeed, the variance of the *signed* coefficients is constant because of Parseval's equality.

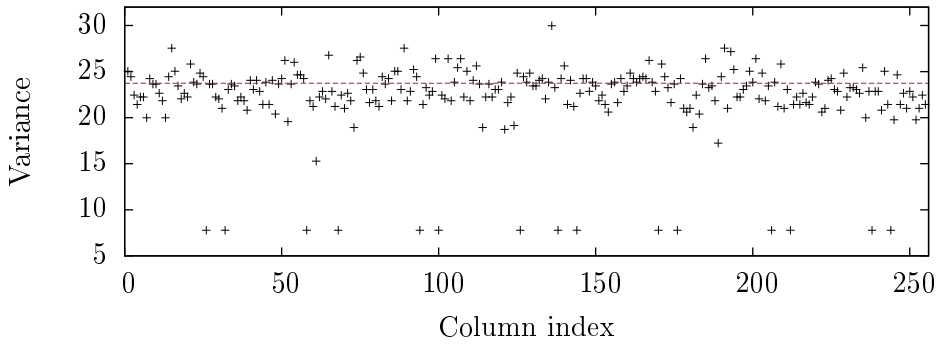


Figure 3: The variance of the columns of the LAT (absolute values) of π (Streebog). The expected variance is represented with a red dashed line.

The columns of the LAT of H yield no such artifact. However, Figure 4 plots the variance of the absolute values of the *lines* of the LAT of BelT and, as we can see, some of the lines have identical and abnormally low variance.

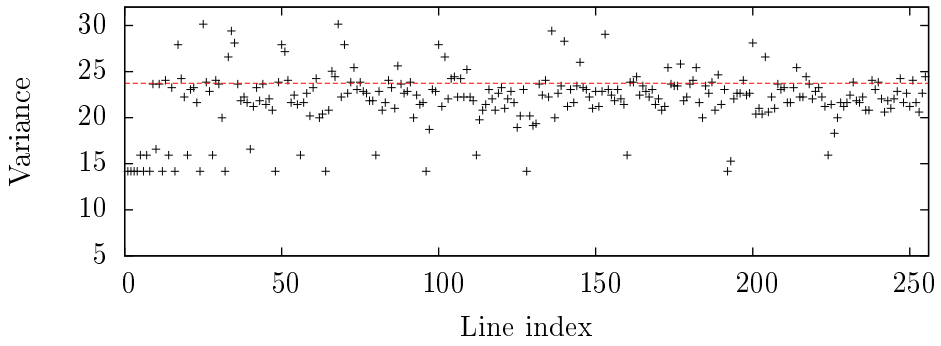


Figure 4: The variance of the lines of the LAT (absolute values) of H (BelT). The expected variance is represented with a red dashed line.

4.2 Alternative DDT

Because of the structure of BelTs Feistel function (see Figure 2b), where the key is added using modular addition and where the output might be added, subtracted or xored into the internal state, it is natural to investigate "hybrid" differentials where the input and output operations are distinct.

For example, we count the number of solutions of the following equation for different S-Boxes s and for all a, b in $\mathbb{F}_2^8$:

$$s(x \boxplus a) \oplus s(x) = b,$$

where $\oplus$ denotes a bit-wise exclusive-or and $\boxplus$ denotes addition modulo 2^8. The results are given in Table 3 for several S-Boxes (the cases $a = 0$ and $b = 0$ are ignored): H, π, that of the AES, one picked uniformly at random using a Knuth shuffle and a theoretical distribution. There exists on average one x such that $s(x \boxplus a) \oplus s(x) = b$ so we model the number of solutions of this equation for a given pair (a, b) as a sample from a Poisson

Table 3: The distribution of the number of solutions of $s(x \boxplus a) \oplus s(x) = b$ depending on a and b for different S-Boxes and for a Poisson distribution.

# solutions N	$\#\{(a,b), \#\text{solutions} = N\}$				
	H	π	AES	Random	Theoretical
0	11175	23252	22270	23582	23921.36
1	42498	24466	25486	24148	23921.36
2	11274	12414	12968	12271	11960.68
3	78	3784	3490	3810	3986.89
4	0	945	685	965	996.72
5	0	132	110	204	199.34
6	0	30	14	37	33.22
7	0	2	2	6	4.75
8	0	0	0	2	0.59

distribution with parameter 1. The corresponding expected number of solutions are listed in the "Theoretical" column of Table 3.

As we can see, the maximum number of solutions for H is far too small.

Observation 2. *It is worth considering other differentials than the usual XOR-differences in order to identify patterns in an unknown S-Box.*

4.3 The Actual Structure of H

While we were not aware of it in the beginning, it turns out that the method used to build H is public — though hard to find. Indeed, in a paper from the Belarusian journal " " [Information Security Management] [AGMH40] the designers explain the rationale behind BelT and its S-Box. The following proposition is a translation of this description.

Proposition 1 (The BelT S-Box Construction and its Properties, [AGMH40] (translated))**.** *The look-up tables of the S-Box coordinate functions were chosen as different segments of length 255 of different linear recurrences defined by the irreducible polynomial $p(\lambda)$:*

$$p(\lambda) = \lambda^8 + \lambda^6 + \lambda^5 + \lambda^2 + 1.$$

Additionally, a zero element was inserted in a fixed position of each segment.

We describe some cryptographic properties of the constructed permutation:

1. *the nonlinearity of S is equal to 102 (for the given dimension the nonlinearity cannot exceed 120; high nonlinearity provides resistance against linear cryptanalysis);*

2. *differential properties of S: $R_{\oplus\oplus} = 8$, $R_{\boxplus\boxplus} = 7$, $R_{\oplus\boxplus} = 6$, $R_{\boxplus\oplus} = 3$, where for example $R_{\boxplus\oplus} = \max_{a\neq 0,b} \#\{x \in \mathbb{F}_2^8 \mid S(x \boxplus a) = S(x) \oplus b\}$;*

3. *the algebraic degrees of all the coordinates are equal to 7 (for the given dimension this is the maximum possible value, high algebraic degree prevents some variants of differential attacks);*

4. *there are no quadratic equations relating between the S-Box input and output bits (such equations may lead to algebraic attacks).*

As pointed out by an anonymous reviewer, it is easy to check that H satisfies this description (with zero inserted at position 10) using the code given in Appendix B.

4.3.1 Exponential S-Boxes

S-Boxes built from such a linear recurrence and where 0 maps to 0 are called *exponential substitution*. They were introduced and some of their properties were studied in papers by some of the BelT designers. The first one is a paper published in “ ” [News of the National Academy of Sciences of Belarus] [AA38] while the second is a translation which the authors published on eprint [AA04].

Definition 1 (Exponential substitution [AA04, AA38])**.** Let α be a primitive element of the finite field $\mathbb{F}_{2^n}$ with minimal polynomial $x^n + \sum_{i=1}^{n-1} m_i x^i + 1$. First, let x be an element of $\mathbb{F}_2^n$ and denote $\overline{x} = \sum_{i=0}^{n-1} x_i 2^i$. An *exponential substitution* is a permutation h such that

$$h(x) = \begin{cases} 0 \text{ if } x = 0, \\ \alpha^{\overline{x}} \text{ otherwise.} \end{cases}$$

In this case, the truth table of each coordinate h_i of h is such that, for all x less than $2^n - n$,

$$h_i(x+n) \oplus m_{n-1} h_i(x+n-1) \oplus \ldots \oplus m_1 h_i(x+1) \oplus h_i(x) = 0.$$

In other words, each coordinate of h is a segment of an LFSR sequence with linear recurrence given by the same irreducible polynomial which is used to define the finite field.

In particular, it is shown in [AA04, AA38] that exponential substitutions have reasonably high non-linearity, hybrid differential resistance and high algebraic degree.

It is trivial to recover the structure of exponential substitutions.

Observation 3. *If a permutation is an exponential substitution, then applying the Berlekamp-Massey algorithm [Mas69] on each coordinate will give the linear recurrence used to generate the permutation.*

This reverse-engineering method will fail if the S-Box is composed with affine layers, a method called *affine whitening* in [BPU16]. Nevertheless, exponential substitutions have a very strong algebraic structure. It thus comes as no surprise that such permutations have specific patterns in their Linear Approximation Tables.

Proposition 2. *Let ρ^d denote the rotation by d bits to the left. The distribution of the coefficients in lines a and $\rho^d(a)$ of the LAT of an exponential substitutions are identical for any d.*

Proof. Because of the relation between LAT and Walsh coefficients, we prove the proposition for $W_{a,b}$. Remember that the Walsh coefficients $W_{a,b}$ of a vectorial Boolean function h are given by:

$$W_{a,b} = \sum_{x \in \mathbb{F}_2^n} (-1)^{a \cdot x + b \cdot h(x)}.$$

Note that the scalar product $a \cdot x$ is equal to $\rho^d(a) \cdot \rho^d(x)$. Using this, we rewrite $W_{a,b}$ as follows:

$$\sum_{x \in \mathbb{F}_2^n} (-1)^{a \cdot x + b \cdot h(x)} = \sum_{x \in \mathbb{F}_2^n} (-1)^{\rho^d(a) \cdot \rho^d(x) + b \cdot h(x)} = \sum_{y \in \mathbb{F}_2^n} (-1)^{\rho^d(a) \cdot y + b \cdot h\left(\rho^{-d}(y)\right)},$$

where $y = \rho^d(x)$. Furthermore, for all b there exists a unique b' such that $b \cdot h\left(\rho^{-d}(y)\right) = b' \cdot h(y)$ for all y. This equality obviously holds for $y = 0$. If $y \neq 0$, then the right-hand side is equal to

$$b \cdot h\left(\rho^{-d}(y)\right) = b \cdot \left(\prod_{i=0}^{n-1} \alpha^{y_i 2^{i-d}} \right) = b \cdot \left(\prod_{i=0}^{n-1} \alpha^{y_i 2^i} \right)^{2^{-d}},$$

where $x \mapsto x^{2^{-d}}$ is a linear permutation which can be written $x \mapsto M_d \times x$ for some $n \times n$ binary matrix M_d. We deduce that $b \cdot h\left(\rho^{-d}(y)\right) = b' \cdot h(y)$ for $b' = (M_d^t \times b)$, where M_d^t is the transpose of M_d. As a consequence, the multisets $\{W_{a,b}, \forall b \in \mathbb{F}_2^n\}$ and $\{W_{\rho^d(a),b}, \forall b \in \mathbb{F}_2^n\}$ are identical, which in turn implies the proposition. □

This proposition can be used to distinguish exponential substitutions from random permutations.

If such an S-Box has been composed with an affine layer, i.e. if a permutation σ is equal to $A \circ f$ where A is an affine permutation and f is an exponential permutation, then the pattern described in Proposition 2 is still present: because of Theorem 1, this composition merely shuffles the columns which leaves the distribution of the coefficients in each line unchanged. Adding another linear layer before f shuffles the rows. While this breaks the rotational pattern, the fact that the rows fall into few distinct classes with regards to the distribution of the coefficients remains unchanged.

Observation 4. *Consider the LAT of a permutation $\sigma = A \circ f \circ B$ where A and B are affine permutations and f is an exponential permutation. The distribution of the coefficients in the different lines are not independent. In fact, it is possible to recover some information about B using the fact that the LAT of $A \circ f$ is such that rows a and $\rho^d(a)$ have the exact same coefficient distribution.*

There are other constructions known to exhibit some form of invariance in their LAT. Notably, power functions $x \mapsto x^d$ of $\mathbb{F}_2^n$ are known to have the exact same distribution of coefficients in each line of both their DDT and LAT [BCC10].

4.3.2 Pseudo-exponential substitution

The notion of exponential substitution can be generalized. Indeed, H is strictly speaking *not* an exponential substitution because 0 was not inserted at position 0 but at position 10. We formalize this difference using the following definition.

Definition 2 (Pseudo-Exponential and Pseudo-Logarithm)**.** We call *pseudo-exponential* of n bits a permutation defined by an exponent λ generating the multiplicative group of some field of size 2^n and a position z at which 0 is inserted. It is denoted $\exp_{\lambda,z}$, so that

$$\exp_{\lambda,z}(x) = \begin{cases} 0 \text{ if } \overline{x} = z, \\ \lambda^{x \boxplus 1} \text{ if } \overline{x} < z, \\ \lambda^x \text{ otherwise .} \end{cases}$$

A *pseudo-logarithm* is the functional inverse of a pseudo-exponential. The functional inverse of $\exp_{\lambda,z}$ is denoted $\log_{\lambda,z}$.

This generalization is also convenient to link the structure described in [AA04, AA38] with other permutations from the literature. Indeed, while both discrete logarithm [HN10] and exponential [BR00, RBF08] have been discussed in the Western literature before, the definitions used differ slightly. In [HN10], the discrete logarithm maps $x \neq 0$ to $\log_\alpha(x)$ for some primitive element α of $\mathbb{F}_2^n$ and it maps 0 to $2^n - 1$. Thus, its inverse is not an exponential substitution in the sense of Definition 1: it maps x to α^x unless $x = 2^n - 1$ which is mapped to 0. In this case, 0 is not inserted at position 0 but at position $2^n - 1$. In [RBF08], rotational symmetries of such exponentials were studied. They were also used to build the small 4-bit S-Box E used inside the Whirlpool hash function [BR00]. Therefore, what is called "exponential" in those papers is here called "pseudo-exponential with a 0 at position $2^n - 1$", which is denoted $\exp_{\lambda,2^n-1}$.

Furthermore, as explained in [HN10], the coordinates of $\exp_{\lambda,2^n-1}$ are closely related to the functions introduced by Feng et al. in [FLY09] and studied in [CF09]. In fact,

when the input and output are of the same size, the function of Feng et al. is the discrete logarithm except in $\{0,1\}$: it maps 0 to 0 and 1 to 2^n-1. Incidentally, the coordinates of H^{-1} with low-variance Walsh spectra identified in Section 4.1 are related to the Boolean functions of Feng et al., though not directly because of the position of the 0.

As said above, H has yet another preimage for zero. These variations are to be expected: since the exponential function has no preimage for zero, this quantity has to be set arbitrarily. Unfortunately, this has been done in different ways in previous works. While the impact of the difference between inserting 0 at position 0 or at position 2^n-1 is limited as it merely rotates the truth table, inserting it in the middle of the truth table of the S-Box has more consequences.

Of particular interest to us is the fact that the LAT of a pseudo-exponential is less structured than that of an exponential substitution. Indeed, the LAT of pseudo-exponentials is not invariant through a rotation of the row indices, unlike exponential substitution. In other words, Proposition 2 does not hold for pseudo-exponentials. Still, some patterns remain as explained in the following proposition and its corollary.

Proposition 3. *Let ℓ_z be the smallest integer such that $2^{\ell_z} > z$. The lines of the LAT of the pseudo-exponential $exp_{\lambda,z}$ with indices $a = k \times 2^{\ell_z}$ for any integer k do not depend on z. In particular, they are identical to those of the LAT of an exponential substitution with the base λ, which corresponds to the case $z = 0$.*

Proof. Let h be a pseudo-exponential substitution with exponent λ and preimage for zero z, i.e. $h = \exp_{\lambda,z}$, and let f be the exponential substitution with the same λ. We must prove that the following quantities are identical as long as $a = k \times 2^{\ell_z}$:

$$\sum_{x\in\mathbb{F}_2^n} (-1)^{a\cdot x \oplus b\cdot h(x)} \;=\; \sum_{x\in\mathbb{F}_2^n} (-1)^{a\cdot x \oplus b\cdot f(x)}$$

First of all, let us rewrite the left-hand side:

$$\begin{aligned}
\sum_{x\in\mathbb{F}_2^n} (-1)^{a\cdot x \oplus b\cdot h(x)} &= \sum_{\overline{x}<z} (-1)^{a\cdot x \oplus b\cdot \lambda^{x\boxplus 1}} + (-1)^{a\cdot z} + \sum_{\overline{x}>z} (-1)^{a\cdot x \oplus b\cdot \lambda^x} \\
&= (-1)^{a\cdot z} + \sum_{0<\overline{x}\leq z} (-1)^{a\cdot (x\boxminus 1) \oplus b\cdot \lambda^x} + \sum_{\overline{x}>z} (-1)^{a\cdot x \oplus b\cdot \lambda^x} \\
&= (-1)^{a\cdot z} + \sum_{x\neq 0} (-1)^{a\cdot \phi_z(x) \oplus b\cdot \lambda^x},
\end{aligned}$$

where $\phi_z(x) = x$ if $\overline{x} > z$ and $\phi_z(x) = x \boxminus 1$ otherwise. It is sufficient to prove the proposition to show that if $\overline{a} = k \times 2^{\ell_z}$ then $a \cdot \phi_z(x) = a \cdot x$ for all $x \neq 0$.

If there exists i in $[0, \ell_z - 1]$ such that $x_i = 1$, then the carry from the subtraction cannot propagate to x_j and the other bits with higher weight. Therefore, it is necessary for $a \cdot \phi_z(x)$ to be different from $a \cdot x$ that $x_0 = ... = x_{\ell_z - 1} = 0$. If it is the case, then $\overline{x} = m \times 2^{\ell_z}$ but, in this case, $\phi_z(x) = x$ because then $x \geq 2^{\ell_z} > z$. Thus, for all $x \neq 0$ and all $a = k \times 2^{\ell_z}$, $a \cdot \phi_z(x) = a \cdot x$. The proposition follows. □

Note that Proposition 3 does not apply when $z = 2^n - 1$, i.e. for exponentials studied in [HN10].

As the LAT of a pseudo-exponential shares some of its lines with an exponential substitution, these lines also share the patterns presented in Proposition 2.

Corollary 1. *The distributions of the coefficients in lines with indices $a2^d$ and $a2^{d'}$ of the LAT of a pseudo-exponential substitutions are identical for any a and any d, d' such that $z < \min(2^d, 2^{d'})$ and $\max(a2^d, a2^{d'}) < 2^n$.*

It is possible to represent H using such a pseudo-exponentiation. In order to find the decomposition described below, we brute-forced all possible bases for the exponentiation and all field representations to obtain pseudo exponentials $\exp_{\lambda,10}$. For each, we checked whether $H \circ \exp_{\lambda,10}^{-1}$ was linear and, if it were, computed its matrix representation. Only one of those matrices has a visible structure; it corresponds to the decomposition we kept. Let ω_H be the symmetric matrix defined by

$$\omega_H = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}$$

and let $\lambda = w^7 + w^3 + w$, where w is the generator of the multiplicative group of $\mathbb{F}_2[x]/(x^8 + x^6 + x^5 + x + 1)$. Then $H(x)$ can be computed as follows:

$$H(x) = (\omega_H \circ \exp_{\lambda,10})(x).$$

A SAGE [Dev16] script performing this computation is provided in Appendix A. It is worth noting that $\omega_H(x)$ is the XOR of 10 shifts of x composed with a bit reversal swapping x_0 and x_7, x_1 and x_6, etc.

As the linear mapping is applied after the pseudo-exponential when computing H, it merely shuffles the columns of the LAT of the pseudo-exponential. Therefore, it exhibits the patterns described in Corollary 1:

- rows with indices 16, 32, 64 and 128, that is $\rho^i(1)$ for i in $\{4, 5, 6, 7\}$, have the same distribution of coefficients,
- rows with indices 48, 96 and 192, that is $\rho^i(3)$ for i in $\{4, 5, 6\}$, have the same distribution of coefficients.

This partially explains the patterns we identified during our attempt at reverse-engineering this S-Box. However, we actually identified stronger patterns than those which can be deduced from Corollary 1. Indeed, we found that rows with indices 2^i and 3×2^i all shared the same distribution of absolute value of coefficients.

These patterns were experimentally confirmed for other such pseudo-exponential, so that we make the following conjecture.

Conjecture 1. *Let $\mathcal{L}$ and $\mathcal{L}'$ be the LAT of the exponential substitution with base λ and $exp_{\lambda,z}$ respectively. Then, for all i, the distribution of the absolute value of the coefficients at line 2^i and 3×2^i is the same in $\mathcal{L}$ and $\mathcal{L}'$, regardless of z.*

Note that the proof of Proposition 2 cannot be simply adapted to prove this conjecture. Indeed, although the distribution of the absolute value of the coefficients is preserved, we observed that neither the sign nor the position of the coefficients are identical if $a \neq k \times 2^{\ell_z}$.

Our strategy to try and reverse-engineer H along with the patterns we found have so far been extremely similar to those from [BPU16]. Indeed, the presence of LAT columns with significantly lower variance is what allowed this previous work. An obvious question thus arises: is it possible to represent the S-Box of Kuznyechik, π, using a pseudo-logarithm along with some simple operations as well? As we show in the next session, the answer is yes.

5 An Exponential in the S-Box of Kuznyechik/Streebog

In the S-Box of BelT, the low-variance lines in the LAT are related to the finite field exponentiation. It is thus plausible that π, the S-Box of Kuznyechik and Streebog, is related to such a function as well, or rather to its inverse as the pattern is present in the columns of its LAT rather than in its rows.

We have also noticed another interesting evidence of the finite field logarithm in π: one of the 4-bit S-Boxes from the decomposition from [BPU16], namely ν_0, is actually affine-equivalent to a finite field logarithm in $\mathbb{F}_{2^4}$. We note also that both these S-Boxes are affine-equivalent to the inverse of the 4-bit S-Box E which is used to compute the 8-bit S-Box of the hash function Whirlpool [BR00] and was built using an exponentiation.

5.1 Finding exponential patterns in π^{-1}

The exponentiation $x \mapsto g^x$ has the property that adding some integer c to an input results in multiplying the corresponding output by the field element g^c: $g^{x+c} = g^x \odot g^c$. We can use this property to check whether some function behaves like exponentiation at least on some inputs. However, if the function is masked with some linear layers, this property does not work anymore. On the input side we can change integer addition to xor: the linear layer will just change the xor constant. If this constant is of the form 2^k, then the k-th bit in the exponent is flipped and the output is either multiplied or divided by g^{2^k}.

We made an exhaustive search for all xor constants and found four high probability relations. Let $c = \{\texttt{0x12}, \texttt{0x26}, \texttt{0x24}, \texttt{0x30}\}$. Then for all but 16 values of x one of the following two relations holds:

$$\pi^{-1}(x \oplus c_i) = \pi^{-1}(x) \odot w^{2^i}, \text{ or}$$
$$\pi^{-1}(x \oplus c_i) = \pi^{-1}(x) / w^{2^i},$$

where i runs over $\{0, 1, 2, 3\}$, multiplication and division are done in the finite field $\mathbb{F}_2[x]/(x^8 + x^4 + x^3 + x^2 + 1)$ and w is the primitive element defined by x. The irreducible polynomial is the default one in Sage [Dev16] and, for other polynomials, the relations hold with much lower probabilities. Interestingly, the linear layer of the block cipher Kuznyechik uses multiplications in a finite field defined by another irreducible polynomial. Therefore it is not clear whether the S-Box and the linear layer of the block cipher interact in some way. We also recall that the linear layer of the block cipher used as a component of the hash function Streebog has also been reverse-engineered [KK13] and, much like in Kuznyechik, uses finite field arithmetic over $\mathbb{F}_2^8$ defined by the irreducible polynomial $x^8 + x^6 + x^5 + x^4 + 1$.

To simplify further analysis, we can remove the exponentiation and work only with exponents. That is, we analyze $\tau = \log_{w,0} \circ \pi^{-1}$.

By our hypothesis, the constants c_i should be mapped by the linear whitening layer to some powers of 2. Moreover, from the observed relations we can guess that these powers should be $(1, 2, 4, 8)$ (the exponents of w). We therefore assume that the unknown linear layer maps c_i to 2^i for $i \in \{0, 1, 2, 3\}$. Now we need to complete the mapping by setting preimages for powers 2^j for $j \in \{4, 5, 6, 7\}$. First we complete it randomly and get some linear mapping α (see Equation 1). We then observe that the look-up table of $\tau \circ \alpha^{-1}$ considered as a 16×16 table is very highly structured (see Table 4). Moreover, by xoring a constant which depends on the column index to the right branch before the permutation it is possible to sort each of the rows. This dependence is linear so it can be included into the linear layer. The matrix of the resulting linear layer β is given in Equation 1 and the look-up table of $\tau \circ \beta^{-1}$ is given in Table 5.

Table 4: The look-up table of $\tau^{-1} \circ \alpha^{-1}$.

	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F
0.	188	189	190	191	192	193	194	195	196	197	198	199	200	201	202	203
1.	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
2.	143	144	141	142	139	140	137	138	151	152	149	150	147	148	145	146
3.	160	161	158	159	156	157	154	155	168	169	166	167	164	165	162	163
4.	216	215	214	213	220	219	218	217	208	207	206	205	212	211	210	209
5.	97	96	95	94	101	100	99	98	89	88	87	86	93	92	91	90
6.	48	47	50	49	44	43	46	45	40	39	42	41	36	35	38	37
7.	82	81	84	83	78	77	80	79	74	73	76	75	70	69	72	71
8.	172	171	174	173	176	175	178	177	180	179	182	181	184	183	186	185
9.	53	52	55	54	57	56	59	58	61	60	63	62	65	64	67	66
A.	127	126	125	124	123	122	121	120	135	134	133	132	131	130	129	128
B.	246	245	244	243	242	241	240	239	254	253	252	251	250	249	248	247
C.	232	233	230	231	236	237	234	235	224	225	222	223	228	229	226	227
D.	113	114	111	112	117	118	115	116	105	106	103	104	109	110	107	108
E.	221	238	255	0	153	170	187	204	85	102	119	136	17	34	51	68
F.	13	14	15	16	9	10	11	12	5	6	7	8	1	2	3	4

Table 5: The look-up table of $\log_{\lambda,0} \circ \pi^{-1} \circ \beta^{-1}$.

	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F
0.	188	189	190	191	192	193	194	195	196	197	198	199	200	201	202	203
1.	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
2.	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152
3.	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169
4.	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220
5.	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101
6.	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
7.	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84
8.	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186
9.	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67
A.	120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135
B.	239	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254
C.	222	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237
D.	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118
E.	17	34	51	68	85	102	119	136	153	170	187	204	221	238	255	0
F.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

$$\alpha = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}^{-1}, \quad \beta = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}^{-1}. \tag{1}$$

Now we can reorder rows by applying a 4-bit S-Box on the left input branch of $\tau \circ \beta^{-1}$. Let

$$q = (12, 2, 9, 10, 13, 6, 3, 5, 11, 4, 8, 15, 14, 7, 0, 1),$$

and let $q_L(x||y) = q(x)||y$ be the permutation of $\mathbb{F}_2^4 \times \mathbb{F}_2^4$. The look-up table of $\tau \circ \beta^{-1} \circ q_L^{-1}$ is given in Table 6. We describe a decomposition of the inverse of π based on these findings in Algorithm 1.

We note that q is affine-equivalent to one of the S-Boxes used in the GOST R 34.11-94 hash function, namely the S-Box "pi[1]" specified in RFC 5831 [Dol10a]. This hash

Table 6: The look-up table of $\log_{\lambda,0} \circ \pi^{-1} \circ \beta^{-1} \circ q_L^{-1}$.

	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F
0.	17	34	51	68	85	102	119	136	153	170	187	204	221	238	255	0
1.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2.	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
3.	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
4.	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67
5.	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84
6.	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101
7.	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118
8.	120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135
9.	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152
A.	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169
B.	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186
C.	188	189	190	191	192	193	194	195	196	197	198	199	200	201	202	203
D.	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220
E.	222	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237
F.	239	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254

Algorithm 1 Computing the inverse of π: $y = \pi^{-1}(x)$.

$(l||r) \leftarrow \beta(x)$
$l \leftarrow q(l)$
if $l = 0$ **then**
    $z \leftarrow 17 \times ((r+1) \mod 16)$
else
    $z \leftarrow 17 \times l + r - 16$
end if
$y \leftarrow \exp_{w,0}(z)$
return y

function is the predecessor of Streebog. It was designed by the Russian Federal Agency of Government Communications and Information, an institution which was later incorporated to the FSB. However, there are only 302 affine-equivalence classes of 4-bit S-Boxes (see Table 5 of [BDCBP03]), meaning that a mere coincidence cannot be ruled out.

5.2 Obtaining a First Decomposition of π

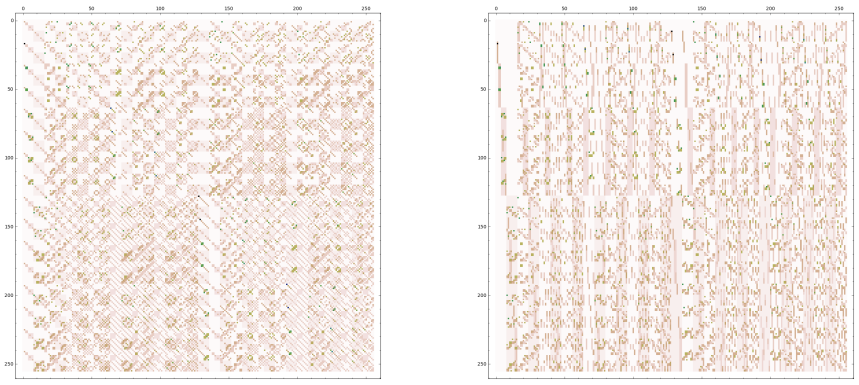
It is easy to invert Algorithm 1 to obtain a direct expression of π. However, doing this requires introducing a base-17 decomposition of the output of a pseudo-logarithm. Our aim in this section is to find a decomposition of π which does not mix arithmetic modulo 16 and 17. In fact, we want to simplify the arithmetic layer as much as possible.

To achieve this, we concentrate on the function $\mathcal{P} = \log_{w,0} \circ \pi^{-1} \circ \beta^{-1} \circ q_L^{-1}$ (which is given in Table 6). The LAT of this function is shown in Figure 5a. We observe that the LAT of $\mathcal{P}$ composed with a branch swap contains a white square in the top left corner (see Figure 5b). As with the decomposition of π performed in [PUB16], this white square corresponds to particular integral properties: if the left 4-bit nibble of the input takes all possible values then the the left 4-bit nibble of the output takes all possible values as well. Therefore, $\mathcal{P}$ has a so-called *TU-decomposition*.

Lemma 1 (TU-Decomposition, [PUB16])**.** *Let f be a function mapping $\mathbb{F}_2^n \times \mathbb{F}_2^n$ to itself such that fixing the right input to any value and letting the left one take all 2^n possible values leads to the left output taking all 2^n possible values. Then f can be decomposed using a keyed n-bit permutation T and a keyed n-bit function U (see Figure 6):*

$$f(x, y) \;=\; \big(T_y(x), U_{T_y(x)}(y)\big),$$

Besides, if f is a permutation then U is a keyed permutation.



(a) (b)

Figure 5: The Jackson Pollock representation of the LATs of $\mathcal{P}$ and $\mathcal{P}$ composed with a branch swap.

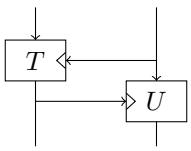


Figure 6: TU-decomposition.

Table 7: The mini-block ciphers from the TU-decomposition of $\mathcal{P}$.

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

(a) T.

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

(b) U.

The function $\mathcal{P}$ does indeed have such a decomposition and the keyed permutations T and U are given in Table 7. The functions T and U are quite simple:

$$T_k(x) = \begin{cases} x + k, \text{ if } k \neq 0, \\ x + k + 1, \text{ otherwise,} \end{cases} \qquad U_k(x) = \begin{cases} ((x - k - 1) \mod 15) + k + 1, \text{ if } x \neq 0, \\ k, \text{ otherwise.} \end{cases}$$

Now it is easy to invert the whole decomposition because we can invert T and U separately. After doing this and simplifying the result, we obtain the algorithm computing π given in Algorithm 2. The corresponding circuit is shown in Figure 7b.

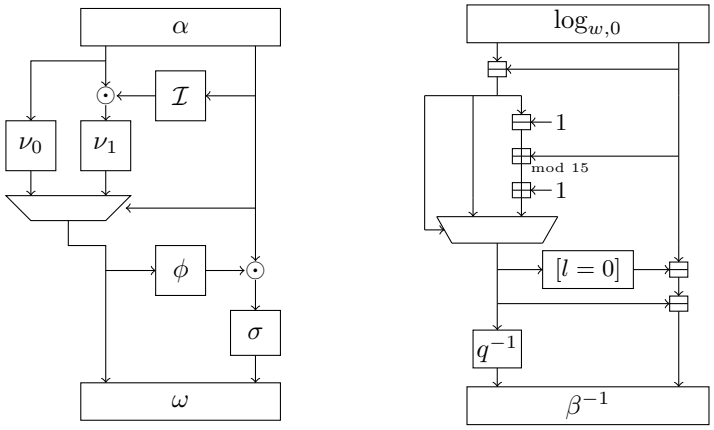
Algorithm 2 Computing the S-Box $y = \pi(x)$ using the first decomposition.

$(l||r) \leftarrow \log_{w,0}(x)$
$l \leftarrow l - r$
if $l = 0$ **then**
  $r \leftarrow r - 1$
else
  $l \leftarrow (l + r - 1) \mod 15 + 1$
end if
$r \leftarrow r - l$
$l \leftarrow q^{-1}(l)$
$y \leftarrow \beta^{-1}(l||r)$
return y

5.3 Another Decomposition of π

The somewhat heavy arithmetic performed in the middle of the computation of π using our new decomposition could point towards the idea that π is built using a pseudo-exponential rather than an exponential substitution. Intuitively, these operations could correspond to a "correction" of the offset.

To explore this idea, we brute-forced all possible preimages z for 0 and all possible bases v for the exponentiation. For each such pseudo-exponential substitution, we ran the following steps.



(a) Old.

(b) New (first).

Figure 7: Old and new decomposition of π. The multiplexers return the left side if the selector is equal to zero and the right side otherwise.

1. Compute[4] $s = \pi \circ \exp_{v,z}$.
2. Compute the differential uniformity and the maximum LAT coefficient of s.
3. If these quantities are high enough then look for vector spaces ℓ_0 and ℓ_1 such that for all (a, b) in $\ell_0 \times \ell_1$, $\mathsf{LAT}[a, b] = 0$.

We restrict ourselves to cases where the maximum LAT coefficient or differential uniformity of s is high. Indeed, this restriction ensures that the functions we study are "weak", which would be the case if it consisted only in few simple operations. In fact, we found that for appropriate choices of v and z, the differential uniformity and the maximum LAT coefficient are respectively above 120 and 80, while typical values for a random permutation would be in the vicinity of 12 and 32 respectively.

The idea behind the search for vector spaces of zeroes is of course to identify possible linear layers to apply before and/or after s in such a way that the resulting permutation has a TU-decomposition. To do so, we used a variant of the algorithm described in [BPU16] to recover part of the linear layer used to whiten a 4-round Feistel Network.

Among the pairs (z, v) such that ℓ_0 and ℓ_1 were large enough, some were such that $\ell_0 = [0, 15]$ and $\ell_1 = \mathcal{V}$, where $\mathcal{V} = \{00, 1a, 20, 3a, 44, 5e, 64, 7e, 8a, 90, aa, b0, ce, d4, ee, f4\}$ (using hexadecimal notations) is the vector space corresponding to the indices of the lines found in the Jackson Pollock representation during the original decomposition of π in [BPU16]. The inverse of the final linear layer ω of this decomposition can therefore be composed with such permutations s to obtain new permutations for which a TU-decomposition exists.

Furthermore, it is possible to find a 4×4 linear layer L_u such that $x \mapsto (L_u \circ U_k)(x) = x \boxplus q'(k)$ for some 4-bit function q'. However, the only pair v, z for which q' is a permutation is $v = w, z = 16$, in which case q' has the following look-up table:

$$q' = \{4, 14, 5, 12, 3, 10, 2, 9, 8, 1, 7, 6, 15, 0, 13, 11\}.$$

Therefore, if we compose s with the inverse of L_u on the right side and the inverse of q' on the left one, we obtain a new permutation whose TU-decomposition is such that

[4] We did consider that π might consist in a pseudo-exponential *preceded* by another permutation, unlike here where it is followed. This line of investigation did not lead to any decomposition.

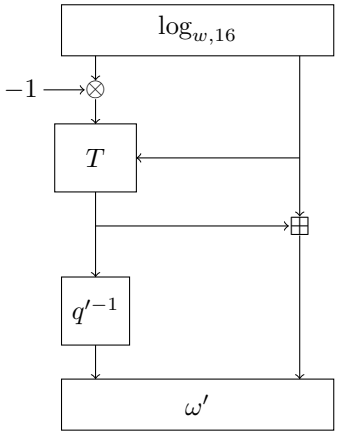
$U_k(x) = x \boxplus k$, which is simpler than in the previous decomposition. Unfortunately, T remains a strange object:

$$T_k((-1) \times x) = \begin{cases} 15, \text{ if } \overline{x} + \overline{k} = 16, \\ x \boxminus 1, \text{ if } \overline{x} + \overline{k} > 16, \\ x, \text{ otherwise,} \end{cases}$$

where $+$ denotes addition without a modulo and where $x \times y = \overline{x} \times \overline{y} \mod 16$. The full codebook of T is provided in Figure 8b. As we can see, each permutation T_k is obtained from the identity by inserting 15 at position $17 - \overline{k}$ (except for T_0), much like 0 is inserted at an arbitrary position in a pseudo-exponential substitution. The complete decomposition is summarized in Algorithm 3 and Figure 8a. It is also implemented, along with the previous decomposition, by the SAGE script in Appendix C. The arithmetic layer of Algorithm 3 is quite sparse: U_k is merely a modular addition and T_k is simply derived from the identity. In fact, it has a differential with probability 1/2.

Algorithm 3 Computing the S-Box $y = \pi(x)$ using a second decomposition.

$(l||r) \leftarrow \log_{w,16}(x)$
$l \leftarrow (-l) \mod 16$
if $l + r = 16$ **then**
  $l \leftarrow 15$
else if $l + r > 16$ **then**
  $l \leftarrow (l - 1) \mod 16$
end if
$r \leftarrow (l + r) \mod 16$
$l \leftarrow q'^{-1}(l)$
$y \leftarrow \omega'(l||r)$
return y



(a) High level view.

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

(b) The code-book of the block-cipher T.

Figure 8: Another decomposition of π.

5.4 Analysis of the New Decompositions

The new decompositions have less "information-heavy" elements - one 4-bit S-Box and one linear layer instead of four and two respectively in the decomposition from [BPU16] (see Figure 7).

The new decompositions contain a central layer consisting of several arithmetic operations whose purpose is not clear. Interestingly, the finite field logarithms or exponentials do not have TU-decompositions, but one appears when the arithmetic layer is added. It is therefore possible that these were added so as to build S-Boxes with better hardware implementations such as the original decomposition [BPU16].

The relationship between our decompositions and the original one is anything but obvious. While the final linear layers are related, the remainder is utterly different: two 4-bit finite field multiplications, 5 random-looking 4-bit S-Boxes and a multiplexer turn out to be functionally equivalent to a pseudo logarithm followed by modular arithmetic and a 4-bit S-Box. The relation between the two representations would have been clearer if the modular arithmetic layer had been placed *before* the pseudo-logarithm: the finite field multiplications then would have been performed on logarithm tables. It is not the case: removing the final linear layer exposes directly a modular addition in the new decompositions.

We also note that the function $\pi^{-1} \circ \log_{w,16}$ is extremely weak from a cryptographic perspective. Its coordinates have an algebraic degree as low as 3, it is differentially 128-uniform and its highest linear bias is $96/128$. We deduce that, in some sense, π is "close" to $\log_{w,16}$. On the other hand, a permutation picked uniformly should yield a random looking S-Box when composed with $\log_{w,16}$.

Our results show that the algebraic structure, whose presence was known thanks to [PUB16], is stronger than hinted in this paper. The permutation π may have been built using one of the known decompositions. However, we think it more likely that each of these decompositions is a consequence of a strong algebraic structure used to design it, probably one related to a finite field exponential. Still this "master decomposition", from which the other would be consequences, remains elusive. Unfortunately, unless the Russian secret service release their design strategy, their exact process is likely to remain a mystery, if nothing else because of the existence of alternative decompositions: which exists by design and which is a mere side-effect of this design?

6 Conclusions

We have analyzed the S-Boxes π and H used by the standard cipher of Russia (Kuznyechik) and Belarus (BelT) respectively.

As our main result, we found a common structure linking these S-Boxes and deduced two new decompositions of π based on a finite field exponentiation. These decompositions improve previous results by Biryukov *et al.*[BPU16] and strengthen the idea that π has a strong algebraic structure hardly compatible with the claims of randomness of the designers. We could not find sensible explanations for using a structure from any of our decompositions as an S-Box.

These results highlight the importance of the "nothing-up-my-sleeve" idea whereby constants and indeed S-Boxes should be chosen in such a way as to convince readers that no weaknesses were introduced in their design. This concept has received a lot of attention recently through the study of so-called *rigid* design procedures for elliptic curves intended to prevent the insertion of backdoors. Using this concept, we can conclude that the S-Box of Kuznyechik and Streebog lacks rigidity.

Acknowledgement

The authors thank the ToSC reviewers for their detailed comments which helped significantly improve the clarity of this paper.

The work of Léo Perrin is supported by the CORE project ACRYPT (ID C12-15-4009992) funded by the Fonds National de la Recherche, Luxembourg. The work of Aleksei Udovenko is supported by the Fonds National de la Recherche, Luxembourg (project reference 9037104).

References

[AA04] Sergey Agievich and Andrey Afonenko. Exponential S-boxes. Cryptology ePrint Archive, Report 2004/024, 2004. http://eprint.iacr.org/2004/024.

[AA38] S.V. Agievich and A.A. Afonenko. [On properties of the exponential S-Boxes]. In *[News of National Academy of Sciences of Belarus]*, volume 1, pages 106–112. National Academy of Sciences of Belarus, 2005 (It is available at http://elib.bsu.by/handle/123456789/24138).

[AGMH40] S.V. Agievich, V.A. Galinskiy, N.D. Mikulich, and Y.S. Harin. BelT [Block Cipher BelT]. In *[Information Security Management]*, volume 6, pages 407–412. 2002 (It is available at http://elib.bsu.by/handle/123456789/24140).

[AY15] Riham AlTawy and Amr M Youssef. A meet in the middle attack on reduced round Kuznyechik. *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, 98(10):2194–2198, 2015.

[BCC10] Céline Blondeau, Anne Canteaut, and Pascale Charpin. Differential properties of power functions. *International Journal of Information and Coding Theory*, 1(2):149–170, 2010.

[BDCBP03] Alex Biryukov, Christophe De Cannière, An Braeken, and Bart Preneel. *A Toolbox for Cryptanalysis: Linear and Affine Equivalence Algorithms*, pages 33–50. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[Bel11] Belarusian State University, National Research Center for Applied Problems of Mathematics and Informatics. "Information technologies. Data protection. Cryptographic algorithms for encryption and integrity control.". State Standard of Republic of Belarus (STB 34.101.31-2011), 2011. http://apmi.bsu.by/assets/files/std/belt-spec27.pdf.

[BP15] Alex Biryukov and Léo Perrin. On Reverse-Engineering S-Boxes with Hidden Design Criteria or Structure. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology – CRYPTO 2015*, volume 9215 of *Lecture Notes in Computer Science*, pages 116–140. Springer Berlin Heidelberg, 2015.

[BPU16] Alex Biryukov, Léo Perrin, and Aleksei Udovenko. *Advances in Cryptology – EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I*, chapter Reverse-Engineering the S-Box of Streebog, Kuznyechik and STRIBOBr1, pages 372–402. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016.

[BR00] Paulo S. L. M. Barreto and Vincent Rijmen. The Whirlpool hashing function. In *First open NESSIE Workshop, Leuven, Belgium*, volume 13, page 14, 2000.

[BS91] Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystems. *Journal of CRYPTOLOGY*, 4(1):3–72, 1991.

[CDL15] Anne Canteaut, Sébastien Duval, and Gaëtan Leurent. Construction of lightweight s-boxes using Feistel and MISTY structures. In Orr Dunkelman and Liam Keliher, editors, *Selected Areas in Cryptography - SAC 2015 - 22nd International Conference, Sackville, NB, Canada, August 12-14, 2015, Revised Selected Papers*, volume 9566 of *Lecture Notes in Computer Science*, pages 373–393. Springer, 2015.

[CF09] Claude Carlet and Keqin Feng. An infinite class of balanced vectorial boolean functions with optimum algebraic immunity and good nonlinearity. In Yeow Meng Chee, Chao Li, San Ling, Huaxiong Wang, and Chaoping Xing, editors, *Coding and Cryptology: Second International Workshop, IWCC 2009, Zhangjiajie, China, June 1-5, 2009. Proceedings*, pages 1–11. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[DD13] V. Dolmatov and A. Degtyarev. GOST R 34.11-2012: Hash Function. RFC 6986 (Informational), August 2013.

[Dev16] The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 7.2)*, 2016. `http://www.sagemath.org`.

[Dol10a] V. Dolmatov. GOST R 34.11-94: Hash Function Algorithm. RFC 5831 (Informational), March 2010. Updated by RFC 6986.

[Dol10b] V. Dolmatov. GOST R 34.12-2015: Block Cipher "Kuznyechik". (Informational), March 2010. Updated by RFC 6986.

[DR02] Joan Daemen and Vincent Rijmen. *The design of Rijndael: AES-the advanced encryption standard*. Springer, 2002.

[DR07] Joan Daemen and Vincent Rijmen. Probability distributions of correlation and differentials in block ciphers. *Journal of Mathematical Cryptology JMC*, 1(3):221–242, 2007.

[Fed12] Federal Agency on Technical Regulation and Metrology. GOST R 34.11-2012: Streebog hash function, 2012. https://www.streebog.net/.

[Fed15] Federal Agency on Technical Regulation and Metrology (GOST). Block Ciphers, 2015. http://www.tc26.ru/en/standard/draft/ENG_GOST_R_bsh.pdf.

[FLY09] Keqin Feng, Qunying Liao, and Jing Yang. Maximal values of generalized algebraic immunity. *Designs, Codes and Cryptography*, 50(2):243–252, 2009.

[HN10] Risto M. Hakala and Kaisa Nyberg. On the nonlinearity of discrete logarithm in $\mathbb{F}_{2^n}$. In Claude Carlet and Alexander Pott, editors, *Sequences and Their Applications – SETA 2010: 6th International Conference, Paris, France, September 13-17, 2010. Proceedings*, pages 333–345. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[KK13] Oleksandr Kazymyrov and Valentyna Kazymyrova. Algebraic Aspects of the Russian Hash Standard GOST R 34.11-2012. *IACR Cryptology ePrint Archive*, 2013:556, 2013.

[Mas69] J. Massey. Shift-register synthesis and bch decoding. *IEEE Transactions on Information Theory*, 15(1):122–127, Jan 1969.

[Mat94] Mitsuru Matsui. Linear cryptanalysis method for des cipher. In *Advances in Cryptology – EUROCRYPT'93*, pages 386–397. Springer, 1994.

[NSA98] National Security Agency National Security Agency. SKIPJACK and KEA Algorithm Specifications, 1998.

[Nyb94] Kaisa Nyberg. Differentially uniform mappings for cryptography. In *Advances in cryptology — Eurocrypt'93*, pages 55–64. Springer, 1994.

[OGK+15] Roman Oliynykov, Ivan Gorbenko, Oleksandr Kazymyrov, Victor Ruzhentsev, Oleksandr Kuznetsov, Yurii Gorbenko, Oleksandr Dyrda, Viktor Dolgov, Andrii Pushkaryov, Ruslan Mordvinov, and Dmytro Kaidalov. A new encryption standard of ukraine: The kalyna block cipher. Cryptology ePrint Archive, Report 2015/650, 2015. http://eprint.iacr.org/2015/650.

[PUB16] Léo Perrin, Aleksei Udovenko, and Alex Biryukov. *Advances in Cryptology – CRYPTO 2016: 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II*, chapter Cryptanalysis of a Theorem: Decomposing the Only Known Solution to the Big APN Problem, pages 93–122. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016.

[RBF08] Vincent Rijmen, Paulo S.L.M. Barreto, and Décio L. Gazzoni Filho. Rotation symmetry in algebraically generated cryptographic substitution tables. *Information Processing Letters*, 106(6):246 – 250, 2008.

[Saa15] Markku-Juhani O Saarinen. STRIBOB: Authenticated encryption from GOST R 34.11-2012 LPS permutation. In *[Mathematical Aspects of Cryptography]*, volume 6, No. 2, pages 67–78. Steklov Mathematical Institute of Russian Academy of Sciences, 2015.

[SB15] Markku-Juhani O. Saarinen and Billy Bob Brumley. *Secure IT Systems: 20th Nordic Conference, NordSec 2015, Stockholm, Sweden, October 19–21, 2015, Proceedings*, chapter WHIRLBOB, the Whirlpool Based Variant of STRIBOB, pages 106–122. Springer International Publishing, Cham, 2015.

[SSA+07] Taizo Shirai, Kyoji Shibutani, Toru Akishita, Shiho Moriai, and Tetsu Iwata. The 128-bit blockcipher clefia. In *Fast software encryption*, pages 181–195. Springer, 2007.

[TCG92] Anne Tardy-Corfdir and Henri Gilbert. A known plaintext attack of feal-4 and feal-6. In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 172–182. Springer Berlin Heidelberg, 1992.

A Generating H Using a Pseudo-Exponential

The following SAGE [Dev16] script generates and prints H. It relies on the Pseudo-Exponential decomposition.

```
from sage.all import *

def applymat(x, mat):
    x = [int(bool(x & (1 << i))) for i in xrange(mat.ncols())][::-1]
    y = mat * vector(GF(2), x)
    return sum(1 << i for i, b in enumerate(y[::-1]) if b)

gamma = Matrix(GF(2), 8, 8, [
    [1, 1, 1, 0, 1, 1, 1, 0],
    [1, 1, 0, 1, 1, 1, 0, 0],
    [1, 0, 1, 1, 1, 0, 0, 0],
    [0, 1, 1, 1, 0, 0, 0, 1],
    [1, 1, 1, 0, 0, 0, 1, 1],
    [1, 1, 0, 0, 0, 1, 1, 1],
    [1, 0, 0, 0, 1, 1, 1, 0],
    [0, 0, 0, 1, 1, 1, 0, 1],
])

X = GF(2).polynomial_ring().gen()
F = GF(2**8, name="a", modulus=X**8 + X**6 + X**5 + X + 1)
g = 0x4a

H = [(F.fetch_int(g)**x).integer_representation() for x in xrange(1, 2**8)]
H.insert(10, 0)
H = [applymat(H[x], gamma) for x in xrange(0, 256)]
print H
```

B Generating H Using LFSR

The following SAGE [Dev16] script is proposed by an anonymous reviewer. It generates and prints H using the LFSR-based decomposition described by the designers.

```
from sage.all import *

toF2 = lambda u: [GF(2)(x) for x in u]
S0 = lfsr_sequence(toF2([1,0,1,1,0,0,1,0,1]),toF2([1,0,0,0,0,0,1,1]),255)
S = [S0[11*i:]+S0[:11*i] for i in range(8)]
H = [ sum(Integer(S[j][i])<<j for j in range(8)) for i in range(255)]
H.insert(10,0)
print H
```

C Generating π Using a Pseudo-Logarithm

The following SAGE [Dev16] script generates and prints π.

```
from sage.all import *

def applymat(x, mat):
    x = [int(bool(x & (1 << i))) for i in xrange(mat.ncols())][::-1]
    y = mat * vector(GF(2), x)
    return sum(1 << i for i, b in enumerate(y[::-1]) if b)
```

```
F8 = GF(2**8, name='a')
toF = F8.fetch_int
fromF = lambda x: x.integer_representation()

# first decomposition

g = toF(2)
exp = [0] + [fromF(g**x) for x in xrange(1, 2**8)]
log = [exp.index(x) for x in xrange(256)]

beta_inv = matrix(GF(2), 8, 8, [
    [1, 0, 0, 0, 0, 0, 0, 0],
    [0, 1, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 1, 1, 1, 0],
    [1, 0, 0, 0, 1, 0, 0, 1],
    [0, 0, 1, 0, 0, 0, 0, 0],
    [0, 1, 0, 0, 0, 1, 1, 0],
    [1, 0, 1, 0, 0, 0, 1, 1],
    [0, 0, 0, 1, 0, 0, 0, 0],
])
q_inv = [14, 15, 1, 6, 9, 7, 5, 13, 10, 2, 3, 8, 0, 4, 12, 11]

sbox = []
for x in xrange(256):
    x = log[x]
    l, r = x >> 4, x & 0xf

    l = (l - r) % 16
    if l != 0:
        l = (l + r - 1) % 15 + 1
    else:
        r = (r - 1) % 16
    r = (r - l) % 16

    l = q_inv[l]
    y = applymat((l << 4) | r, beta_inv)
    sbox.append(y)
print "S-Box:", sbox



# second decomposition

pseudo_exp = [fromF(g**x) for x in xrange(1, 2**8)]
pseudo_exp.insert(16, 0)
pseudo_log = [pseudo_exp.index(x) for x in xrange(256)]

omega_prime = Matrix(GF(2), 8, 8, [
    1,0,1,0,0,0,0,0,
    0,1,0,0,0,0,0,0,
    0,0,0,0,1,1,1,0,
    1,0,1,0,1,0,0,1,
    1,0,0,0,0,0,0,0,
    0,1,0,0,0,1,1,0,
    0,0,1,0,0,0,1,1,
    0,0,0,1,0,0,0,0,
```

```
])
q2_inv = [13, 9, 6, 4, 0, 2, 11, 10, 8, 7, 5, 15, 3, 14, 1, 12]

sbox2 = []
for x in xrange(0, 256):
    y = pseudo_log[x]
    l, r = y >> 4, y & 0xf
    l = (-l) % 16
    if l+r == 16:
        l = 0xf
    elif l+r > 16:
        l = (l-1) % 16
    r = (r + l) % 16
    l = q2_inv[l]
    y = applymat((l << 4) | r, omega_prime)
    sbox2.append(y)
print "S-Box2: ", sbox2
```