

Partitions in the S-Box of Streebog and Kuznyechik

Léo Perrin

▶ **To cite this version:**

Léo Perrin. Partitions in the S-Box of Streebog and Kuznyechik. IACR Transactions on Symmetric Cryptology, 2019, 2019 (1), pp.302-329. 10.13154/tosc.v2019.i1.302-329 . hal-02396814

HAL Id: hal-02396814
https://inria.hal.science/hal-02396814

Submitted on 6 Dec 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Partitions in the S-Box of Streebog and Kuznyechik*

Léo Perrin

Inria, France
leo.perrin@inria.fr

Abstract. Streebog and Kuznyechik are the latest symmetric cryptographic primitives standardized by the Russian GOST. They share the same S-Box, π, whose design process was not described by its authors. In previous works, Biryukov, Perrin and Udovenko recovered two completely different decompositions of this S-Box.
We revisit their results and identify a third decomposition of π. It is an instance of a fairly small family of permutations operating on $2m$ bits which we call TKlog and which is closely related to finite field logarithms. Its simplicity and the small number of components it uses lead us to claim that it has to be the structure intentionally used by the designers of Streebog and Kuznyechik.
The $2m$-bit permutations of this type have a very strong algebraic structure: they map multiplicative cosets of the subfield $\mathrm{GF}(2^m)^*$ to additive cosets of $\mathrm{GF}(2^m)^*$. Furthermore, the function relating each multiplicative coset to the corresponding additive coset is always essentially the same. To the best of our knowledge, we are the first to expose this very strong algebraic structure.
We also investigate other properties of the TKlog and show in particular that it can always be decomposed in a fashion similar to the first decomposition of Biryukov et al., thus explaining the relation between the two previous decompositions. It also means that it is always possible to implement a TKlog efficiently in hardware and that it always exhibits a visual pattern in its LAT similar to the one present in π.
While we could not find attacks based on these new results, we discuss the impact of our work on the security of Streebog and Kuznyechik. To this end, we provide a new simpler representation of the linear layer of Streebog as a matrix multiplication in the exact same field as the one used to define π. We deduce that this matrix interacts in a non-trivial way with the partitions preserved by π.

Keywords: Boolean functions · Kuznyechik · Streebog · Reverse-Engineering · Partitions · Cosets · TKlog

1 Introduction

Many symmetric primitives rely on S-Boxes as their unique source of non-linearity, including the AES [AES01]. Such objects are small functions mapping $\mathbb{F}_2^m$ to $\mathbb{F}_2^n$ which are often specified via their look-up tables.

Their choice is crucial as both the security and the efficiency of the primitive depends heavily on their properties. For example, a low *differential uniformity* [Nyb94] implies a higher resilience against differential attacks [BS91a, BS91b]. On the other hand, the existence of a simple decomposition greatly helps with an efficient bitsliced or hardware implementation [LW14, CDL16]. Thus, algorithm designers are expected to provide detailed explanation about their choice of S-Box. Each cipher that was published at a cryptography or security conference has provided such explanations.

*This paper will appear in the IACR Transactions on Symmetric Cryptology.

There are two prominent S-Boxes for which this information has not been provided. The first is the so-called "F-table" of Skipjack [U.S98], a lightweight block cipher designed by the American National Security Agency (NSA). The second is π, the 8-bit permutation used by the Russian standard hash function (nicknamed Streebog [Fed12]) and block cipher (nicknamed Kuznyechik [Fed15]), as well as the first version of the CAESAR candidate STRIBOBr1 [Saa14] (which later changed its components for those of Whirlpool [SB15]).

While Streebog and Kuznyechik were first published as national standards in Russia (GOST), they have since been included in other standards. For instance, both have been included by the IETF as RFC 6986 [DD13] and RFC 7801 [Dol16] respectively. ISO/IEC is also in the process of adding Kuznyechik to their list of standard block ciphers, namely standard 18033-3[1].

S-Boxes have been found to be potential vehicules for the insertion of a backdoor in a symmetric algorithm. In 1997, Rijmen and Preneel suggested an S-Box generation strategy which ensured that a high probability linear transition existed [RP97]. The idea was that only the designer would be able to know about this linear approximation but this claim was later proven wrong [WBDY98]. Later, Paterson designed a backdoored variant of the DES with modified S-Boxes [Pat99]. His overall approach was recently refined by Bannier et al. [BBF16] to build a block cipher which preserves a partition of the plaintext space independently from the key.

In this context, cryptanalysts have tried to reverse-engineer the structure of poorly specified S-Boxes. The first such attempt occurred in the late 1970's, shortly after the publication of the DES: Hellman et al. identified some patterns in the S-Boxes of this block cipher [HMS+76]. Much more recently, Biryukov et al. devised new tools for this purpose. For example, a statistical analysis of the differential and linear properties allowed them to show that the S-Box of Skipjack diplayed a higher resilience against linear attacks [Mat94] than expected [BP15].

More importantly in our case, they provided the first decomposition of the Russian S-Box, π, in [BPU16a, BPU16b]. The corresponding structure operates on two branches, much like a Feistel or Misty structure. It is however much more complex than both of them as it involves finite field multiplications in $\mathrm{GF}(2^4)$ and a multiplexer. Later, Perrin and Udovenko found discrete logarithm-based decompositions of this component [PU16]. They are very different from the previous decomposition but remain somewhat unsatisfactory due to the complex "arithmetic layer" they use. Their authors concluded that "We could not find sensible explanations for using a structure from any of our decompositions as an S-Box." In fact, the existence of these new structures raised more questions than it answered, although they "strengthen the idea that π has a strong algebraic structure hardly compatible with the claims of randomness of the designers" [PU16]. In the end, Perrin and Udovenko conjectured the existence of a "master decomposition" of which the decompositions of both [BPU16a] and [PU16] would be mere side effects.

Our Contribution. We show that the intuition in [PU16] was correct and present what we claim to be said "master decomposition". It holds that π, the S-Box used by the last two Russian standards, operates as follows:

$$\begin{cases} \pi(0) & = \kappa(0) \\ \pi(\alpha^{0+17j}) & = \kappa(16-j) \\ \pi(\alpha^{i+17j}) & = \kappa(16-i) \oplus \left(\alpha^{17}\right)^{s(j)} \quad \text{for} \quad i > 0 \ , \end{cases}$$

where α is a root of the primitive polynomial defining the finite field $\mathrm{GF}(2^8)$, where s is a permutation of $\mathbb{Z}/15\mathbb{Z}$, and where $\kappa : \mathbb{F}_2^4 \to \mathrm{GF}(2^8)$ is an affine function such that

[1] The corresponding "amendment" to standard 18033-3 can be previewed on the ISO website at https://www.iso.org/standard/73205.html.

any element $x \in \mathrm{GF}(2^8)$ can be written as $x = x_4 \oplus \kappa(x_\kappa) \oplus \kappa(0)$ with $x_4 \in \mathrm{GF}(2^4)$ and $x_\kappa \in \mathbb{F}_2^4$. We also use "$+$" and "$-$" to denote integer addition and substration, and "$\oplus$" denotes the addition in the finite field.

We generalize this new structure by allowing κ and s to be picked in appropriate sets and call the resulting class of permutation *TKlog*. This new structure allows us to easily explain a particular property of π: it maps the partition of $\mathrm{GF}(2^8)$ into multiplicative cosets of $\mathrm{GF}(2^4)^*$ to the partition of $\mathrm{GF}(2^8)$ into additive cosets of $\mathrm{GF}(2^4)^*$.[2] Furthermore, the restriction of π to each independent multiplicative coset is always the same simple function. Thus, not only does it map a simple partition of $\mathrm{GF}(2^8)$ to another one, it does so in a very straightforward way.

We also prove that such permutations can always be written in a fashion similar to the decomposition of [BPU16a] so our new decomposition provides the missing link between the first decomposition of [BPU16a] and the logarithm-based decomposition of [PU16]. Using counting arguments, we show that the size of the set of the TKlogs operating on 8 bits and the number of affine 8-bit permutations are of comparable magnitudes, meaning that the probability that a random permutation is a TKlog instance is negligible. We therefore claim that the presence of this structure in π is a deliberate choice by its designers. Using experimental arguments, we propose a simple generation algorithm of which π would be a typical output.

Finally, we remark that the linear layer of Streebog is an MDS matrix with coefficients in $\mathrm{GF}(2^8)$ where the primitive polynomial used to define the representation of its elements is actually the same as the one used to define π as a TKlog. Thus, the cosets interacting with π also interact with the linear layer of Streebog. We provide a first discussion of the consequences of the new structure in π in terms of security but leave their exploitation in a cryptanalysis as an open problem.

Outline. Section 2 recalls the background necessary, both in terms of mathematics and in terms of previous results. The TKlog, its relationship with π and its partition-preserving property are presented in Section 3. We show that the TKlog is the "missing link" between [BPU16a] and [PU16] and list several consequences of this fact in Section 4. Then, we investigate the consequences of the fact that π is a TKlog for the higher level primitives themselves in Section 5. We also present a new representation of the linear layer of Streebog which can be of independent intrerest. Finally, Section 6 concludes this paper.

2 Background

2.1 Notations and Basic Definitions

Finite Fields. There exists, up to isomorphisms, a unique finite field with 2^n elements which we denote $\mathrm{GF}(2^n)$. We use "$\oplus$" to denote the addition in the field and $a \odot b$ or ab to denote the product of $a, b \in \mathrm{GF}(2^n)$. For all $x \in \mathrm{GF}(2^n)$, it holds that $x^{2^n} \oplus x = 0$.

If $n = 2m$ then we define the trace from $\mathrm{GF}(2^{2m})$ to $\mathrm{GF}(2^m)$ as the function $\mathrm{Tr}_m : \mathrm{GF}(2^{2m}) \to \mathrm{GF}(2^m)$ such that $\mathrm{Tr}_m(x) = x^{2^m} \oplus x$. For any $\beta \in \mathrm{GF}(2^{2m})$ such that $\mathrm{Tr}_m(\beta) = 1$, the elements of $\mathrm{GF}(2^{2m})$ can all be decomposed in a unique way into $a\beta \oplus b$, where a and b are in $\mathrm{GF}(2^m)$. The bijection mapping $x \in \mathrm{GF}(2^{2m})$ to $(a, b) \in \mathrm{GF}(2^m)^2$ such that $x = a\beta \oplus b$ is denoted Split_β, so that $\mathsf{Split}_\beta(a\beta \oplus b) = (a, b)$. It is a linear function.

We use S^* to denote a set S minus the element 0. Even though $\mathrm{GF}(2^m)^*$ is not an additive group, we call $\{a \oplus x, x \in \mathrm{GF}(2^m)^*\}$ an additive coset of $\mathrm{GF}(2^m)^*$ for the sake

[2] Strictly speaking, $\mathrm{GF}(2^4)^*$ is not an additive subgroup of $\mathrm{GF}(2^8)$. Thus, the set $\{a \oplus x, x \in \mathrm{GF}(2^4)^*\}$ is not formally an additive coset. However, for the sake of simplicity, we will slightly abuse this term and call such sets "additive cosets of $\mathrm{GF}(2^4)^*$".

of simplicity. If S is a set and a is a constant, we denote $a \oplus S = \{a \oplus x, x \in S\}$ and $aS = a \odot S = \{a \odot x, x \in S\}$.

Binary Strings as Ring Elements. The field $\mathrm{GF}(2^n)$ can be identified with $\mathbb{F}_2[X]/p(X)$ for some irreducible polynomial p of degree n. If α is a root of p then we can represent all the elements of $\mathrm{GF}(2^n)$ as $\sum_{i=0}^{n-1} x_i\alpha^i$, where $x_i \in \mathbb{F}_2$. A binary string $(x_0, ..., x_{n-1}) \in \mathbb{F}_2^n$ is therefore naturally interpreted as $\sum_{i=0}^{n-1} x_i\alpha^i \in \mathrm{GF}(2^n)$ in much the same way that it can also be interpreted as $\sum_{i=0}^{n-1} x_i 2^i \in \mathbb{Z}/2^n\mathbb{Z}$. In this case, the binary representation of $a \oplus b$ for $a, b \in \mathrm{GF}(2^n)$ is indeed the XOR of the binary representations of a and b.

Logarithms. For all $x \in \mathrm{GF}(2^n)^*$, the logarithm $\log_\alpha(x)$ is the integer of $\mathbb{Z}/(2^n-1)\mathbb{Z}$ such that $\alpha^{\log_\alpha(x)} = x$. Such a function is not a permutation because it is not defined in 0. This problem can be solved in different ways. In [HN10], Hakala and Nyberg study the function which we denote $\log_\alpha^{\mathrm{HN}}$ while Feng et al. in [FLY09] introduced another variant, a special case of which is a permutation of $\mathrm{GF}(2^n)$ and which we call $\log_\alpha^{\mathrm{FLY}}$. These two functions map $\mathrm{GF}(2^n)$ to $\mathbb{Z}/2^n\mathbb{Z}$ and are defined by

$$\log_\alpha^{\mathrm{HN}}(x) = \begin{cases} 2^n - 1 & \text{if } x = 0\ , \\ 0 & \text{if } x = 1\ , \\ \log_\alpha(x) & \text{if } x \notin \{0,1\}\ , \end{cases} \quad \text{and} \quad \log_\alpha^{\mathrm{FLY}}(x) = \begin{cases} 0 & \text{if } x = 0\ , \\ 2^n - 1 & \text{if } x = 1\ , \\ \log_\alpha(x) & \text{if } x \notin \{0,1\}\ . \end{cases}$$

In other words $\log_\alpha^{\mathrm{FLY}}$, is a variant of $\log_\alpha^{\mathrm{HN}}$ where the outputs of 0 and 1 are swapped. We remark that the built-in discrete logarithm in SAGE [Dev17] actually implements $\log_\alpha^{\mathrm{FLY}}$ on $\mathrm{GF}(2^n)^*$. Another variant of the logarithm called "pseudo-logarithm" was introduced in [PU16] when investigating π but we will not use it here.

Boolean Functions. Let $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ be a function. The *Linear Approximations Table (LAT)* or *Walsh transform* of F is the $2^n \times 2^m$ matrix $\mathcal{W}_F$ such that

$$\mathcal{W}_F(a,b) = \sum_{x \in \mathbb{F}_2^n} (-1)^{a \cdot x \oplus b \cdot F(x)}\ ,$$

where $a \cdot b$ is the usual scalar product in $\mathbb{F}_2^n$. The maximum value of $|\mathcal{W}_F(a,b)|$ for $b \neq 0$ is the *linearity* of F. The *Difference Distribution Table (DDT)* of F is the $2^n \times 2^m$ matrix δ_F such that

$$\delta_F(a,b) = \#\left\{x \in \mathbb{F}_2^n, F(x \oplus a) \oplus F(x) = b\right\}\ .$$

The maximum value of $\delta_F(a,b)$ for $a \neq 0$ is the *differential uniformity* of F. If A and B are affine permutations of $\mathbb{F}_2^m$ and $\mathbb{F}_2^n$ respectively, then F is *affine equivalent* to $G = B \circ F \circ A$. Furthermore, let L_A and L_B be the linear parts of A and B. Then the LAT of G is:

$$\mathcal{W}_G(a,b) = \mathcal{W}_F\left((L_A^{-1})^T(a), L_B^T(b)\right)\ .$$

Affine-equivalence can be generalized into CCZ-Equivalence [CCZ98]: two functions $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ and $G : \mathbb{F}_2^n \to \mathbb{F}_2^m$ are *CCZ-equivalent* if there exists an affine permutation $\mathcal{A}$ of $\mathbb{F}_2^n \times \mathbb{F}_2^m$ such that

$$\left\{(x, F(x)), x \in \mathbb{F}_2^n\right\} \ = \ \mathcal{A}\big(\left\{(x, G(x)), x \in \mathbb{F}_2^n\right\}\big)\ .$$

This form of equivalence is known to preserve, among other things, the differential uniformity and the linearity.

2.2 On the S-Box π

While the specification of both Streebog and Kuznyechik have always been public, a complete design rationale has not been provided. In particular, its designers gave very little information about the design method of their S-Box. Its look-up table is given in Table 2 in Appendix 8.

This lack of information prompted academics to try and reverse-engineer this S-Box. We present the two decompositions that were found by Biryukov, Perrin and Udovenko further below but first we summarize the information that the designers did provide.

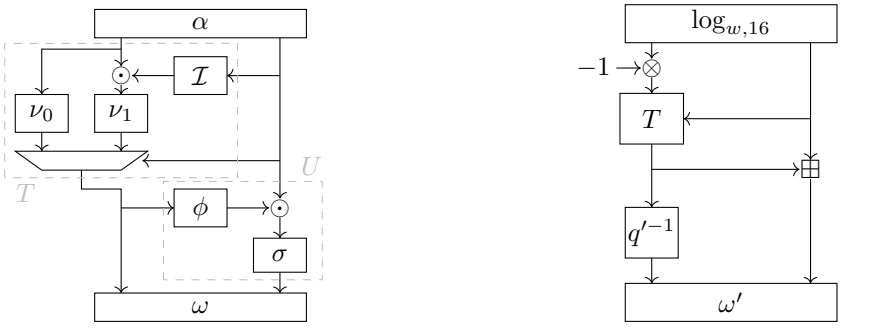
2.2.1 From the Designers

At RusCrypto'13 [Shi13], Shishkin gave a talk presenting the design principles of their upcoming block cipher (Kuznyechik was standardized in 2015). While they considered using S-Boxes from a known class of good S-Boxes, their prefered design approach was different. It is summarized in the following (translated) quote from their slides.

> [The properties of S-Boxes designed via a] Random search with a specified parameter restriction
>
> - are not optimal when considering the aggregate of the values of the basic cryptographic properties
> - do not have a pronounced analytical structure

In other words, S-Boxes randomly generated trade their non-optimal cryptographic properties (differential uniformity, etc.) for the absence of an analytical structure which could be used by an attacker. Further in their presentation, they state that the number of bit operations needed to implement the S-Box should be minimized so as to help with both hardware and vectorized implementations. Those design criteria make sense; in fact, many algorithms use S-Boxes chosen with a similar rationale such as, for example, CLEFIA [SSA+07].

However, up until the decomposition found by Biryukov et al. [BPU16a] (see Section 2.2.2) no efficient implementation strategy was known for π. Furthermore, this permutation was shown to be somewhat close to a finite field logarithm [PU16] (see Section 2.3), which seems at odd with the claimed lack of analytical structure.



(a) *The TU-decomposition.* **(b)** *The log-based decomposition.*

Figure 1: *The decompositions of π in the literature.*

2.2.2 TU-Decomposition

Because the design method of π was not published, Biryukov et al. tried to look for additional design criteria or even for hidden structure using (and improving) techniques

from [BP15]. Their results are presented in [BPU16a, BPU16b]. They showed that π has a TU-decomposition, i.e. that it is affine-equivalent to a permutation $(x, y) \mapsto \big(T_y(x), U_{T_y(x)}(y)\big)$ where both T_y and U_x are 4-bit permutations for all $(x, y) \in (\mathbb{F}_2^4)^2$. They further provided a decomposition of both T and U, so that the overall structure of π is as described in Figure 1a where ν_0, ν_1 and σ are 4-bit permutations, ϕ is a 4-bit function such that $\phi(x) \neq 0$ and $\mathcal{I}$ is the multiplicative inversion in $\mathrm{GF}(2^4)$. The multiplexer selects the output of ν_0 if the right branch is equal to 0 and the output of ν_1 otherwise. The last components are α and ω, two 8-bit linear permutations which are linked by the following relation:

$$\left\{\alpha^{-1}(x||0), x \in \mathbb{F}_2^4\right\} \;=\; \left\{\omega(0||x), x \in \mathbb{F}_2^4\right\} \;. \tag{1}$$

Biryukov et al. started by composing π with an 8-bit linear layer L^* applied at both the input and the output. The fact that the same function is applied in both cases is a consequence of the relation in Equation (1). They recovered L^* using visual patterns in the LAT of π.

Another remarkable property was noted in [PU16]: ν_0 is affine-equivalent to a discrete logarithm in $\mathrm{GF}(2^4)$.

2.3 Discrete Logarithm

While investigating the S-Box of the Belarussian standard block cipher BelT [Bel11], Perrin and Udovenko found a completely different decomposition of π [PU16]. Indeed, they showed that it had the structure summarized in Figure 1b, i.e. that it was the composition of:

- a "pseudo-logarithm", i.e. a permutation obtained by inverting a "pseudo-exponential" which is itself built as a sequence $[\alpha^0, \alpha^1, \alpha^2, ..., \alpha^{j-1}, 0, \alpha^j, ..., \alpha^{2^n-2}]$ for a generator α of $\mathrm{GF}(2^8)^*$ and some preimage j for 0;
- a layer of modular arithmetic operations which they could not simplify,
- a 4-bit permutation q'^{-1}, and
- an 8-bit linear permutation ω'.

This decomposition uses the primitive polynomial $p_{\min}(X) = X^8 \oplus X^4 \oplus X^3 \oplus X^2 \oplus 1$ to define the finite field used in its logarithm. It is the first primitive polynomial of degree 8 in the lexicographic order as can be seen e.g. in Table C of [LN97]. It is also the default polynomial used when building a finite field of size 2^8 in SAGE [Dev17].

2.3.1 Relations Between the Decompositions

These decompositions are functionally equivalent since they both correspond to the same permutation π, and yet they have little to nothing in common. When evaluating the TU-decomposition, the input first goes through a linear layer mapping $\mathbb{F}_2^8$ to $(\mathrm{GF}(2^4))^2$ and then undergoes $(a, b) \mapsto (a/b, b)$, except if $b = 0$. On the other hand, evaluating the log-based decomposition first requires using a variant of the discrete logarithm. What is the relation between these operations?

We thus find it surprising that both decompositions exist. Furthermore, none of them seems like a natural choice for building an S-Box. These observations led Perrin and Udovenko to the following conclusion [PU16]:

> we think it more likely that [the TU-decomposition and log-based decomposition are] a consequence of a strong algebraic structure used to design [π], probably one related to a finite field exponential. Still this "master decomposition", from which the others 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?

In the next section, we present what we believe to be this "master decomposition". It relies on a discrete logarithm, which relates it to the second decomposition, and it turns out that a TU-decomposition identical to the one of [BPU16a] is always possible for permutations with a similar structure. We argue that this new decomposition is likely to be the one intended by the designers later in Section 5.1.

3 The TKlog

In this section, we introduce a new type of permutation which we call TKlog of which π turns out to be a particular case. As Stribog and Kuznyechik were both designed by the "TK-26"[3], we used the letters "TK" to name this structure. It has log-like properties, though mapping $\mathrm{GF}(2^{2m})$ to itself rather than to $\mathbb{Z}/2^{2m}\mathbb{Z}$, hence the "log" part of the name. More precisely, it maps the partition of $\mathrm{GF}(2^{2m})$ into multiplicative cosets of $\mathrm{GF}(2^m)^*$ to its partition into additive cosets of $\mathrm{GF}(2^m)$ and its restriction to each multiplicative coset is essentially the same for all cosets. We define this structure in Section 3.1 and present the details of this partition-preserving property in Section 3.2.

3.1 The TKlog Permutation Structure

The TKlog. Let $\mathrm{GF}(2^{2m}) = \mathbb{F}_2[X]/p(X)$ be a finite field of even degree defined by a primitive polynomial p. The multiplicative subgroup $\mathrm{GF}(2^{2m})^*$ is cyclic and generated by α which is such that $p(\alpha) = 0$. In this context, α^{2^m+1} is a generator of the multiplicative subgroup of the subfield $\mathrm{GF}(2^m)$.

Definition 1 (TKlog)**.** A TKlog is permutation operating on $\mathrm{GF}(2^{2m}) = \mathbb{F}_2[X]/p(X)$ for some primitive polynomial p with root α. It is parametrized by:

- an affine function $\kappa : \mathbb{F}_2^m \to \mathrm{GF}(2^{2m})$ such that any $x \in \mathrm{GF}(2^{2m})$ can be written $x = x_m \oplus \kappa(x_\kappa) \oplus \kappa(0)$ for some $x_m \in \mathrm{GF}(2^m)$ and $x_\kappa \in \mathbb{F}_2^m$,[4] and
- a permutation s of $\mathbb{Z}/(2^m - 1)\mathbb{Z}$.

The corresponding TKlog is denoted $\mathscr{T}_{\kappa,s}$ and it works as follows:

$$\begin{cases} \mathscr{T}_{\kappa,s}(0) & = \kappa(0)\ , \\ \mathscr{T}_{\kappa,s}\left((\alpha^{2^m+1})^j\right) & = \kappa(2^m - j), \quad \text{for } \ 1 \leq j \leq 2^m - 1\ , \\ \mathscr{T}_{\kappa,s}\left(\alpha^{i+(2^m+1)j}\right) & = \kappa(2^m - i) \oplus \left(\alpha^{2^m+1}\right)^{s(j)}, \quad \text{for } \ 0 < i, 0 \leq j < 2^m - 1\ , \end{cases}$$

where the fact that $1 \leq j \leq 2^m - 1$ rather than $0 \leq j < 2^m - 1$ when $x \in \mathrm{GF}(2^m)^*$ comes from the implicit use of Feng et al.'s logarithm. Algorithm 3 (in Appendix 9) evaluates such a permutation.

[3] The official name of this organisation is "Технический Комитет По Стандартизации Криптографическая Защита Информации", which stands for "Technical Committee for Standardization of cryptographic information protection"

[4] Equivalently, we can write that the linear part of κ mapping $y \in \mathbb{F}_2^m$ to $\kappa(y) \oplus \kappa(0) \in \mathrm{GF}(2^{2m})$ maps $\mathbb{F}_2^m$ to a vector space in direct sum with the subfield $\mathrm{GF}(2^m)$.

To convince ourselves that a TKlog instance as defined above is indeed a permutation, we define its functional inverse, TKexp. It uses $\Phi_\kappa : \mathrm{GF}(2^{2m}) \to \mathbb{F}_2^m \times \mathrm{GF}(2^m)$ which is the affine permutation such that

$$\Phi_\kappa\big(\kappa(x) \oplus v\big) = (x, v)$$

for $(x, v) \in \mathbb{F}_2^m \times \mathrm{GF}(2^m)$. The TKexp works as follows:

$$\begin{cases} \left(\mathscr{T}_{\kappa,s}^{-1} \circ \Phi_\kappa^{-1}\right)(0,0) & = 0\ , \\ \left(\mathscr{T}_{\kappa,s}^{-1} \circ \Phi_\kappa^{-1}\right)(k,0) & = \alpha^{(2^m+1)(2^m-k)}\ , \\ \left(\mathscr{T}_{\kappa,s}^{-1} \circ \Phi_\kappa^{-1}\right)(k,v) & = \alpha^{2^m-k+(2^m+1)s^{-1}(j)} \text{ where } j = \log_\alpha^{\mathrm{FLY}}(v)/(2^m+1)\ . \end{cases}$$

Alternatively, it can be evaluated using Algorithm 4 (in Appendix 9).

On the Substraction. The integer substraction in the input of κ is made necessary by the fact that $i \in \{1, ..., 2^m\}$, so that the binary representation of i does not always fit in m bits. We therefore need a small function that maps $\{1, ..., 2^m\}$ to $\{0, ..., 2^m - 1\}$ since the case $i = 0$ is handled separately. A natural choice would obviously be $i \mapsto i - 1$ but, in the case, we would need a different function when $i = 0$. Indeed, since a FLY logarithm is used, we have $j \in \{1, ..., 2^m - 1\}$ when $i = 0$. Thus, we would need to compose κ with two different functions depending on whether $i = 0$. On the other hand, the function $i \leftarrow 2^m - i$ maps both $\{1, ..., 2^m\}$ to $\{0, ..., 2^m - 1\}$ and $\{1, ..., 2^m - 1\}$ to itself, meaning that it can be used in both cases.

We could not find such a simple function when $\log_\alpha^{\mathrm{HN}}$ is used.

The Particular case of π. The S-Box π is a TKlog operating on 8 bits. For implementation purposes, we identify $\mathrm{GF}(2^8)$ with $\mathbb{F}_2^8$ using the method we described in Section 2.1. When written as TKlog instance, π uses the following components:

- the finite field $\mathrm{GF}(2^8) = \mathbb{F}_2[X]/p_{\min}(X)$ where $p_{\min}(X) = X^8 \oplus X^4 \oplus X^3 \oplus X^2 \oplus 1$ and its root α;
- an affine function κ mapping $\mathbb{F}_2^4$ to $\mathbb{F}_2^8$ such that $\kappa(0) = \texttt{0xFC}$ with a linear part Λ defined by
$$\Lambda(1) = \texttt{12}\ , \Lambda(2) = \texttt{26}\ , \Lambda(4) = \texttt{24}\ , \Lambda(8) = \texttt{30}\ ,$$
where the numbers are written in hexadecimal form and where the linear function Λ verifies $\big\{x_4 \oplus \Lambda(y), x_4 \in \mathrm{GF}(2^4), y \in \mathbb{F}_2^4\big\} = \mathrm{GF}(2^8)$; and
- a permutation s of $\mathbb{Z}/15\mathbb{Z}$ defined in Table 1.

Table 1: *The look up table of the permutation s of $\mathbb{Z}/15\mathbb{Z}$.*

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$s(x)$	0	12	9	8	7	4	14	6	5	10	2	11	1	3	13

The evaluation of π using this structure is summarized in Algorithm 1. An implementation of π based on this algorithm is given as a SAGE [Dev17] script in Appendix 12. It is noteworthy that the default logarithm function in SAGE is $\log_\alpha^{\mathrm{FLY}}$, which simplifies the implementation of the function.

We actually obtained the TKlog structure by first decomposing π and then generalizing the structure we found. A summary of our reverse-engineering process is given in Appendix 10.

Algorithm 1 A new decomposition of π.

function $\pi(x \in \mathrm{GF}(2^8))$
  if $x = 0$ **then**
    return $\kappa(0)$
  else
    $k = \log_\alpha^{\mathrm{FLY}}(x)$
    $i \leftarrow k \mod 17$; $j \leftarrow \lfloor k/17 \rfloor$ ▷ $x = \alpha^{i+17j}$
    if $i = 0$ **then** ▷ Case where $x \in \mathrm{GF}(2^4)$
      return $\kappa(16 - j)$ ▷ $i = 0$ so $j \in \{1, ..., 15\}$
    else
      return $\kappa(16 - i) \oplus (\alpha^{17})^{s(j)}$ ▷ $i \neq 0$ so $16 - i \neq 16$
    end if
  end if
end function

3.2 Cosets to Cosets

3.2.1 A Partition-Preserving Property

Recall that $\mathrm{GF}(2^m)^*$ is the field of size 2^m minus 0 and that it is contained in $\mathrm{GF}(2^{2m})^*$. Let α be a multiplicative generator of $\mathrm{GF}(2^{2m})^*$ so that α^{2^m+1} is a multiplicative generator of $\mathrm{GF}(2^m)^*$. The field $\mathrm{GF}(2^{2m})$ can be written in two different ways using the multiplicative cosets of $\mathrm{GF}(2^m)^*$ on one hand and the additive cosets of $\mathrm{GF}(2^m)^*$ on the other.

- All the elements in $\mathrm{GF}(2^{2m})^*$ can be written as $\alpha^{i+(2^m+1)j} = \alpha^i \left(\alpha^{2^m+1}\right)^j$, so that

$$\mathrm{GF}(2^{2m}) \;=\; \{0\} \cup \left(\bigcup_{i=0}^{2^m} \alpha^i \odot \mathrm{GF}(2^m)^* \right) \;=\; \mathrm{GF}(2^m) \cup \left(\bigcup_{i=1}^{2^m} \alpha^i \odot \mathrm{GF}(2^m)^* \right) .$$

- As $\mathrm{GF}(2^m)$ is a vector space of dimension m, there exists a vector space W of elements of $\mathrm{GF}(2^{2m})$ of dimension m such that $\mathrm{GF}(2^{2m})$ is the direct sum of W and $\mathrm{GF}(2^m)$. In this case, we can write

$$\mathrm{GF}(2^{2m}) \;=\; \bigcup_{w \in W} w \oplus \mathrm{GF}(2^m) \;=\; W \cup \left(\bigcup_{w \in W} w \oplus \mathrm{GF}(2^m)^* \right) .$$

Both W and $\mathrm{GF}(2^m)$ are vector spaces of dimension m and, in each decomposition, 2^m cosets of $\mathrm{GF}(2^m)^*$ are used. It is therefore possible to map these decompositions from one to the other. It is precisely what a TKlog does. More formally, the following theorem holds.

Theorem 1 (Cosets to Cosets)**.** *Let $\mathscr{T}_{\kappa,s} : \mathrm{GF}(2^{2m}) \to \mathrm{GF}(2^{2m})$ be a valid TKlog instance. Then the following equalities are always true:*

$$\begin{cases} \mathscr{T}_{\kappa,s}(\mathrm{GF}(2^m)) & = \kappa\left(\mathbb{F}_2^m\right) \\ \mathscr{T}_{\kappa,s}\left(\alpha^i \odot \mathrm{GF}(2^m)^*\right) & = \kappa(2^m - i) \oplus \mathrm{GF}(2^m)^* \ , \ \forall i \neq 0 \ . \end{cases}$$

Corollary 1 (Vector spaces to affine spaces)**.** *Let $\mathscr{T}_{\kappa,s} : \mathrm{GF}(2^{2m}) \to \mathrm{GF}(2^{2m})$ be a valid TKlog instance. Then it always holds that*

$$\mathscr{T}_{\kappa,s}(\mathrm{GF}(2^m)) = \kappa\left(\mathbb{F}_2^m\right) \quad \textit{and} \quad \mathscr{T}_{\kappa,s}(\alpha^{2^m} \odot \mathrm{GF}(2^m)) = \kappa(0) \oplus \mathrm{GF}(2^m) \ ,$$

where all the spaces involved in these equalities are of dimension m. Furthermore,

$$\begin{aligned}\mathrm{GF}(2^{2m}) &= \left\{x \oplus y,\ x \in \mathrm{GF}(2^m),\ y \in \alpha^{2^m} \odot \mathrm{GF}(2^m)\right\} \\ &= \left\{x \oplus y,\ x \in \kappa(0) \oplus \mathrm{GF}(2^m),\ y \in \kappa(0) \oplus \kappa\left(\mathbb{F}_2^m\right)\right\} ,\end{aligned}$$

so a TKlog maps two vector spaces of dimension m spanning $\mathrm{GF}(2^{2m})$ to two affine spaces of dimension m spanning $\mathrm{GF}(2^{2m})$.

As π is a TKlog instance, Theorem 1 implies that it verifies the following set equalities

$$\begin{cases}\pi\left(\mathrm{GF}(2^4)\right) & = \kappa(\mathbb{F}_2^4) \\ \pi\left(\alpha^i \odot \mathrm{GF}(2^4)^*\right) & = \kappa(16-i) \oplus \mathrm{GF}(2^m)^* \ ,\ \forall i \neq 0 \ ,\end{cases}$$

and applying Corollary 1 yields $\pi(\alpha^{16} \odot \mathrm{GF}(2^4)) = \kappa(0) \oplus \mathrm{GF}(2^4)$. These equalities are summarized in Figure 2 where relationships between complete affine spaces are represented by dashed and thick arrows while those linking sets of size $2^m - 1$ are represented by plain thin ones.

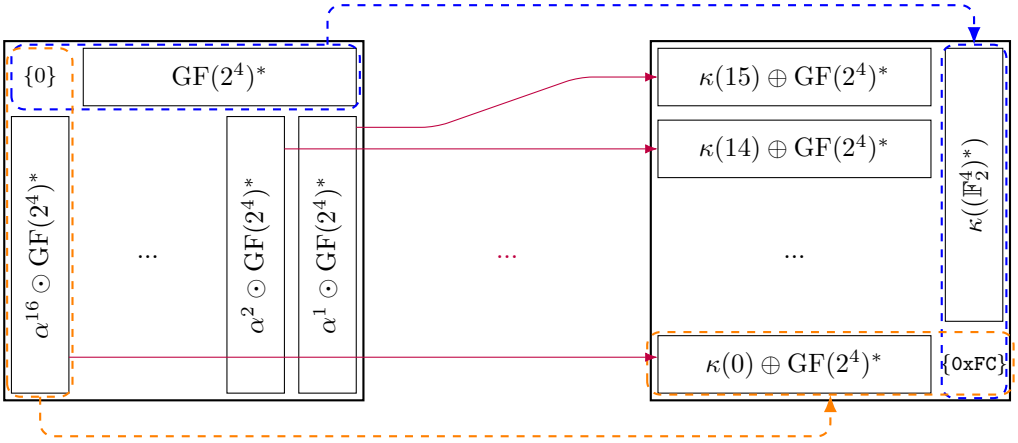


Figure 2: *The partition of $\mathrm{GF}(2^8)$ into multiplicative cosets of $\mathrm{GF}(2^4)^*$ (left), additive cosets of $\mathrm{GF}(2^4)^*$ (right), and the action of π on them (arrows).*

3.2.2 The Simplicity of the TKlog Properties

The partitions dealt with in Theorem 1 have simple algebraic descriptions. Indeed, $x, y \in \mathrm{GF}(2^{2m})^*$ are in the same additive coset of $\mathrm{GF}(2^m)^*$ if and only if $\mathrm{Tr}_m(x) = \mathrm{Tr}_m(y)$. Then, we remark that

$$\left(\alpha^{i+(2^m+1)j}\right)^{2^m-1} = \alpha^{(2^m-1)i+(2^{2m}-1)j} = \alpha^{(2^m-1)i} \ ,$$

so that $\left(\alpha^{i+(2^m+1)j}\right)^{2^m-1} = \left(\alpha^{i'+(2^m+1)j'}\right)^{2^m-1}$ if and only if $i = i'$. Thus, $x, y \in \mathrm{GF}(2^{2m})^*$ are in the same multiplicative coset of $\mathrm{GF}(2^m)^*$ if and only if $x^{2^m-1} = y^{2^m-1}$. Furthermore, $x \in \mathrm{GF}(2^m)^*$ if and only if $x^{2^m-1} = 1$. We deduce the following corollary of Theorem 1.

Corollary 2. *Let $\mathscr{T}_{\kappa,s} : \mathrm{GF}(2^{2m}) \to \mathrm{GF}(2^{2m})$ be a valid TKlog instance. Then we always have that*

$$x^{2^m-1} = y^{2^m-1} \neq 1 \ \Leftrightarrow\ \mathrm{Tr}_m\left(\mathscr{T}_{\kappa,s}(x)\right) = \mathrm{Tr}_m\left(\mathscr{T}_{\kappa,s}(y)\right) \neq \mathrm{Tr}_m\left(\kappa(0)\right) \ .$$

As a consequence, for any constant $c \in \mathrm{GF}(2^{2m})\backslash\{0,1\}$, we have the following implication involving only linear equations:

$$x^{2^m} \oplus cx = y^{2^m} \oplus cy = 0 \quad \Longrightarrow \quad (\mathscr{T}_{\kappa,s}(x))^{2^m} \oplus \mathscr{T}_{\kappa,s}(x) \;=\; (\mathscr{T}_{\kappa,s}(y))^{2^m} \oplus \mathscr{T}_{\kappa,s}(y)\ .$$

The interaction of a TKlog with these two partitions goes beyond mapping one to the other. Indeed, consider a more general structure corresponding to permutations P such that:

$$P : \begin{cases} 0 & \mapsto \kappa\left(t_0(0)\right) \\ \alpha^{(2^m+1)j} & \mapsto \kappa\left(t_0(2^m-j)\right) \\ \alpha^{i+(2^m+1)j} & \mapsto \kappa\left(t_1(2^m-i)\right) \oplus \left(\alpha^{2^m+1}\right)^{s_i(j)} \text{ for } i>0\ , \end{cases} \tag{2}$$

where t_0 and t_1 are permutations of $\mathbb{F}_2^m$, and where s_i is a permutation of $\mathbb{Z}/(2^m-1)\mathbb{Z}$ for all $i \in \mathbb{Z}/(2^m+1)\mathbb{Z}$. Any such P is a permutation with the exact partition-preserving property described in Theorem 1 but its contributions on $\mathrm{GF}(2^m)$ and on $\kappa(\mathbb{F}_2^m)$ each depend on both i and j, even when we restrict ourselves to $i > 0$. It is not the case for TKlogs; these permutations are far simpler.

Lemma 1 (Separation Property)**.** *Let $\mathscr{T}_{\kappa,s} : \mathrm{GF}(2^{2m}) \to \mathrm{GF}(2^{2m})$ be a valid TKlog instance. Then, for any i, j such that $0 < i \leq 2^m$ and $0 \leq j < 2^m - 1$, it holds that*

$$\mathscr{T}_{\kappa,s}(\alpha^{i+(2^m+1)j}) \;=\; \underbrace{\kappa(2^m-i)}_{\in\kappa(\mathbb{F}_2^m)} \oplus \underbrace{(\alpha^{2^m+1})^{s(j)}}_{\in\mathrm{GF}(2^m)^*}\ ,$$

so that the contribution of i is restricted to $\kappa(\mathbb{F}_2^m)$ and that of j is restricted to $\mathrm{GF}(2^m)$.

In other words, a TKlog interacts with each multiplicative coset other than $\mathrm{GF}(2^m)^*$ in the exact same way, even though this property is in no way implied by Theorem 1.

We could not find any attack leveraging these surprising properties of π. However, we did find that these partitions interact in a non-trivial way with the linear layer of Streebog. We discuss the consequences of the presence of this structure in π in Section 5.

4 The Missing Link

The TKlog structure is the missing link between the two previous decompositions of π. Its relationship with the logarithm-based decompositions of [PU16] is natural since both consist in a variant of the discrete logarithm followed by some arithmetic. The fact that π has a TU-decomposition remains a priori surprising but, in Section 4.1, we show that it is always the case for TKlogs. We also list some of the consequences of this property in Section 4.2.

4.1 A TU-Decomposition Always Exists

TKlogs can always be expressed in a fashion very similar to the first decomposition of Biryukov et al. [BPU16a]. In order to establish it, we first derive the following lemma.

Lemma 2. *There exists a function $\gamma : (\mathbb{Z}/(2^m+1)\mathbb{Z})^* \to \mathbb{Z}/(2^m-1)\mathbb{Z}$ such that*

$$\left\{x \in \mathrm{GF}(2^{2m}), \mathrm{Tr}(x) = 1\right\} \;=\; \left\{\alpha^{i+(2^m+1)\gamma(i)},\ i \in \mathbb{Z}/(2^m+1)\mathbb{Z}^*\right\}\ .$$

Proof. Let $(a,b) = \mathsf{Split}_\beta(x)$. If $x \notin \mathrm{GF}(2^m)$, then $a \neq 0$ and we can write $x = a(\beta \oplus c)$ where $c = b/a$ is an element of $\mathrm{GF}(2^m)$, so that $\mathrm{Tr}(\beta \oplus c) = \mathrm{Tr}(\beta) = 1$. As such a decomposition exists for all $x \notin \mathrm{GF}(2^m)$, we deduce that the set

$$\left\{\log_\alpha^{\mathrm{FLY}}(c \oplus \beta), c \in \mathrm{GF}(2^m)\right\}$$

must contain a representative of each equivalence class modulo 2^m+1 as the contrary would imply that some elements $\alpha^{i+(2^m+1)j}$ with $i \neq 0$ could not be written $a(\beta \oplus c)$. As a consequence, $\left\{ \log_\alpha^{\mathrm{FLY}}(c \oplus \beta) \mod (2^m+1), c \in \mathrm{GF}(2^m) \right\} = \{1, ..., 2^m\}$ and there must therefore exist a function γ as described above such that

$$\left\{ \log_\alpha^{\mathrm{FLY}}(c \oplus \beta), c \in \mathrm{GF}(2^m) \right\} = \left\{ 1 + (2^m+1)\gamma(1), ..., 2^m + (2^m+1)\gamma(2^m) \right\} .$$

The lemma follows. □

Theorem 2 (TU-Decomposition of the TKlog)**.** *The permutation $\mathscr{T}_{\kappa,s}$ of $\mathrm{GF}(2^{2m})$ has a TU-decomposition involving three m-bit permutations τ, ν and σ, and an m-bit function ϕ. It is given in Algorithm 2.*

The specification of the subcomponents is given in the proof of this theorem.

Algorithm 2 The TU-decomposition of TKlog.

function $\mathscr{T}_{\kappa,\alpha,s}(x \in \mathrm{GF}(2^8))$
  ▷ Input linear layer
  $a, b \leftarrow \mathsf{Split}_\beta(x)$
  ▷ Evaluation of $\ell = T_a(b)$
  if $a = 0$ **then**
    $\ell \leftarrow \tau(b)$
  else
    $\ell \leftarrow \nu(b/a)$
  end if
  ▷ Evaluation of $h = U_{T_a(b)}(a) = U_\ell(a)$
  $h \leftarrow \sigma\big(\phi(\ell) \odot a\big)$
  ▷ Output linear layer
  return $\kappa(\ell) \oplus h$
end function

Proof. Let $(a, b) = \mathsf{Split}_\beta(x)$. Intuitively, a will correspond to the right-side branch of the TU-decomposition and b/a to the left-side one. In particular, the case $a = 0$ is a special case. This proof is direct in the sense that it "simply" consists in defining the subcomponents τ, σ, ν and ϕ correctly and then checking that they work.

a = 0. If $a = 0$ then $x \in \mathrm{GF}(2^m)$, so that the definition of the TKlog yields $\mathscr{T}_{\kappa,s}(x) = \kappa(\tau(b))$ where

$$\tau : \begin{cases} \mathrm{GF}(2^m) & \to \mathbb{F}_2^m \\ 0 & \mapsto 0 , \\ b & \mapsto 2^m - \log_{(\alpha^{2^m+1})}^{\mathrm{FLY}}(b) \text{ if } b \neq 0 . \end{cases}$$

Indeed, let $x = b = \alpha^{(2^m+1)j}$ for $0 < j \leq 2^m - 1$. For $b \neq 0, 1$, we have that $\log_{(\alpha^{2^m+1})}^{\mathrm{FLY}}(b) = \log_\alpha(b)/(2^m+1) = j$, so that $\kappa(\tau(b))$ is indeed equal to $\mathscr{T}_{\kappa,s}(x)$. For $b = 0$, we also immediately have that $\kappa(\tau(b)) = \mathscr{T}_{\kappa,s}(x)$. For $b = 1$, $\log_\alpha^{\mathrm{FLY}}(b) = 2^{2m} - 1 = (2^m - 1)(2^m + 1)$, so that $j = 2^m - 1$ which is indeed equal to $\log_{(\alpha^{2^m+1})}^{\mathrm{FLY}}(b)$.

a ≠ 0. Otherwise, since $a \neq 0$ we can write $x = a(\beta \oplus b/a)$ and, as 0 and 1 are in $\mathrm{GF}(2^m)$, we have that $x \neq 0, 1$ so that $\log_\alpha^{\mathrm{FLY}}(x) = \log_\alpha(x)$. In order to evaluate $\mathscr{T}_{\kappa,s}(x)$, we need to find i and j such that $a(\beta \oplus b/a) = \alpha^{i+(2^m+1)j}$. It holds that $\mathrm{Tr}_m(a) = 0$ and $\mathrm{Tr}_m(\beta \oplus b/a) = 1$, so we can apply Lemma 2 to write

$$\begin{cases} \beta \oplus b/a &= \alpha^{i+(2^m+1)\gamma(i)} \\ a &= \alpha^{(2^m+1)(j-\gamma(i))} . \end{cases} \tag{3}$$

We define the permutations ν, σ and the function ϕ as follows.

- The permutation ν captures the way the logarithm and the arithmetic operation $x \mapsto 2^m - x$ operate on b/a to return the correct input for κ:

$$\nu : \begin{cases} \mathrm{GF}(2^m) & \to \mathbb{F}_2^4 \\ c & \mapsto 2^m - \left(\log_\alpha(\beta \oplus c) \mod (2^m+1)\right) , \end{cases}$$

  so that $\nu(b/a) = 2^m - i$.

- The permutation σ corresponds to the permutation s applied on j:

$$\sigma : \begin{cases} \mathrm{GF}(2^m) & \to \mathrm{GF}(2^m) \\ 0 & \mapsto 0 \\ c & \mapsto \left(\alpha^{2^m+1}\right)^{s\left(\log_{(\alpha^{2^m+1})}(c)\right)} \quad \text{when} \quad c \neq 0 . \end{cases}$$

- The function ϕ corresponds to the function γ introduced because of Lemma 2. It is defined by

$$\phi : \begin{cases} \mathbb{F}_2^m & \to \mathrm{GF}(2^m) \\ i & \mapsto \alpha^{(2^m+1)\gamma(2^m-i)} \end{cases} ,$$

  where $i \in \mathbb{F}_2^m$ is interpreted as an element of $\mathbb{Z}/(2^m+1)\mathbb{Z}$. Note that $\phi(i) \neq 0$ for all i. Using that $\nu(b/a) = 2^m - i$, we have

$$(\phi \circ \nu)(b/a) \;=\; \phi(2^m - i) \;=\; \alpha^{(2^m+1)\gamma(i)}$$

  and, using Equation 3 as well, we obtain that

$$(\phi \circ \nu)(b/a) \odot a \;=\; \alpha^{(2^m+1)\gamma(i)} \odot \alpha^{(2^m+1)(j-\gamma(i))} \;=\; \alpha^{(2^m+1)j} .$$

Combining this result with the definition of σ, we obtain that

$$\sigma\big((\phi \circ \nu)(b/a) \odot a\big) \;=\; \sigma\left(\alpha^{(2^m+1)j}\right) \;=\; \left(\alpha^{2^m+1}\right)^{s(j)} .$$

We can then write for any $a \neq 0$ that

$$\mathscr{T}_{\kappa,s}(a\beta \oplus b) \;=\; \kappa\big(\underbrace{\nu(b/a)}_{2^m-i}\big) \oplus \underbrace{\sigma\big((\phi \circ \nu)(b/a) \odot a\big)}_{(\alpha^{2^m+1})^{s(j)}} .$$

When $a = 0$, the term on the right of the XOR cancels out because $\sigma(0) = 0$. Therefore, when $a = 0$, we can write:

$$\mathscr{T}_{\kappa,s}(a\beta \oplus b) \;=\; \underbrace{\kappa\big(\tau(b)\big)}_{\mathscr{T}_{\kappa,s}(b)} \oplus \underbrace{\sigma\big((\phi \circ \nu)(b/a) \odot a\big)}_{0} ,$$

which is the same expression as when $a \neq 0$ except that the input of κ is changed from $\nu(b/a)$ to $\tau(b)$. As a consequence, it is possible to evaluate this function in two stages.

1. Let $(a, b) = \mathsf{Split}_\beta(x)$. If $a = 0$, let $\ell = \tau(b)$; otherwise let $\ell = \nu(b/a)$.
2. Return $\kappa(\ell) \oplus \sigma\big(\phi(\ell) \odot a\big)$.

This process is exactly the one described in Algorithm 2, an observation which concludes the proof. □

The existence of the TU-decomposition of Biryukov et al. is obviously a direct consequence of this theorem. The relationship between this decomposition and the original TKlog structure further allows us to better understand some of the patterns existing in this decomposition.

First, the component Biryukov et al. called ν_0 plays the role of τ in Theorem 2. Since τ is essentially a logarithm in base $\alpha^{-(2^m+1)}$, it is not surprising that ν_0 was noted in [PU16] to be affine-equivalent to a logarithm.

Second, the linear layers α and ω of the original decomposition are basically splitting $\mathrm{GF}(2^{2m})$ into $\mathrm{GF}(2^m)^2$ and then putting it back together, hence the relation presented in Equation 1. Furthermore, the output layer ω is more complex than the input layer α since it also evaluates the linear part of κ.

4.2 Properties Explained by the TU-Decomposition

Efficient Hardware Implementation. The knowledge of the decomposition from Theorem 2 allowed Biryukov et al. to significantly improve the implementation of π: they made a circuit implementing such a permutation with an area and delay divided by 2.5 and 8 respectively [BPU16b, Table 8]. We can expect similar gains for all 8-bit TKlogs.

Visual Artefact in the LAT. Biryukov et al. used a particular visual artefact in the LAT of π to perform the first step of their TU-decomposition, namely the presence of columns with a lower number of different coefficients than the others. They showed that this pattern was a direct consequence of the structure they identified in this S-Box, namely the one imposed by Theorem 2. As a consequence, we can generalize their result to all TKlog instances.

Lemma 3. *Consider a TKlog instance $\mathscr{T}_{\kappa,\alpha,s}$, let $\mathsf{Split}_\beta, \tau, \sigma, \nu$ and ϕ be the subfunctions used to implement its TU-decomposition as presented in Algorithm 2, and let $\phi_\kappa : \mathrm{GF}(2^{2m}) \to \mathrm{GF}(2^m) \times \mathrm{GF}(2^m)$ be the affine bijection such that $\phi_\kappa(\kappa(x) \oplus v) = (x, v)$.*

We further let F be the permutation of $\mathrm{GF}(2^m)^2$ defined by $F = \phi_\kappa \circ \mathscr{T}_{\kappa,s} \circ \mathsf{Split}_\beta^{-1}$, so that it consists in the TU-decomposition of $\mathscr{T}_{\kappa,s}$ without the input and output linear layers. Then, for $(a_L, a_R) \neq (0,0)$ and $b_L \neq 0$, we have

$$\mathcal{W}_F\left[(a_L, a_R),\ (b_L, 0)\right] \ = \ \big(\mathcal{W}_\tau[a_L, b_L] \ \pm \ 2^m\big)[a_L \neq 0] \ ,$$

where $[a_L \neq 0] = 0$ if $a_L = 0$ and 1 otherwise.

The proof of this lemma is essentially the same as the one Biryukov et al. gave in the full version of their paper [BPU16b]. For the sake of completeness, we also give it in Appendix 11.

Since τ is essentially a discrete logarithm, its non-linearity is always high, so that the pattern is always visible. Indeed, because of this high non-linearity, $|\mathcal{W}_\tau[a_L, b_L]|$ can only take values much lower than 2^m, which implies that the low contrast will always be present.

In [BPU16a], the authors of the original decomposition remarked that they had been lucky since the conjunction of circumstances that lead to the success of their decomposition seemed unlikely. In their own words (page 19 of [BPU16b]):

> [the visual pattern] is caused by the conjunction of three elements:
>
> - the use of a multiplexer,
> - the use of finite field inversion, and
> - the fact that ν_0 has good non-linearity.

> Ironically, the only "unsurprising" sub-component of π, namely the inverse function, is one of the reasons why we were able to reverse-engineer this S-Box in the first place. Had [the inversion] been replaced by a different (and possibly weaker!) S-Box, there would not have been any of the lines in the LAT which got our reverse-engineering started.

We can now see that, rather than luck, all of these events are direct consequences of the TKlog structure of π.

Incidentally, it provides an excellent method for spotting such structure should someone else decide to use one—and try to keep this fact hidden. In [PU16], Perrin and Udovenko suggested to plot the variance of the absolute value of the coefficients in each row/column of the LAT. They noticed that, in the case of π, there was a sharp drop for some columns. Because of Lemma 3, we can see that this pattern is an inherent property of the TKlog which will therefore always betray the presence of such a structure. Furthermore, should the TKlog instance be obfuscated by composing it with affine layers, this pattern would remain and in fact provide some information about the linear part of said affine layers.

Furthermore, since a TKlog always has a TU-decomposition, it will always have a vector space of dimension $n = 2m$ inside the set of the coordinates of the zeroes of its LAT [CP19]. This other pattern can also be detected and thus identicate a TKlog instance to be such.

CCZ-Equivalence. Because of Theorem 2, we know that a TKlog is always affine-equivalent to a permutation $P : (\mathrm{GF}(2^m))^2 \to (\mathbb{F}_2^m)^2$ such that

$$P(b,a) = \left(T_a(b), U_{T_a(b)}(a)\right) \text{ where } \begin{cases} T_a(b) & = \tau(b) \times [a=0] \oplus \nu(b/a) \times [a \neq 0] \ , \\ U_k(a) & = \sigma\left(a/\phi(k)\right) \ , \end{cases}$$

where $[a = 0]$ is a Boolean function mapping 0 to 1 and $a \neq 0$ to 0 and where $[a \neq 0] = 1 \oplus [a = 0]$.

As explained in [CP19], the existence of such a decomposition where T_r is a permutation for all $r \in \mathbb{F}_2^m$ is equivalent to the possibility of a so-called *m-twist*, an operation which preserves the CCZ-equivalence class but a priori does not preserve the AE-equivalence class. As a functional inversion also preserves the CCZ-Equivalence class, we deduce the following corollary of Theorem 2.

Corollary 3. *Let T and U be as defined above. Both the corresponding TKlog instance and its inverse are CCZ-equivalent to the function $F : \mathbb{F}_2^m \times \mathrm{GF}(2^m) \to \mathbb{F}_2^m \times \mathrm{GF}(2^m)$ such that*

$$F(b,a) = \left(T_a^{-1}(b), U_b(a)\right) \textit{ where } \begin{cases} T_a^{-1}(b) & = \tau^{-1}(b) \times [a=0] \oplus \left(\nu^{-1}(b) \odot a\right) \times [a \neq 0] \ , \\ U_b(a) & = \sigma\left(a/\phi(b)\right) \ , \end{cases}$$

Proof. By definition (see [CP19]), the m-twist maps a function $P : (r, \ell) \mapsto \left(T_r(\ell), U_{T_r(\ell)}(r)\right)$ to $F : (r, \ell) \mapsto \left(T_r^{-1}(\ell), U_\ell(r)\right)$. The corollary follows directly. □

In the case of π, we generated the function F_π which is CCZ-equivalent to it as specified in Corollary 3. Its lookup-table is provided in the Supplementary Material for the sake of completeness (see Table 3). It of course has the same differential and extended Walsh spectra as π and, again like π, all of its coordinates have degree 7. However, it is not a permutation: 15 elements in its image have 3 preimages, 75 have 2 and the 61 remainding ones have 1.

5 New Information on the Russian Primitives

In this section, we discuss the consequences of the fact that π is (up to a translation) a TKlog instance for the primitives using it. First, we argue that the presence of a TKlog structure must be a deliberate choice from the designers (Section 5.1). We then introduce a new representation of the binary matrix used in Streebog in Section 5.2 which we use to highlight some interactions between the partitions preserved by π and this linear component. Finally, we discuss our findings and their consequences in Section 5.3.

5.1 The Likely Design Process of π

In light of our results, we can deduce some information about the design process of π. First, we establish that the number of TKlog instances is extremely small, meaning that the choice of this structure must have been deliberate. Then, using some experimental results, we obtain a design process which yields results extremely similar to π.

Density of the TKlog set. Apart from the high level algorithm, a TKlog operating on 8 bits is fully defined by three components: a primitive polynomial p of degree 8 (there are 16 possible choices), an affine function $\kappa : x \mapsto \Lambda(x) \oplus \kappa(0)$ where the 8×4 binary matrix Λ is such that $\Lambda(\mathbb{F}_2^4)$ and $\mathrm{GF}(2^4)$ span $\mathrm{GF}(2^8)$, and a permutation s of $\mathbb{Z}/15\mathbb{Z}$. The matrix Λ must be such that its first column Λ_0 is not in $\mathrm{GF}(2^4)$, the second one is not in $\mathrm{GF}(2^4) \cup (\Lambda_0 \oplus \mathrm{GF}(2^4))$, etc. so that there are $(2^8 - 2^4)(2^8 - 2^5)(2^8 - 2^6)(2^8 - 2^7) \approx 2^{30.3}$ choices for this matrix. As a consequence, there are about

$$\underbrace{16}_{p} \times \underbrace{2^{30.3}}_{\Lambda} \times \underbrace{2^8}_{\kappa(0)} \times \underbrace{15!}_{s} \;\approx\; 2^{82.6}$$

distinct TKlog instances on 8 bits.

This number is very small. To put it into perspective, there is a total of $2^8! \approx 2^{1684}$ permutations of $\mathbb{F}_2^8$ out of which $2^8 \times \prod_{i=0}^{7}(2^8 - 2^i) \approx 2^{70.2}$ are affine. The number of TKlog instances is thus about 4000 times larger than the number of affine permutations.

Our point with these estimates is to give an intuition of how *small* the number of TKlogs is. A random permutation generator returning an affine permutation would be assumed to deliberately generate such object. In much the same way, the generation process that led the designers of Streebog to choose π can be assumed to have deliberately returned a TKlog instance.

Claim. *Given how small the number of TKlog is, we are confident that the designers of π deliberately chose to use this structure.*

Experimental Results. How good are the differential and linear properties of TKlog instances compared to those expected from a random permutation? To answer this question, we build upon the analysis of the S-Box of Skipjack in [BP15] to introduce the following concepts.

Definition 2 (Anomaly of an S-Box)**.** Let $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$ be a permutation, $u(F)$ be its differential uniformity, and $N_k(F)$ be the number of occurrences of k in its DDT. The *differential anomaly of F* is equal to

$$\mathsf{A}_F^{\mathrm{d}} = -\log_2\left(\Pr\left[u(G) \le u(F) \text{ and } N_{u(F)}(G) \le N_{u(F)}(F)\right]\right),$$

where the probability is taken over all permutations G. If $\ell(F)$ is the linearity of F and $N'_k(F)$ is the sum of the number of occurrences of k and $-k$ in the LAT of F, then the *linear anomaly of F* is equal to

$$\mathsf{A}_F^{\ell} = -\log_2\left(\Pr\left[\ell(G) \le \ell(G) \text{ and } N'_{\ell(F)}(G) \le N'_{\ell(F)}(F)\right]\right),$$

where the probability is taken over all permutations G.

An S-Box with a differential anomaly close to 0 has differential properties close to those of a random S-Box or worse. The differential anomaly behaves as we would expect: when the differential uniformity decreases under its expected value, the anomaly increases. As it contains more information than the differential uniformity, it allows a comparison of S-Boxes for which this quantity is the same. From a cryptographic standpoint, the higher the anomaly the better as it means that the S-Box will provide a better security against differential attacks. The same can be said for the linear anomaly.

In [BP15], Biryukov and Perrin provided formulas for computing the differential and linear anomalies based on the statistical distribution of the DDT and LAT coefficients presented in [DR07]. They also showed that the linear anomaly of the S-Box of Skipjack was equal to 55.4 so that this component could not have been generated randomly.

To try and gather more information about the design process of π, we generated 10^6 random 8-bit TKlog instances. We plotted the differential and linear anomalies of each of them in a two dimensional graph given in Figure 3. Each instance corresponds to a light gray point; darker points are obtained when multiple instances have the same differential and linear anomalies. We also put the anomalies of π, $\log_\alpha^{\mathrm{FLY}}$ and $\log_\alpha^{\mathrm{HN}}$ in the same graph for comparison.

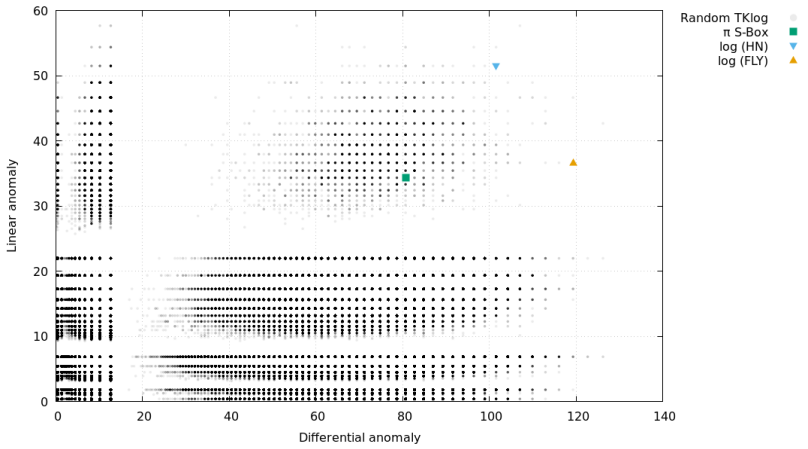


Figure 3: *The differential and linear anomalies of random 8-bit TKlog instances, π, $\log_\alpha^{\mathrm{FLY}}$ and $\log_\alpha^{\mathrm{HN}}$.*

As we can see, the differential and linear anomalies of π are somewhat good but not exceptional compared to those of a random TKlog. More precisely, an 8-bit TKlog instance has both a differential and linear anomaly at least as high as that of π with probability about $2^{-10.6}$ and it is not hard to obtain much better instances. They are also lower than those of both $\log_\alpha^{\mathrm{FLY}}$ and $\log_\alpha^{\mathrm{HN}}$.

However, none of our random instances have a better differential uniformity or linearity than π (including $\log_\alpha^{\mathrm{FLY}}$ and $\log_\alpha^{\mathrm{HN}}$). Furthermore, π is in the area of Figure 3 containing most instances with the same differential uniformity and linearity. Thus, its anomalies are on par with those of a random TKlog instance with the same differential uniformity and linearity.

Design Process Outline. In light of the experimental results above, we can see that the following design process would yield a result very similar to π.

1. Figure out that the best possible differential uniformity for an 8-bit TKlog instance is 8 and the best linearity is 56, for example via extensive computer simulations.

2. Pick a TKlog instance at random among those with said differental uniformity and linearity—without taking the anomaly into account.

This strategy is natural as long as there is a reason to impose the use of a TKlog—though we cannot think of one. Since both the differential and linear anomalies of π are inferior to those of $\log_\alpha^{\mathrm{FLY}}$ and of $\log_\alpha^{\mathrm{HN}}$, the purpose of the use of a TKlog in this case could not be an improvement of the cryptographic properties of the discrete logarithm. More importantly, the very strong algebraic properties of such components which we described in Section 3.2 would a priori invite caution; even more so in the case of Streebog. Indeed, as we explain below, its linear layer interacts non-trivially with the corresponding partitions.

5.2 On the Linear Layer of Streebog

The binary matrix corresponding to the L operation of Streebog is given in Figure 4 where a black pixel corresponds to 1 and a white one to 0.

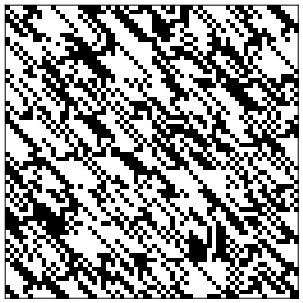


Figure 4: *The* 64×64 *matrix* L *used in Streebog.*

As we can see, it has a strong structure. In [KK13], Kazymyrov and Kazymyrova showed that it could be written as the composition of:

- a layer of 8-bit linear permutations ℓ which simply inverts the order of the bits in each byte,
- the multiplication by an 8×8 MDS matrix of $\mathrm{GF}(2^8) = \mathbb{F}_2[X]/P_{\mathrm{KK}}(X)$ where $P_{\mathrm{KK}}(X) = X^8 \oplus X^6 \oplus X^5 \oplus X^4 \oplus 1$ is a primitive polynomial of degree 8, and
- the inverse of the layer ℓ.

We used a very direct approach to try and simplify this structure: by setting each byte in a row to 1 one after the other, multiplying it by L, and then writing its bytes as elements of $\mathrm{GF}(2^8) = \mathbb{F}_2[X]/p_{\min}(X)$, we generated the matrix L^{F} such that

$$\mathsf{L}^{\mathsf{F}} = \begin{bmatrix} 83 & 47 & 8b & 07 & b2 & 46 & 87 & 64 \\ 46 & b6 & 0f & 01 & 1a & 83 & 98 & 8e \\ ac & cc & 9c & a9 & 32 & 8a & 89 & 50 \\ 03 & 21 & 65 & 8c & ba & 93 & c1 & 38 \\ 5b & 06 & 8c & 65 & 18 & 10 & a8 & 9e \\ f9 & 7d & 86 & d9 & 8a & 32 & 77 & 28 \\ a4 & 8b & 47 & 4f & 9e & f5 & dc & 18 \\ 64 & 1c & 31 & 4b & 2b & 8e & e0 & 83 \end{bmatrix}.$$

The polynomial used by Kazymrov and Kazymrova is the reciprocate of $p_{\min}$, i.e. $P_{\mathrm{KK}}(1/X) = p_{\min}(X)/X^8$. In hindsight, it was obvious that the reversal of the bit order at the byte level in their expression of the linear layer could be removed by considering this polynomial instead.

In the end, if we let A be the 8×8 matrix of elements of $\mathrm{GF}(2^8)$ corresponding to the internal state of Streebog and let P denote the transposition of A (as is done in the specification of Streebog), then applying the whole linear part of the round function of Streebog can be written

$$(\mathsf{L} \circ \mathsf{P})(A) = A^T \times \mathsf{L}^{\mathsf{F}} \ ,$$

where "$\times$" denotes the usual matrix multiplication.

Additive to Multiplicative Cosets. The subfield $\mathrm{GF}(2^4)^*$ has a particular interaction with L. Indeed, applying the matrix multiplication of Streebog to the vector $x^i = [0, ..., 0, x, 0, ..., 0]$ of $\mathrm{GF}(2^8)^8$ such that $x^i_i = x$ and $x^i_k = 0$ if $k \neq i$ is equivalent to computing

$$v = x^i \times \mathsf{L}^{\mathsf{F}} = \left[\mathsf{L}^{\mathsf{F}}_{i,0} \odot x, ..., \mathsf{L}^{\mathsf{F}}_{i,7} \odot x\right] \ ,$$

so that if $x \in \mathrm{GF}(2^m)^*$ then $v_j \in \mathsf{L}^{\mathsf{F}}_{i,j} \odot \mathrm{GF}(2^m)^*$, i.e. it maps the subfield to its multiplicative cosets. However, it is unclear what happens when multiple cells of the input vector are active.

Kuznyechik. The linear layer of Kuznyechik is specified as an LFSR made of 16 cells, each of which is an element of $\mathrm{GF}(2^8)$, which is clocked 16 times. It can also be represented as a multiplication by a 16×16 matrix. However the representation of the field elements uses a different polynomial, namely $p_{\mathrm{kuz}}(X) = X^8 \oplus X^7 \oplus X^6 \oplus X \oplus 1$. While $p_{\min}(X) = X^8 \oplus X^4 \oplus X^3 \oplus X^2 \oplus 1$ is the first primitive polynomial of degree 8 in lexicographic order, p_{kuz} is the last such polynomial of weight 5 (see Table C of [LN97]).

Unlike the matrix multiplication in Streebog, the one in Kuznyechik cannot be written as a matrix multiplication in $\mathbb{F}_2[X]/p_{\min}(X)$, so that the coset propagation described above for the hash function does not seem applicable to the block cipher.

5.3 Discussion

The consequences of the partition-preserving properties of π and its non-trivial interaction with the linear layer are hard to assess.

In the literature, we can find other S-Boxes mapping cosets to cosets. For example, monomials map multiplicative cosets of the subfield to multiplicative cosets of the subfield: if $F : x \mapsto x^d$ is a permutation of $\mathrm{GF}(2^{2m})$, then

$$F\left(\alpha^i \odot \mathrm{GF}(2^m)\right) \ = \ \alpha^{d \times i} \odot \mathrm{GF}(2^m) \ .$$

If we remove their affine components, the S-Boxes of the AES [AES01] and Misty1 [Mat97] (among many others) exhibit this behaviour. Yet, despite their appearance in some very prominent targets, multiplicative cosets have never been used in symmetric cryptanalysis. Note that algorithm designers always compose the inverse with unrelated affine layers so as to break its algebraic structure. This conservative decision is likely to prevent the use of multiplicative cosets to attack these ciphers in practice.

It is not the case for additive cosets. In fact, the authors of [BBF16] purposefully built an S-Box mapping additive cosets to additive cosets with the explicit purpose of using this pattern as a backdoor. They show that such a partition can be preserved if the linear layer is chosen carefully and can thus hold for an arbitrary number of rounds. The reason why Bannier et al. considered additive cosets is the following observation.

Remark 1. If $F_k : x \mapsto x \oplus k$ is a key addition in $\mathbb{F}_2^n$ and V is a vector subspace of $\mathbb{F}_2^n$, then

$$F_k\big(c \oplus V\big) = (k \oplus c) \oplus V \ ,$$

so that the partition of $\mathbb{F}_2^n$ into additive cosets of V is preserved under F_k *independently from* k.

In the cipher of Bannier et al., the key schedule can therefore be arbitrarily complex without hindering the backdoor. This property is not shared by the partition into multiplicative cosets.

The only case (other than the TKlog) we can think of where a partition into cosets is mapped to a different partition is that of (plain) discrete logarithms. Indeed, a Hakala-Nyberg type of logarithm operating on $\mathrm{GF}(2^{2m})$, which maps α^{2^m-1} to 0, always maps multiplicative cosets of the subfield to additive cosets of $\mathbb{Z}/(2^m-1)\mathbb{Z}$. In this case, the multiplication is in the finite field and the addition over the integers. Since these two operations are completely different, we deem it unlikely that a cryptanalysis is aided by this property.

In the end, when looking at the impact of cosets on symmetric primitives, we have one of the following situations:

1. the partition into cosets cannot be iterated since the input partition and output partition are over completely different structures (case of the logarithm);
2. although the S-Box and linear layer are defined over similar structures, a small function was added with the explicit purpose of breaking this similarity (case of the AES and the affine permutation used in its S-Box); or
3. the S-Box and the linear layer were chosen with aligned structures that preserve the same partition so as to purposefully introduce a backdoor in a block cipher (backdoored cipher of of [BBF16]).

Kuznyechik seems to be in the second situation. While the designers did not disclose their security analysis, it would make sense for them to choose the polynomial used to define the finite field in which the linear layer operates so as not to "align" it with the structure used to construct π.

However, Streebog falls in neither category. The input and output partitions are defined over the same structure (the finite field) so it is not in the first situation. The S-Box could have been composed with an affine layer breaking its relationship with $\mathrm{GF}(2^8)$ (like in the AES) or the linear layer could have been defined over a different finite field (like in Kuznyechik) but neither is the case so it does not fall in the second category either. Still, while the linear layer is defined over the same structure as the partitions preserved by the S-Box, these partitions are different and it is unclear how they may interact with the matrix multiplication. It is therefore not obvious that Streebog fits into the third category and the following question remains open.

Open Problem 1. *Is there a way to leverage the partition-preserving property of π to mount an attack against Streebog?*

6 Conclusion

We have extracted a new structure from π which we claim to be the one originally intended by its designers. Its generalization, the TKlog, is obtained by composing a discrete logarithm with a simple layer of arithmetic. The TKlog explains both previous decompositions of π, thus providing the missing link between these two results.

The knowledge of this decomposition allowed us to explain a very specific partition-preserving property of π. Surprisingly, we also found a new expression of the linear

layer of Streebog expressed in the same finite field as π. While we cannot leverage these properties to attack this hash function, we question the wisdom of this design choice. Indeed, when dealing with components defined over identical mathematical structures, academic designers break this alignment e.g. by composing their S-Boxes with unrelated affine permutations. We are of the opinion that it would have been more cautious to do the same in Streebog.

7 Acknowledgement

We thank the IACR ToSC reviewers for their careful reading and comments. We also thank Anne Canteaut for remarks that greatly improved the presentation of this paper. The work of the author was financed by a post-doc grant from the *Fondation Sciences Mathématiques de Paris* (FSMP.)

References

[AES01] Advanced Encryption Standard (AES). National Institute of Standards and Technology (NIST), FIPS PUB 197, U.S. Department of Commerce, November 2001.

[BBF16] Arnaud Bannier, Nicolas Bodin, and Eric Filiol. Partition-based trapdoor ciphers. Cryptology ePrint Archive, Report 2016/493, 2016. http://eprint.iacr.org/2016/493.

[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.

[Bih97] Eli Biham, editor. *Fast Software Encryption – FSE'97*, volume 1267 of *Lecture Notes in Computer Science*. Springer, Heidelberg, January 1997.

[BP15] Alex Biryukov and Léo Perrin. On reverse-engineering S-boxes with hidden design criteria or structure. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 116–140. Springer, Heidelberg, August 2015.

[BPU16a] Alex Biryukov, Léo Perrin, and Aleksei Udovenko. Reverse-engineering the S-box of streebog, kuznyechik and STRIBOBr1. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part I*, volume 9665 of *Lecture Notes in Computer Science*, pages 372–402. Springer, Heidelberg, May 2016.

[BPU16b] Alex Biryukov, Léo Perrin, and Aleksei Udovenko. Reverse-engineering the S-box of streebog, kuznyechik and STRIBOBr1. Cryptology ePrint Archive, Report 2016/071, 2016. http://eprint.iacr.org/2016/071.

[BS91a] Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystems. In Alfred J. Menezes and Scott A. Vanstone, editors, *Advances in Cryptology – CRYPTO'90*, volume 537 of *Lecture Notes in Computer Science*, pages 2–21. Springer, Heidelberg, August 1991.

[BS91b] Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystems. *Journal of Cryptology*, 4(1):3–72, 1991.

[CCZ98] Claude Carlet, Pascale Charpin, and Victor Zinoviev. Codes, bent functions and permutations suitable for DES-like cryptosystems. *Designs, Codes and Cryptography*, 15(2):125–156, 1998.

[CDL16] Anne Canteaut, Sébastien Duval, and Gaëtan Leurent. Construction of lightweight S-boxes using Feistel and MISTY structures. In Orr Dunkelman and Liam Keliher, editors, *SAC 2015: 22nd Annual International Workshop on Selected Areas in Cryptography*, volume 9566 of *Lecture Notes in Computer Science*, pages 373–393. Springer, Heidelberg, August 2016.

[CP19] Anne Canteaut and Léo Perrin. On CCZ-equivalence, extended-affine equivalence, and function twisting. *Finite Fields and Their Applications*, 56:209–246, 2019.

[DD13] V. Dolmatov and A. Degtyarev. Gost r 34.11-2012: Hash function. RFC 6986, RFC Editor, August 2013.

[Dev17] The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 7.5.1)*, 2017. `http://www.sagemath.org`.

[Dol16] V. Dolmatov. Gost r 34.12-2015: Block cipher "kuznyechik". RFC 7801, RFC Editor, March 2016.

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

[Fed12] Federal Agency on Technical Regulation and Metrology. Information technology – data security: Hash function. English version available at http://wwwold.tc26.ru/en/standard/gost/GOST_R_34_11-2012_eng.pdf, 2012.

[Fed15] Federal Agency on Technical Regulation and Metrology. Information technology – data security: Block ciphers. English version available at http://wwwold.tc26.ru/en/standard/gost/GOST_R_34_12_2015_ENG.pdf, 2015.

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

[Hel94] Tor Helleseth, editor. *Advances in Cryptology – EUROCRYPT'93*, volume 765 of *Lecture Notes in Computer Science*. Springer, Heidelberg, May 1994.

[HMS+76] M. Hellman, R. Merkle, R. Schroeppel, L. Washington, W. Diffie, S. Pohlig, , and P. Schweitzer. Results of an initial attempt to cryptanalyze the NBS Data Encryption Standard. Technical report, Stanford University, Information Systems Laboratory, 1976. Available at https://ee.stanford.edu/~hellman/resources/1976_sel_des_report.pdf.

[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*, pages 333–345, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[KK13] Oleksandr Kazymyrov and Valentyna Kazymyrova. Algebraic aspects of the russian hash standard GOST r 34.11-2012. Cryptology ePrint Archive, Report 2013/556, 2013. http://eprint.iacr.org/2013/556.

[LN97] Rudolf Lidl and Harald Niederreiter. *Finite fields*, volume 20 of *Encyclopedia of Mathematics and its Applications*. Cambridge university press, 1997.

[LW14] Yongqiang Li and Mingsheng Wang. Constructing S-boxes for lightweight cryptography with Feistel structure. In Lejla Batina and Matthew Robshaw, editors, *Cryptographic Hardware and Embedded Systems – CHES 2014*, volume 8731 of *Lecture Notes in Computer Science*, pages 127–146. Springer, Heidelberg, September 2014.

[Mat94] Mitsuru Matsui. Linear cryptanalysis method for DES cipher. In Helleseth [Hel94], pages 386–397.

[Mat97] Mitsuru Matsui. New block encryption algorithm MISTY. In Biham [Bih97], pages 54–68.

[Nyb94] Kaisa Nyberg. Differentially uniform mappings for cryptography. In Helleseth [Hel94], pages 55–64.

[Pat99] Kenneth G. Paterson. Imprimitive permutation groups and trapdoors in iterated block ciphers. In Lars R. Knudsen, editor, *Fast Software Encryption – FSE'99*, volume 1636 of *Lecture Notes in Computer Science*, pages 201–214. Springer, Heidelberg, March 1999.

[PU16] Léo Perrin and Aleksei Udovenko. Exponential s-boxes: a link between the s-boxes of BelT and Kuznyechik/Streebog. *IACR Transactions on Symmetric Cryptology*, 2016(2):99–124, 2016. http://tosc.iacr.org/index.php/ToSC/article/view/567.

[RP97] Vincent Rijmen and Bart Preneel. A family of trapdoor ciphers. In Biham [Bih97], pages 139–148.

[Saa14] Markku-Juhani O Saarinen. The STRIBOBr1 authenticated encryption algorithm. Candidate of the 1st round of the CAESAR competition, available online at https://competitions.cr.yp.to/round1/stribobr1.pdf, 2014.

[SB15] Markku-Juhani O. Saarinen and Billy Bob Brumley. Whirlbob, the whirlpool based variant of STRIBOB. In Sonja Buchegger and Mads Dam, editors, *Secure IT Systems, 20th Nordic Conference, NordSec*, volume 9417 of *Lecture Notes in Computer Science*, pages 106–122. Springer, 2015.

[Shi13] Vassilij Shishkin. Принципы синтеза перспективного алгоритма блочного шифрования с длиной блока 128 бит [design principles of the perspective block encryption algorithm with a block length of 128 bits]. Presentation at RusCrypto'13, available online at https://www.ruscrypto.ru/resource/archive/rc2013/files/03_shishkin.pdf (in Russian), March 2013.

[SSA+07] Taizo Shirai, Kyoji Shibutani, Toru Akishita, Shiho Moriai, and Tetsu Iwata. The 128-bit blockcipher CLEFIA (extended abstract). In Alex Biryukov, editor, *Fast Software Encryption – FSE 2007*, volume 4593 of *Lecture Notes in Computer Science*, pages 181–195. Springer, Heidelberg, March 2007.

[U.S98] U.S. Department Of Commerce/National Institute of Standards and Technology. Skipjack and KEA algorithms specifications, v2.0, 1998. http://csrc.nist.gov/groups/ST/toolkit/documents/skipjack/skipjack.pdf.

[WBDY98] Hongjun Wu, Feng Bao, Robert H. Deng, and Qin-Zhong Ye. Cryptanalysis of Rijmen-Preneel trapdoor ciphers. In Kazuo Ohta and Dingyi Pei, editors, *Advances in Cryptology – ASIACRYPT'98*, volume 1514 of *Lecture Notes in Computer Science*, pages 126–132. Springer, Heidelberg, October 1998.

8 Look-up Tables

The S-Box π was only specified via its look-up table. It is provided in Table 2. The function F_π is obtained from π via Corollary 3. Its look-up table is provided in Table 3.

Table 2: *The look-up table of π. For example $\pi(7A) = C6$.*

	.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

Table 3: *The look-up table of F_π, a function CCZ-equivalent to π.*

	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F
0.	8C	42	87	C8	9E	DF	13	54	B6	FB	3D	75	29	60	AA	E1
1.	BC	70	E8	9A	53	25	BF	C6	A9	D2	41	37	FD	84	1B	6E
2.	0C	D2	37	E8	6E	BF	53	84	C6	1B	FD	25	A9	70	9A	41
3.	2C	E3	59	BD	A2	44	F0	15	D7	3E	8A	66	78	9F	21	CB
4.	9C	8F	9D	11	B0	36	24	A7	F8	73	6B	E9	4A	C5	DE	52
5.	1C	98	B3	2F	F9	61	4D	DB	72	E7	C4	5E	80	1A	35	A6
6.	4C	1D	20	34	48	5B	6A	7E	83	99	A5	B2	CF	D1	E6	F7
7.	FC	0F	0D	01	00	06	04	07	08	03	0B	09	0A	05	0E	02
8.	CC	6A	CF	A5	1D	7E	D1	B2	20	48	E6	83	34	5B	F7	99
9.	5C	34	6A	5B	CF	F7	A5	99	1D	20	7E	48	D1	E6	B2	83
A.	7C	FB	75	87	E1	13	9E	60	54	AA	29	DF	B6	42	C8	3D
B.	3C	A3	D9	7D	32	94	E0	45	67	CE	BA	16	58	FF	81	2B
C.	EC	26	4B	62	85	A8	C7	ED	91	B4	D3	FA	1E	39	50	7F
D.	DC	C4	1A	DB	2F	E7	35	F9	4D	80	5E	98	61	A6	72	B3
E.	6C	BA	FF	45	7D	CE	81	32	E0	58	16	A3	94	2B	67	D9
F.	AC	52	A7	F8	DE	8F	73	24	36	6B	9D	C5	E9	B0	4A	11

9 Algorithms

An algorithm evaluating a TKlog is provided in Algorithm 3 and one evaluating its inverse, a TKexp, is given in Algorithm 4.

Algorithm 3 A TKlog permutation.

function $\mathscr{T}_{\kappa,s}(x \in \mathrm{GF}(2^{2m}))$
  $k \leftarrow \log_\alpha^{\mathrm{FLY}}(x)$
  if $k = 0$ **then** ▷ Case where $x = 0$
    return 0
  else
    $i \leftarrow k \mod (2^m + 1)$
    $j \leftarrow \lfloor k/(2^m + 1) \rfloor$ ▷ $x = \alpha^{i+(2^m+1)j}$
    if $i = 0$ **then** ▷ Case where $x \in \mathrm{GF}(2^m)$
      return $\kappa(2^m - j)$ ▷ As $i = 0$, $j \in \{1, ..., 2^m - 1\} = (\mathbb{F}_2^m)^*$
    else
      return $\kappa(2^m - i) \oplus \alpha^{(2^m+1)s(j)}$ ▷ $i \neq 0$ so $(2^m - i) \in \mathbb{F}_2^m$
    end if
  end if
end function

Algorithm 4 A TKexp permutation.

function $\mathscr{T}_{\kappa,s}^{-1}(x \in \mathrm{GF}(2^{2m}))$
  if $x = 0$ **then**
    return 0
  else
    $(k, v) \leftarrow \Phi_\kappa(x)$
    if $v = 0$ **then** ▷ Case where $x \in \kappa(\mathbb{F}_2^m)$
      return $\alpha^{(2^m+1)(2^m-k)}$
    else
      $j \leftarrow \log_\alpha^{\mathrm{FLY}}(v)/(2^m + 1)$ ▷ Always an integer as $(\alpha^{2^m+1})^j = v$
      return $\alpha^{2^m-k+(2^m+1)s^{-1}(j)}$ ▷ $k \in \mathbb{F}_2^m$ so $k + 1 \in \{1, ..., 2^m\}$
    end if
  end if
end function

10 How Did We Obtain the TKlog from π?

We had the intuition for the overall structure of the TKlog because of observations on the TU-decomposition of Biryukov et al. which we present below.

Figure 5 recalls the TU-decomposition of π and describes the notations used below. For each input c of ν_1, i.e. for all $c \in \mathbb{F}_2^4$, we can define two sets A_c and B_c as

$$A_c = \left\{\alpha^{-1}(x, x \odot \mathcal{I}(c)), \forall x \in \mathbb{F}_2^4\right\} ,$$
$$B_c = \left\{\omega(\nu_1(c), y), \forall y \in \mathbb{F}_2^4\right\} .$$

The sets A_c are all vector spaces while the sets B_c are additive cosets of the vector space $\left\{\omega(0, y), \forall y \in \mathbb{F}_2^4\right\}$.

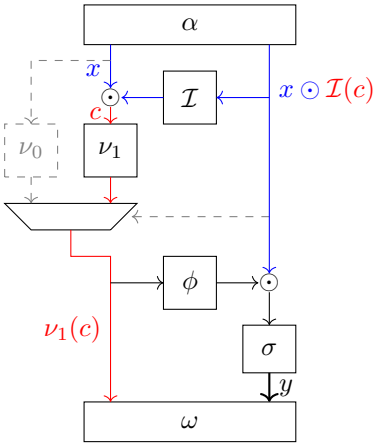


Figure 5: *The propagation of particular vector spaces through π.*

As we can see in Figure 5, if we apply π to all the elements of A_c we obtain 16 elements out of which 15 are in B_c. Furthermore, as we recalled in Equation (1), it holds that $\left\{\alpha^{-1}(x, 0), \forall x \in \mathbb{F}_2^4\right\} = \left\{\omega(0, x), \forall x \in \mathbb{F}_2^4\right\}$. We then have that

$$B_c = \omega(\nu_1(c), 0) \oplus A_0$$

and that A_c is somehow related to A_0 using a finite field multiplication. We also noticed that these sets had a special relationship with the matrix multiplication used in Streebog. Let L be the 64×64 binary matrix used in Streebog and let $[a, 0, ..., 0] \times \mathsf{L} = [a'_0, ..., a'_7]$. If a takes all values in A_c for some $c \in \mathbb{F}_2^4$ then a'_i takes all values in A_{c_i} for some c_i. This property holds regardless of the position of a in the initial vector. Since we knew from the work of Kazymrov and Kazymrova [KK13] that L is somehow related to an MDS matrix with coefficients in $\mathrm{GF}(2^8)$, we deduced that the sets A_c had to have a particular relation with this field.

These observations, in combination with the fact that π is somehow related to a logarithm [PU16], gave us the intuition that the vector spaces A_c and the affine spaces B_c were in fact respectively the multiplicative and additive cosets of a unique vector space of dimension 4 which we quickly identified as the subfield.

This intuition allowed us to write a first very crude decomposition of π which we improved iteratively by re-writing its subcomponents in progressively simpler ways. The final result of this long and tedious process was the decomposition of π as a TKlog we then generalized.

11 Proof of Lemma 3

The following proof is essentially the same as the one in [BPU16b]; we only provide it for the sake of completeness.

Proof. We study $F = \Phi_\kappa \circ \mathscr{T}_{\kappa,h,g_m,s} \circ \mathsf{Split}_\beta^{-1}$. Let a_L, a_R and $b_L \neq 0$ be some m-bit linear masks. By definition, of the LAT and of F, we have

$$\begin{aligned}
&\mathcal{W}_F[a_L||a_R, b_L||0] \\
&= \sum_{r\in \mathrm{GF}(2^m)} \sum_{\ell \in \mathrm{GF}(2^m)} (-1)^{a_L\cdot\ell + a_R\cdot r + (b_L||0)\cdot F(r||\ell)} \\
&= \underbrace{\sum_{\ell\in \mathrm{GF}(2^m)} (-1)^{a_L\cdot\ell + b_L\cdot\tau(\ell)}}_{r=0} + \sum_{r\in \mathrm{GF}(2^m)^*} \sum_{\ell \in \mathrm{GF}(2^m)} (-1)^{a_L\cdot\ell + a_R\cdot r + b_L\cdot\nu(\ell/r)} \ .
\end{aligned}$$

The first term, which corresponds to $r = 0$, is equal to $\mathcal{W}_\tau[a_L, b_L]$. To evaluate the second term (where $r \neq 0$), we set $u = \nu(\ell/r)$ so that $\ell = r \odot \nu^{-1}(u)$. Then we write

$$\begin{aligned}
&\sum_{r\in \mathrm{GF}(2^m)^*} \sum_{u \in \mathrm{GF}(2^m)} (-1)^{a_L\cdot(r\odot\nu^{-1}(u)) + a_R\cdot r + b_L\cdot u} \\
&= \sum_{u\in \mathrm{GF}(2^m)} (-1)^{b_L\cdot u} \left(\sum_{r\in \mathrm{GF}(2^m)} (-1)^{a_L\cdot(r\odot\nu^{-1}(u)) + a_R\cdot r} - \underbrace{1}_{r=0} \right) \\
&= \sum_{u\in \mathrm{GF}(2^m)} (-1)^{b_L\cdot u} \sum_{r\in \mathrm{GF}(2^m)} (-1)^{a_L\cdot(r\odot\nu^{-1}(u)) + a_R\cdot r} - \underbrace{\sum_{u\in \mathrm{GF}(2^m)} (-1)^{b_L\cdot u}}_{=0}
\end{aligned}$$

The sum over r corresponds to the evaluation of a Walsh coefficient of a linear function, namely $r \mapsto r \times \nu^{-1}(u)$. As a consequence, it is equal to 0 unless $r \mapsto a_R\cdot r \oplus a_L\cdot(r\times\nu^{-1}(u))$ is constant.

If $a_L = 0$, then this quantity is never constant. Thus, for $a_L = 0$, we have

$$\mathcal{W}_F[a_L||a_R, b_L||0] = 0 + \mathcal{W}_\tau[0, b_L] = 0 \ .$$

However, if $a_L \neq 0$, then $r \mapsto a_R \cdot r \oplus a_L \cdot (r \odot \nu^{-1}(u))$ is constant for exactly one value of u, in which case the sum is equal to 2^m. Therefore, we have in this case

$$\left(\sum_{u\in \mathrm{GF}(2^m)} (-1)^{b_L\cdot u} \sum_{r\in \mathrm{GF}(2^m)} (-1)^{a_L\cdot(r\odot\nu^{-1}(u)) + a_R\cdot r} \right) \ \in \ \{-2^m, +2^m\}$$

and we conclude that, if $(a_L, a_R) \neq (0, 0)$ and $b_L \neq 0$, then

$$\mathcal{W}_F[a_L||a_R, b_L||0] \ = \ \big(\mathcal{W}_\tau[a_L, b_L] \ \pm \ 2^m\big)[a_L \neq 0] \ .$$

The lemma follows. □

12 Implementation of Our Decomposition

The following SAGE [Dev17] script prints the lookup table of π after generating it using its TKlog decomposition.

```
#!/usr/bin/sage

from sage.all import *

# arithmetic machinery
N = 8
X = GF(2).polynomial_ring().gen()
F = GF(2**8, name="a", modulus=X**8+X**4+X**3+X**2+1)
alpha = F.gen()
xor = lambda x,y : Integer(x).__xor__(Integer(y))

# arbitrary components
s              = [0, 12, 9, 8, 7, 4, 14, 6, 5, 10, 2, 11, 1, 3, 13]
lambda_vectors = [0x12, 0x26, 0x24, 0x30]
cstte          = 0xFC

# subfunction
def kappa(x):
    result = 0
    for j in xrange(0, 4):
        if (x >> j) & 1 == 1:
            result = xor(result, lambda_vectors[j])
    return xor(result, cstte)


# generating pi
# -- pi[0]
pi = [kappa(0)]
# -- pi[x] for x > 0
for x in xrange(1, 2**N):
    l = int(F.fetch_int(x)._log_repr())
    i, j = l % 17, floor(l / 17)
    if i == 0:
        y = kappa(16-j)
    else:
        gf_elmt = (alpha**17)**s[j]
        y = xor(kappa(16-i), gf_elmt.integer_representation())
    pi.append(y)

print pi
```