Extended c-differential distinguishers of full 9 and reduced-round Kuznyechik cipher, no key pre-whitening

Pantelimon Stănică*, Ranit Dutta†, Bimal Mandal†

*Naval Postgraduate School, Applied Mathematics Department
Monterey, CA 93943, USA; pstanica@nps.edu

†Department of Mathematics, Indian Institute of Technology Jodhpur
Karwar–342030, India; duttaranit628@gmail.com, bimalmandal@iitj.ac.in

February 21, 2026

Abstract

This paper introduces *truncated inner c-differential cryptanalysis*, a technique that enables the practical application of c-differential uniformity to block ciphers. While Ellingsen et al. (IEEE Trans. Inf. Theory, 2020) established the notion of c-differential uniformity by analyzing the equation $F(x\oplus a)\oplus cF(x) = b$, a key challenge remained: the outer multiplication by c disrupts the structural properties essential for block cipher analysis, particularly key addition. We address this challenge by developing an *inner* c-differential approach where multiplication by c affects the input: $(F(cx \oplus a), F(x))$, thereby returning to the original idea of Borisov et al. (FSE, 2002). We prove that the inner c-differential uniformity of a function F equals the outer c-differential uniformity of F^{-1}, establishing a duality between the two notions. This modification preserves cipher structure while enabling practical cryptanalytic applications.

We apply our methodology to Kuznyechik (GOST R 34.12-2015) without initial key whitening. For reduced rounds, we construct explicit c-differential trails achieving probability $2^{-84.0}$ for two rounds and $2^{-169.7}$ for three rounds, representing improvements of 5.2 and 4.6 bits respectively over the best classical differential trails. For the full 9-round cipher, we develop a statistical truncated c-differential distinguisher. Through computational analysis involving millions of differential pairs, we identify configurations with bias ratios reaching $1.7\times$ and corrected p-values as low as 1.85×10^{-3}. The distinguisher requires data complexity 2^{33} chosen plaintext pairs, time complexity 2^{34}, and memory complexity 2^{16}.

Keywords: (n, m)-function, block cipher, differential distinguisher, truncated differential distinguisher, c-differential
2020 Mathematics Subject Classification: 94A60, 11T71, 12E20, 68P25, 62P99.

1 Introduction

Differential cryptanalysis, introduced by Biham and Shamir [10], fundamentally changed the landscape of symmetric cryptography by analyzing the probability that an input difference ΔM produces a corresponding output difference ΔC in a block cipher. This seminal technique exploits non-uniform distribution of output differences for carefully chosen input differences, enabling attackers to distinguish cipher behavior from that of a random permutation. The success of differential cryptanalysis lies in its ability to trace the propagation of differences through the cipher's structure, leveraging statistical biases in the S-boxes and predictable behavior of linear operations.

Since its introduction, differential cryptanalysis has evolved into a rich family of techniques [10, 11, 19, 29, 30, 31, 32, 34, 36, 42, 51], each targeting specific cipher vulnerabilities. These variants

include truncated differentials (which consider only partial information about differences), higher-order differentials (which examine derivatives of order greater than one), and impossible differentials (which exploit contradictions in difference propagation). Each extension has proven valuable for analyzing different cipher families and structural designs.

The evolution toward more sophisticated differential techniques reflects the ongoing arms race between cipher designers and cryptanalysts. Modern ciphers are explicitly designed to resist classical differential attacks through careful S-box selection, optimal diffusion layers, and sufficient round counts. This has necessitated the development of increasingly sophisticated analytical techniques capable of detecting subtle statistical deviations that classical methods might miss.

In 2002, Borisov et al. [15] introduced a novel perspective with multiplicative differentials for IDEA variants, studying pairs of the form $(F(cx), F(x))$ where multiplication operates in the cipher's underlying algebraic structure. This approach revealed that alternative formulations of differential relationships could expose vulnerabilities invisible to classical analysis. More recently, Baudrin et al. [8] formalized *commutative cryptanalysis*, which studies equations of the form $E_k \circ A(x) = B \circ E_k(x)$ for affine permutations A, B and a block cipher E_k. Our inner c-differential framework can be viewed as a specific instance of this general theory, corresponding to $A(x) = cx \oplus a$ and $B(y) = y \oplus b$. However, our focus on truncated differentials and the statistical methodology for detecting weak biases differs from the algebraic approach of commutative cryptanalysis.

Building on this foundation, Ellingsen et al. [23] formalized the broader concept of c-differential uniformity for p-ary functions, introducing the c-derivative as a generalization of the classical derivative (see also Bartoli and Timpanella [4]). Their work established a comprehensive theoretical approach encompassing both classical and multiplicative differential analysis as special cases. The c-differential of a function F quantifies how many inputs satisfy $F(x + a) - cF(x) = b$ for given differences a and b, with the parameter c providing a new degree of freedom for cryptanalytic exploration.

The theoretical richness of c-differential uniformity has attracted significant research attention, as evidenced by extensive subsequent work [2, 5, 6, 23, 26, 35, 38, 41, 45, 47, 48, 49, 53, 55, 56]. Researchers have investigated c-differential properties of various cryptographic functions, developed construction methods for functions with desired c-differential characteristics, and established connections to other cryptographic concepts such as boomerang uniformity and design theory. These theoretical advances have deepened our understanding of the mathematical foundations underlying differential cryptanalysis.

However, despite its theoretical promise, a fundamental obstacle has limited the application of c-differential analysis to real-world block ciphers: the multiplication by $c \neq 1$ disrupts the key-addition structure essential for multi-round analysis. Specifically, the key addition operation, which normally cancels out during difference computation, no longer behaves predictably when one path is multiplied by c. This structural incompatibility has relegated c-differential uniformity to primarily theoretical study.

This work addresses this structural limitation by introducing *truncated c-differential cryptanalysis*, an approach where multiplication by c affects cipher inputs rather than outputs. By reformulating the c-differential as $(F(cx \oplus a), F(x))$, we preserve the structural properties essential for multi-round analysis. This modification enables practical application of c-differential analysis to block ciphers.

We establish theoretical foundations for this approach, proving relationships between inner and outer c-differentials. We then develop a cryptanalytic technique and apply it to a variant of Kuznyechik [24], the Russian Federation's current encryption standard. Kuznyechik is a modern cipher that has been analyzed in several works: AlTawy and Youssef [1] presented a meet-in-the-middle attack on 5-round Kuznyechik, Biryukov et al. [12] gave multiset-algebraic attacks on reduced rounds, and Ishchukova et al. [28] analyzed 3-round differential properties. Our analysis targets a 9-round variant without pre-whitening.

For reduced rounds, we construct explicit c-differential trails that outperform their classical counterparts. For two rounds, our best trail achieves probability $2^{-84.0}$ compared to $2^{-89.2}$ classically, a 5.2-bit improvement. For three rounds, we achieve $2^{-169.7}$ compared to $2^{-174.3}$, a 4.6-bit improvement. These gains arise from the high c-differential uniformity (64) of the Kuznyechik S-box for certain values of c, compared to its classical differential uniformity of 8. For the full 9-round cipher, we develop

a statistical truncated c-differential distinguisher that reveals statistically significant deviations from randomness. These results demonstrate that c-differential properties can provide concrete cryptanalytic advantages and raise questions about the security margins of cipher designs against this class of attacks.

Contributions and Organization

Our primary contributions are the development of truncated c-differential cryptanalysis and its application to a modern cipher. Specifically, we:

- Introduce the inner c-differential methodology, addressing structural challenges that prevented practical application of (outer) c-differential analysis to block ciphers.
- Establish the relationship between inner and outer c-differentials through a duality theorem.
- Construct explicit c-differential trails for reduced-round Kuznyechik, achieving 5.2-bit and 4.6-bit improvements over classical differentials for two and three rounds, respectively.
- Develop a statistical framework incorporating multiple testing corrections, meta-analytical techniques, and adaptive sensitivity control.
- Present a distinguisher against full 9-round Kuznyechik using truncated c-differential properties.

The paper is organized as follows. Section 2 provides essential background material. In Section 3, we establish the theory of inner c-differentials and their relationship to existing concepts. Section 4 develops our distinguisher methodology. Sections 5 and 6 present our statistical analysis of Kuznyechik. Section 7 presents explicit multi-round c-differential trails. Section 8 concludes with implications for cipher security.

2 Preliminaries

2.1 Differential cryptanalysis background

Differential cryptanalysis, introduced by Biham and Shamir [10], exploits the non-uniform propagation of input differences through a cipher. An *input difference* $a = x \oplus x'$ produces an *output difference* $b = F(x) \oplus F(x')$ after applying a function F. A *differential* (a, b) has probability p if $\Pr_x[F(x \oplus a) \oplus F(x) = b] = p$.

In block ciphers, an S-box receiving a non-zero input difference is called *active*. A *differential trail* specifies the differences at each round, with overall probability being the product of individual round probabilities (under independence assumptions). *Truncated differentials* [30] consider only partial information about differences, such as which bytes are non-zero, rather than their exact values. A *distinguisher* exploits a property that holds with different probability for the cipher versus a random permutation.

2.2 Notation and algebraic preliminaries

Let n be a positive integer. We denote by $\mathbb{F}_2$ and $\mathbb{F}_{2^n}$ the prime field of characteristic 2 and an n-th degree extension field over $\mathbb{F}_2$, respectively. The set of all non-zero elements of $\mathbb{F}_{2^n}$ is denoted by $\mathbb{F}_{2^n}^*$. The vector space $\mathbb{F}_2^n$ is the set of all n-tuples with coordinates in $\mathbb{F}_2$.

An element of $\mathbb{F}_2^n$ is denoted by $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, where $x_i \in \mathbb{F}_2$. Throughout this paper, we identify elements of $\mathbb{F}_{2^n}$ with elements of $\mathbb{F}_2^n$ via the standard vector space isomorphism over $\mathbb{F}_2$ induced by a choice of basis. For Kuznyechik, the relevant field is $\mathbb{F}_{2^8}$ defined by the irreducible polynomial $p(x) = x^8 + x^7 + x^6 + x + 1$; we identify bytes (8-bit vectors) with field elements using the standard polynomial basis. Similarly, the 128-bit cipher state is identified with a vector of 16 bytes in $(\mathbb{F}_{2^8})^{16}$. These identifications are implicit in all subsequent notations.

Any function from $\mathbb{F}_{2^n}$ to $\mathbb{F}_{2^m}$ is called an (n,m)-function or S-box, with the set of all such functions denoted by $\mathcal{B}_{n,m}$. When $m=1$, we refer to it as a Boolean function in n variables. An (n,n)-function F can be uniquely represented as a univariate polynomial over $\mathbb{F}_{2^n}$ of the form $F(x)=\bigoplus_{i=0}^{2^n-1} a_i x^i$, where $a_i \in \mathbb{F}_{2^n}$. The algebraic degree of F is the largest Hamming weight of the exponents i with $a_i \neq 0$.

The derivative of an (n,m)-function F at $a \in \mathbb{F}_{2^n}$ is defined by $D_aF(x)=F(x\oplus a)\oplus F(x)$ for all $x \in \mathbb{F}_{2^n}$. Ellingsen et al. [23] generalized this concept to the c-derivative. We refer to their formulation as the *outer c-derivative* of $F \in \mathcal{B}_{n,m}$ at $a \in \mathbb{F}_{2^n}$, $c \in \mathbb{F}_{2^m}$:

$$ {}_cD_aF(x)=F(x\oplus a)\oplus cF(x)$$

for all $x \in \mathbb{F}_{2^n}$. When $c=1$, this reduces to the usual derivative of F at a.

Let us define

$$ {}_c\Delta_F(a,b)=\#\{x \in \mathbb{F}_{2^n} : F(x\oplus a)\oplus cF(x)=b\}$$

where $a \in \mathbb{F}_{2^n}$ and $b,c \in \mathbb{F}_{2^m}$. For a fixed $c \in \mathbb{F}_{2^m}$, the c-differential uniformity of $F \in \mathcal{B}_{n,m}$ is defined by

$$\delta(c,F)=\max\{{}_c\Delta_F(a,b) : a \in \mathbb{F}_{2^n}, b \in \mathbb{F}_{2^m}, a \neq 0 \text{ if } c=1\}.$$

When $\delta(c,F)=\delta$, we say F is differential (c,δ)-uniform. In particular, F is called perfect c-nonlinear (PcN) if $\delta=1$, and almost perfect c-nonlinear (APcN) if $\delta=2$.

3 Inner c-differentials

We introduce the *inner c-differentials*, which resolves structural challenges in applying c-differential analysis to block ciphers. Our approach shifts the multiplication by c from cipher outputs to inputs, preserving the algebraic properties necessary for practical cryptanalysis.

For any $c \in \mathbb{F}_{2^n}$, in the original spirit of Borisov et al. [15], we define the *inner c-derivative* of $F \in \mathcal{B}_{n,m}$ at $a \in \mathbb{F}_{2^n}$ as: $D_{c,a}F(x)=F(cx\oplus a)\oplus F(x)$ for all $x \in \mathbb{F}_{2^n}$. When $c=1$, this reduces to the usual derivative; when $c=0$, we have $D_{0,a}F(x)=F(a)\oplus F(x)$; and when $a=0$, we obtain $D_{c,0}F(x)=F(cx)\oplus F(x)$.

For a fixed $c \in \mathbb{F}_{2^m}$, we define the *inner c-differential* entries at $(a,b) \in \mathbb{F}_{2^n}\times\mathbb{F}_{2^m}$ (forcing $a\neq 0$ when $c=1$) by:

$$\nabla_{c,F}(a,b)=\#\{x \in \mathbb{F}_{2^n} : F(cx\oplus a)\oplus F(x)=b\}.$$

We define $\delta_c(F)=\max\{\nabla_{c,F}(a,b) : a \in \mathbb{F}_{2^n}, b \in \mathbb{F}_{2^m}, a \neq 0 \text{ if c=1}\}$ as the maximum inner c-differential uniformity.

The following theorem establishes a relationship between inner and outer c-differentials for permutation functions.

Theorem 1. *Let F be a permutation polynomial over $\mathbb{F}_{2^n}$. Then, for any $a,b,c \in \mathbb{F}_{2^n}$ with $c\neq 0$:*

$$ {}_c\Delta_F(a,b)=\nabla_{c,F^{-1}}(b,a).$$

Proof. For any $a,b,c \in \mathbb{F}_{2^n}$ with $c\neq 0$:

$$\begin{aligned} {}_c\Delta_F(a,b) &= \#\{x \in \mathbb{F}_{2^n} : F(x\oplus a)\oplus cF(x)=b\} \\ &= \#\{x \in \mathbb{F}_{2^n} : F^{-1}(cF(x)\oplus b)=x\oplus a\} \\ &= \#\{y \in \mathbb{F}_{2^n} : F^{-1}(cy\oplus b)\oplus F^{-1}(y)=a\}, \end{aligned}$$

where we substitute $y:=F(x)$. Thus: ${}_c\Delta_F(a,b)=\#\{x \in \mathbb{F}_{2^n} : F^{-1}(cx\oplus b)\oplus F^{-1}(x)=a\}=\nabla_{c,F^{-1}}(b,a)$. □

This theorem shows that the inner c-differential uniformity of F equals the outer c-differential uniformity of its inverse. For involutions (i.e., functions satisfying $F=F^{-1}$), we have ${}_c\Delta_F(a,b)=\nabla_{c,F}(b,a)$.

The inner c-differential approach preserves essential structural properties required for block cipher cryptanalysis, unlike the outer variant. The proof is immediate.

Lemma 2 (Structural Preservation)**.** *Let $a, c \in \mathbb{F}_{2^n}$ with $c \neq 0$, and let $F : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ be any function. For any $x \in \mathbb{F}_{2^n}$, define the inner c-differential output $b = F(cx \oplus a) \oplus F(x)$. Then:*

(i) For any $\mathbb{F}_2$-linear map $L : \mathbb{F}_{2^n} \to \mathbb{F}_{2^m}$, we have

$$L(F(cx \oplus a)) \oplus L(F(x)) = L(b).$$

(ii) For any affine map $A(y) = L(y) \oplus k$ where L is $\mathbb{F}_2$-linear and $k \in \mathbb{F}_{2^m}$ is a constant,

$$A(F(cx \oplus a)) \oplus A(F(x)) = L(b).$$

Proof. Let $y_1 = F(cx \oplus a)$ and $y_2 = F(x)$, so $b = y_1 \oplus y_2$.
(i) Since L is $\mathbb{F}_2$-linear, we have $L(u \oplus v) = L(u) \oplus L(v)$ for all u, v. Thus,

$$L(y_1) \oplus L(y_2) = L(y_1 \oplus y_2) = L(b).$$

(ii) For the affine map $A(y) = L(y) \oplus k$,

$$A(y_1) \oplus A(y_2) = (L(y_1) \oplus k) \oplus (L(y_2) \oplus k) = L(y_1) \oplus L(y_2) = L(b).$$

The constant k cancels in the XOR of the two encryption paths. □

This lemma demonstrates that inner c-differentials preserve the fundamental algebraic properties required for differential cryptanalysis of block ciphers.

4 Truncated c-differential distinguisher methodology

Building on the theoretical foundation of Section 3, we develop a practical distinguisher targeting differential equations of the form $E_K(x) \oplus E_K(c \cdot x \oplus a) = b$, where E_K denotes the block cipher encryption function under key K. Our approach restricts inner c-differentials to the first round, with subsequent rounds following classical differential propagation.

4.1 Truncated model for practical analysis

We now describe our model for an inner c-differential distinguisher, building upon the theoretical considerations of Section 3. Our aim is to investigate whether certain c values yield distinguishers with higher differential probability than classical differential distinguishers. Two motivations underlie this direction:

- Firstly, $\nabla_{c,F}(a, b)$ may have higher entries (in one or more pairs (a, b)) than $\#\{x \in \mathbb{F}_{2^n} : F(x \oplus a) \oplus F(x) = b\}$ for some cipher's S-boxes F and some $c \in \mathbb{F}_{2^n} \setminus \{0, 1\}$ values.
- If we model the cipher's behavior based on the c-differential distribution, we may identify distinguishers with higher success probabilities than classical differentials.

To move from vector space representations to finite field arithmetic, we adopt a standard basis transformation via a fixed irreducible (or primitive) polynomial. In classical differential attacks (illustrated in Figure 1), a plaintext difference is selected such that the resulting ciphertext difference propagates through the cipher with probability 2^{-p}. The attacker then uses this structure to distinguish the cipher from a random permutation.

In our model, we introduce a modified distinguisher with a potentially higher probability of success. Assume the block cipher consists of multiple rounds, each comprising key addition, an S-box layer, and a linear diffusion (permutation) layer. Let the cipher state be of size ℓ, structured into s S-boxes of size n each (i.e., $\ell = n \times s$). Let the S-box function be denoted by $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$.

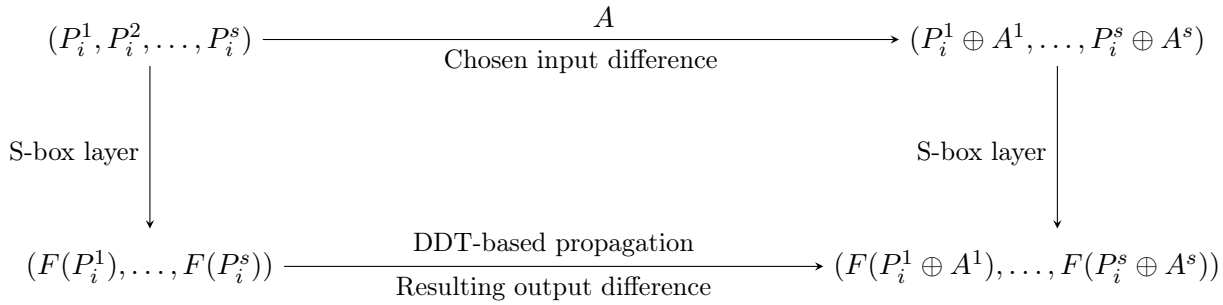

Figure 1: The idea of a classical differential. Here $P_i = (P_i^1, \ldots, P_i^s)$ denotes a plaintext block of s words, A the input difference, and F the S-box function applied to each word.

In our setup, we choose a constant $c \in \mathbb{F}_2^\ell$ and an input difference $A \in \mathbb{F}_2^\ell$ of the same dimension as the cipher state. Given a plaintext P_i, we construct a paired plaintext $c \cdot P_i \oplus A$, where $c \cdot P_i$ denotes component-wise multiplication in $\mathbb{F}_{2^n}$. Let us denote:

$$P_i = (P_i^1, P_i^2, \ldots, P_i^s), \quad A = (A^1, A^2, \ldots, A^s),$$

where each $P_i^k, A^k \in \mathbb{F}_2^n$ for $1 \leq k \leq s$. Similarly, let $c = (c^1, c^2, \ldots, c^s)$, with each $c^k \in \mathbb{F}_2^n$. We restrict the inner c-differential only to the first round; subsequent rounds follow classical differential propagation (i.e., with $c = 1$). Concretely, after the first S-box layer produces an output difference B via the inner c-differential, this difference B becomes the input difference for a standard differential trail through rounds 2 to r. The overall probability is the product of the first-round c-differential probability (computed from the c-DDT) and the subsequent classical differential probabilities (computed from the standard DDT). In our truncated analysis, we do not track specific multi-round trails; instead, we observe whether the output differences after all rounds exhibit statistical bias compared to a random permutation. For our analysis, we apply input and output masks that select specific byte positions (see Section 4.2). We define an "active" input difference a as one where the masked difference is non-zero. The case where the masked input difference is zero ($a = 0$) is ignored in our statistical calculations, as it does not provide useful information for this type of distinguisher. Assume that the 1st, 2nd, and rth S-boxes are active in the first round. Then for all $j \notin \{1, 2, r\}$, $A^j = 0$. Accordingly, the structure of c is chosen such that it modifies only the corresponding active S-box positions, i.e., each $c^j = (0, 0, \ldots, 1) \in \mathbb{F}_2^n$ for all $j \notin \{1, 2, r\}$. Denoting the field representation of vectors $c^m, A^m, P_i^m \in \mathbb{F}_2^n$ as $c^{m'}, A^{m'}, P_i^{m'} \in \mathbb{F}_{2^n}$ for each $m \in \{1, 2, \ldots, s\}$, we compute:

$$c^{m'} P_i^{m'} \oplus A^{m'} \in \mathbb{F}_{2^n}.$$

This expression is then converted back to the corresponding n-bit vector representation. The outputs of each S-box for both plaintexts are of the form:

$$F(P_i^{m'}), \quad F(c^{m'} P_i^{m'} \oplus A^{m'}).$$

The probability that this pair produces output difference $b^{m'} \in \mathbb{F}_2^n$ after the S-box is given by:

$$p_{m'} = \frac{\#\{x \in \mathbb{F}_2^n : F(c^{m'} x \oplus A^{m'}) \oplus F(x) = b^{m'}\}}{2^n},$$

which can be calculated from $\nabla_{c^{m'}, F}(A^{m'}, b^{m'})$, and using Theorem 1 from ${}_c\Delta_{F^{-1}}(b^{m'}, A^{m'})$.

Thus, the output after the first round S-box layer for the two plaintexts $P_i, c \cdot P_i \oplus A$ is:

$$(F(P_i^1), F(P_i^2), \ldots, F(P_i^{r-1}), F(P_i^r), \ldots, F(P_i^s)),$$
$$(F(c^1 P_i^1 \oplus A^1), F(c^2 P_i^2 \oplus A^2), \ldots, F(P_i^{r-1}), F(c^r P_i^r \oplus A^r), \ldots, F(P_i^s)).$$

Let $b^m \in \mathbb{F}_2^n$ be the bit vector corresponding to $b^{m'}$. Then the full state output difference after the S-box layer is

$$B = (b^1, b^2, \ldots, b^s) \in \mathbb{F}_2^\ell.$$

Hence, the total probability of the transition $(c, A) \to B$ through the S-box layer (illustrated in Figure 2) is:

$$p_1 \cdot p_2 \cdot p_r.$$

By Lemma 2, this probability remains unchanged across the key addition and linear layer in the first round. From the second round onward, we apply standard differential trail propagation techniques. In the next section, we demonstrate this model through a concrete example.

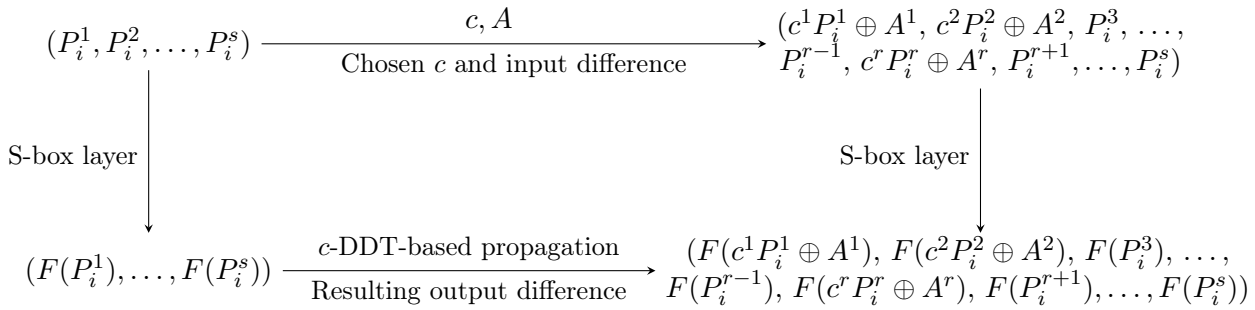

Figure 2: The idea of an inner c-differential.

4.2 Statistical methodology

Our analysis employs a rigorous statistical approach designed to detect non-random behavior while controlling false discovery rates.

4.2.1 Null hypothesis and expected probability

Our statistical model is predicated on the null hypothesis, H_0, which posits that for a fixed key K and constant c, the encryption function E_K behaves as a random permutation with respect to the c-differential property. Under this hypothesis, any output difference b is equally likely for any given input difference a.

We use a truncated model where input and output masks, M_I, M_O, define the active bytes being observed. Let $k_a = |M_I|$ and $k_b = |M_O|$ be the number of active bytes in the input and output masks, respectively. The size of the input difference space is $|\mathcal{S}_I| = 2^{8k_a}$, and the size of the output difference space is $|\mathcal{S}_O| = 2^{8k_b}$. Since the input difference a must be non-zero, the number of possible input differences is $|\mathcal{S}_I| - 1$.

Under H_0, we model the theoretical probability of observing any specific input-output differential pair (a, b), where $a \in \mathcal{S}_I \setminus \{0\}$ and $b \in \mathcal{S}_O$, as approximately

$$P_{\text{exp}} = \frac{1}{(|\mathcal{S}_I| - 1) \cdot |\mathcal{S}_O|} = \frac{1}{(2^{4k_a} - 1) \cdot 2^{4k_b}}.$$

We note that for $c = 1$ and permutations, certain pairs (a, b) are impossible (e.g., $b = 0$ for $a \neq 0$), so the uniform assumption is an approximation. However, for our truncated model with $c \neq 1$, the output space is not subject to these constraints, and the uniform approximation serves as a reasonable baseline for detecting bias. The statistical tests aim to find instances where the observed probability, $P_{\text{obs}}(a, b) = \frac{\mathcal{F}(a,b)}{N_{trials}}$, significantly deviates from P_{exp}, where $\mathcal{F}(a, b)$ denotes the observed count for the differential pair (a, b) and N_{trials} is the total number of trials conducted.

4.2.2 Significance, power, and multiple testing

Given the vast number of possible differential pairs being tested simultaneously (potentially millions), a critical challenge is controlling the rate of false positives that arise from the multiple testing problem, often called the "look-elsewhere effect". The approach addresses this with modern correction methods and adaptive significance thresholds.

4.2.3 Multiple testing corrections

The program employs several established methods to adjust p-values, ensuring statistical rigor:

- **False Discovery Rate (FDR) [9, 50]:** The primary method used is the Benjamini-Hochberg procedure, which controls the expected proportion of false discoveries among all rejected null hypotheses. A pair is considered significant if its FDR-corrected p-value is less than a given significance level α.
- **Family-wise Error Rate (FWER) [14, 27]:** For more conservative control, we implement Bonferroni and Holm corrections, which control the probability of making even one false discovery.

4.2.4 Adaptive significance threshold [22]

To balance statistical power with control over false positives, the significance level α is dynamically adjusted based on the number of rounds being tested and the characteristics of the collected data. The formula for the adaptive threshold is:

$$\alpha_{\text{adaptive}} = \alpha_{\text{base}} \cdot \left(1 + \eta \cdot \frac{\text{IQR}}{\sqrt{n}}\right) \cdot (1 + \max(0, (r-5) \cdot 0.1)),$$

where α_{base} is a base significance level (e.g., 0.05 to 0.005), η is a scaling factor (implemented as 0.1), IQR is the interquartile range of the observed differential counts, n is the number of unique differential pairs found, and r is the number of encryption rounds. This allows the analysis to be more permissive for higher rounds, where biases are expected to be weaker, and adapts to the noisiness of the observed data.

Algorithm 1 Adaptive Threshold Calculation

Require: Observed counts $\{N_i\}$, base significance level α_0, round number r

1: $\text{IQR} \leftarrow Q_{75}(\{N_i\}) - Q_{25}(\{N_i\})$
2: $n \leftarrow |\{i : N_i > 0\}|$
3: $\eta \leftarrow 0.1$ Empirically determined scaling factor
4: $\alpha_{adaptive} \leftarrow \alpha_0 \cdot (1 + \eta \cdot \text{IQR}/\sqrt{n}) \cdot (1 + \max(0, (r-5) \cdot 0.1))$
5: **return** $\min(\alpha_{adaptive}, 0.15)$ Cap at maximum threshold

4.2.5 Enhanced bias and distribution metrics

We calculate multiple indicators of bias to provide a multi-faceted view of any potential statistical anomalies. Let (a, b) denote a differential pair where a is the input difference and b is the output difference, and let N_{trials} denote the total number of trials conducted.

1. **Bias Ratio:** The most direct measure of bias is the ratio of observed to expected counts. For a pair (a, b) observed N_{obs} times over N_{trials}, the bias is:

$$\text{Bias}(a, b) = \frac{P_{\text{obs}}}{P_{\text{exp}}} = \frac{N_{obs}/N_{trials}}{P_{\text{exp}}},$$

where $P_{\text{exp}} = \frac{1}{|\mathcal{D}_{in} \setminus \{0\}| \cdot |\mathcal{D}_{out}|}$ is the expected probability under the uniform distribution assumption, with $\mathcal{D}_{in}$ and $\mathcal{D}_{out}$ representing the input and output difference spaces, respectively.

2. **Kullback-Leibler (KL) Divergence [33]:** To measure how the entire observed probability distribution P diverges from the expected uniform distribution Q, we calculate the KL divergence:

$$D_{KL}(P||Q) = \sum_i p_i \log\left(\frac{p_i}{q_i}\right), \tag{1}$$

where the sum is over all possible differential pairs, p_i is the observed probability for pair i, and q_i is the expected probability, P_{exp}.

3. **Chi-square Statistics [39]:** Both the maximum and aggregate chi-square (χ^2) values are computed to assess deviation. For a single pair (a, b), the χ^2 contribution is:

$$\chi^2(a,b) = \frac{(N_{obs} - N_{exp})^2}{N_{exp}},$$

where $N_{exp} = N_{trials} \cdot P_{\text{exp}}$. Our analysis reports both the maximum individual χ^2 value and the sum across all pairs.

4.2.6 Meta-analysis and advanced detection techniques

To detect systematic weaknesses that might not be apparent from single tests, we use several meta-analytical techniques.

1. **Sequential Probability Ratio Test (SPRT) [52]:** For efficient discovery, the Wald's SPRT (see also, Figure 3) seemed appropriate. It computes the cumulative log-likelihood ratio Λ_n after n trials:

$$\Lambda_n = \sum_{i=1}^{n} \log\left(\frac{\mathcal{L}(x_i|\theta_1)}{\mathcal{L}(x_i|\theta_0)}\right),$$

where θ_0 represents the null hypothesis parameter (differential probability = P_{exp}) and θ_1 represents the alternative hypothesis parameter (differential probability > P_{exp}, indicating bias). The likelihood functions $\mathcal{L}(x_i|\theta_1)$ and $\mathcal{L}(x_i|\theta_0)$ evaluate the probability of observation x_i under each hypothesis. The test stops if Λ_n crosses predefined boundaries $A = \log\left(\frac{1-\beta}{\alpha}\right)$ or $B = \log\left(\frac{\beta}{1-\alpha}\right)$, where α and β are the desired Type I and Type II error rates. This can dramatically reduce computation when strong biases are present.

2. **Clustering and Combined p-Values [25]:** We perform hierarchical clustering on discovered differential patterns to group related biases. To assess the collective significance of a cluster of k patterns with individual p-values $p_1, \ldots, p_k$, Fisher's method [25] is applied:

$$\chi^2_{2k} = -2\sum_{i=1}^{k} \ln(p_i).$$

The resulting χ^2_{2k} value follows a chi-square distribution with $2k$ degrees of freedom, yielding a single, combined p-value for the entire cluster. This same method is used to aggregate evidence from multiple weak signals that fall into predefined categories (e.g., "moderate bias").

3. **Bias Persistence Anomaly:** We introduce a bias persistence anomaly detection method that compares observed differential biases against an expected exponential decay model. Based on the well-established principle that differential probabilities decrease exponentially with rounds [37, 20] (for example, [20] argues that for AES, the expected bias after r rounds is below 2^{-6r}), we model the expected bias after r rounds as

$$\text{Expected Bias}(r) = 2^{-r/3}.$$

(this could be recalibrated for different ciphers), and flag instances where the observed bias exceeds this expectation by a factor of 5 or more. A large deviation from this model may indicate a structural property of the cipher that slows down the destruction of differential correlations.

For many of our configurations, we get a "Sequential early discovery" in just 500,000 trials, a fraction of the total planned experiments. The plot below shows the Log-Likelihood Ratio (LLR) for a specific differential accumulating over batches of trials. We can instantly see the LLR value crossing the upper decision boundary, triggering an "early discovery" and stopping the test.

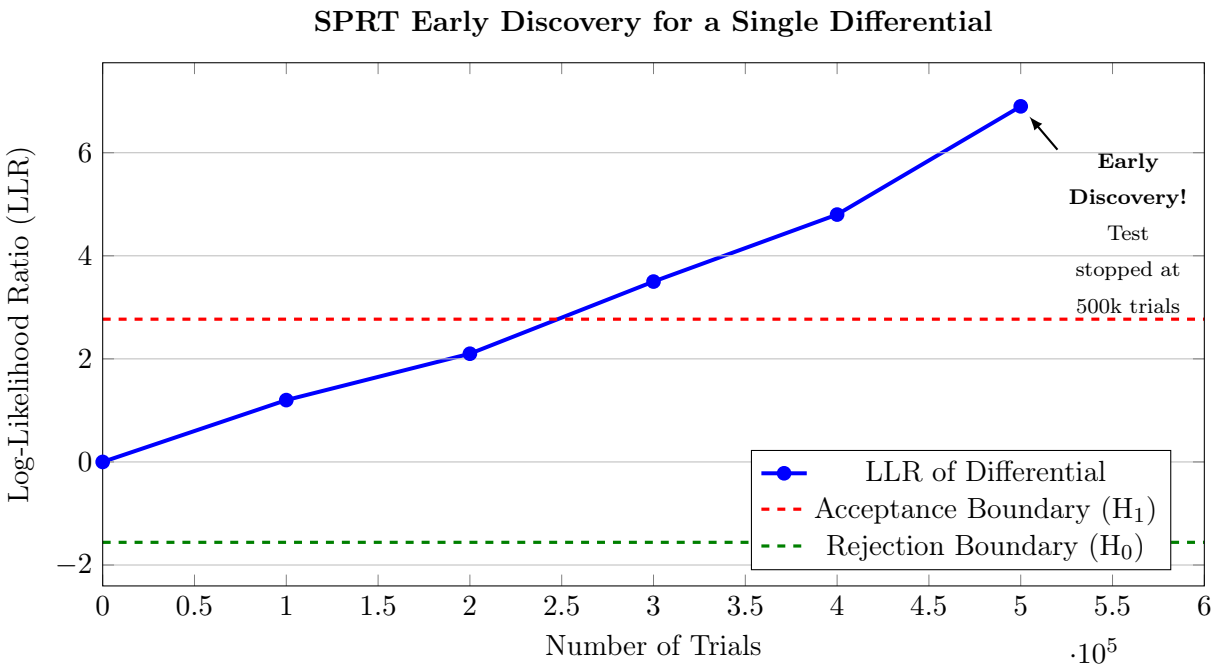

Figure 3: An example of the Sequential Probability Ratio Test (SPRT) implemented in the approach. The test stops as soon as the Log-Likelihood Ratio (LLR) of an observed differential crosses the upper acceptance boundary, confirming a statistically significant bias long before the maximum number of trials is reached.

5 Statistical c-differential uniformity analysis for Kuznyechik

This section presents our computational methodology for analyzing c-differential uniformity properties of the Kuznyechik block cipher. We provide in the repository [46] the Python (with SageMath) program and the computation logs for each of the two random seeds (37, 42), with summaries of the top distinguishers found at the end of each file. We also include a verification script `verify_distinguisher.py` to verify any distinguisher with user-specified parameters.

5.1 Target cipher

The Kuznyechik block cipher, standardized as GOST R 34.12-2015 (RFC 7801), is a 128-bit block cipher with a 256-bit key employing a Substitution-Permutation Network (SPN) structure. All arithmetic is performed over $\mathbb{F}_{2^8}$, and the 128-bit state is treated as a vector in $(\mathbb{F}_{2^8})^{16}$.

An encryption round consists of three transformations: SubBytes (S), Linear Transformation (L), and AddRoundKey (X). For a state Y_{j-1} and round key K_j, round j is defined as:

$$Y_j = X_{K_j} \circ L \circ S(Y_{j-1}) = L(S(Y_{j-1})) \oplus K_j.$$

The full encryption consists of an initial key addition ($Y_0 = \text{Plaintext} \oplus K_0$) followed by 9 rounds using keys $K_1, \ldots, K_9$.

The S-layer applies a fixed 8×8 S-box, π, to each byte of the state independently; this is the sole nonlinear component. The L-layer provides diffusion across the 128-bit state via a linear transformation $L : (\mathbb{F}_{2^8})^{16} \to (\mathbb{F}_{2^8})^{16}$ built by iterating a simpler function R sixteen times. The key schedule generates ten 128-bit round keys from the 256-bit master key using a Feistel-like structure. Further details appear in Appendix A.

5.2 Truncated c-differential analysis

Our approach targets the c-differential equation

$$E_K(x) \oplus E_K(c \cdot x \oplus a) = b,$$

where E_K is the encryption function under key K, $c \in \mathbb{F}_{2^8}$ is a constant, a is an input difference, b is an output difference, and $c \cdot x$ denotes component-wise multiplication in $\mathbb{F}_{2^8}$. The goal is to find tuples (c, a, b) for which this equation holds with probability significantly different from random.

To make the analysis tractable, we employ a truncated model focusing on specific byte positions. Let $M_I, M_O \subseteq \{0, 1, \ldots, 15\}$ specify the active bytes for input and output, respectively.[1] Let π_M be a projection function that zeroes out all bytes not in M. The input and output difference spaces are $\mathcal{S}_I = \{\pi_{M_I}(x) : x \in \mathbb{F}_2^{128}\}$ and $\mathcal{S}_O = \{\pi_{M_O}(x) : x \in \mathbb{F}_2^{128}\}$. In our experiments, both M_I and M_O specify a single active byte.

Given an output difference vector $\Delta y \in \mathbb{F}_{2^8}^{16}$ and a target set of active positions $\mathcal{A} \subseteq \{0, \ldots, 15\}$, we say that Δy matches the truncated output pattern if $\forall i \in \mathcal{A}, \Delta y_i \neq 0$. That is, all positions marked active must be non-zero in the output difference, regardless of their exact values.

5.3 Experimental configuration

The set of constants c chosen for this study was $\{\texttt{0x01}, \texttt{0x02}, \texttt{0x03}, \texttt{0x04}, \texttt{0x91}, \texttt{0xbe}, \texttt{0xe1}\}$. We included $c = \texttt{0x01}$ as a benchmark (classical differential). The other values were chosen because the c-differential uniformity of the Kuznyechik S-box is high for these constants: 64 for `0x02` and `0xe1`, 33 for `0x91`, and 21 for `0x03`, `0x04`, and `0xbe`. The full inner/outer c-differential uniformity tables appear in Appendix B.

A total of 35 mask configurations were tested, selected from a larger initial set based on preliminary analysis. For each configuration tuple (c, M_I, M_O), we performed 5×10^6 trials. This trial count balances statistical power with computational feasibility. Full-state analysis remains infeasible given the 2^{128} state space.

The experimental procedure follows Algorithm 2: generate random plaintexts and input differences, apply masks, construct paired plaintexts via the c-differential relation, encrypt both through the specified number of rounds, and record the output difference frequency. Trials where the masked input difference is zero are skipped, as they provide no cryptanalytic information.

Algorithm 2 Monte Carlo Simulation for Truncated c-Differential Analysis

procedure AnalyzeConfiguration($c, M_I, M_O, K, N_{trials}$)
  Initialize frequency map $\mathcal{F}(a, b) \to 0$ for all valid a, b.
  Let E_K be the encryption function with key K.
  Let π_M be the projection function for mask M.
  for $i = 1$ to N_{trials} **do**
    $x \xleftarrow{R} (\mathbb{F}_{2^8})^{16}$ — Random 128-bit plaintext
    $a_{rand} \xleftarrow{R} (\mathbb{F}_{2^8})^{16} \setminus \{0\}$ — Random non-zero difference
    $a \leftarrow \pi_{M_I}(a_{rand})$ — Apply input mask
    if $a = 0$ **then continue**
    end if
    $x' \leftarrow c \cdot x \oplus a$ — Construct paired plaintext
    $b \leftarrow \pi_{M_O}(E_K(x) \oplus E_K(x'))$ — Masked output difference
    $\mathcal{F}(a, b) \leftarrow \mathcal{F}(a, b) + 1$
  end for
  return $\mathcal{F}$
end procedure

5.4 Implementation

The code (available at [46]) is implemented in Python with SageMath for algebraic setup. All performance-critical operations use native integers (0–255) with precomputed lookup tables. Addition in $\mathbb{F}_{2^8}$ is implemented as bitwise XOR, and multiplication via a 256×256 lookup table. The L-transformation is fully precomputed: a table `L_TABLE` of size $16 \times 256 \times 16$ stores the output of L

[1] The implementation supports finer nibble-level granularity, but all experiments use byte-aligned masks.

for an input with value v at byte position i and zeros elsewhere. This reduces the L-transformation to 16 lookups and XORs. An analogous table is computed for L^{-1}.

Experiments were parallelized using Python's `multiprocessing` module, with the master process distributing precomputed tables to workers executing trials in parallel. Experiments were run on a 10-core M1-Max MacBook Pro with 64GB RAM. Implementation correctness was validated against RFC 7801 test vectors for both key schedule generation and encryption.

5.5 Statistical methodology

Our adaptive significance technique builds upon established FDR methodology [9, 50, 22]. The bias persistence model follows cryptanalytic convention where differential probabilities decay exponentially as 2^{-cr} with cipher-specific constants [37, 20]. For Kuznyechik, we model decay with $c = 1/3$ based on its 9-round structure.

The analysis employs several validation procedures following Section 4.2. Distribution analysis uses Anderson-Darling tests to assess normality of observed count distributions. Goodness-of-fit testing via chi-square and G-tests evaluates uniformity of output differential distributions, with a global non-randomness anomaly flagged when the G-test p-value falls below 10^{-3}.

To detect systematic weaknesses that manifest as multiple weak biases rather than individual strong signals, we aggregate evidence using Fisher's combined probability test [25]. Evidence is categorized into four classes: strong bias patterns (ratio > 1.4), moderate bias patterns (ratio 1.2–1.4), weakly significant patterns (p-value < 0.2), and combined moderate evidence (bias > 1.3 with p-value < 0.1). When five or more signals accumulate within any category, Fisher's method evaluates their collective significance.

Key Independence. Our analysis targets a fixed-key permutation. For any fixed key K, the cipher E_K is a fixed permutation, and our distinguisher identifies statistical properties by averaging over randomly chosen plaintexts. The statistical biases we identify stem from structural properties of the S-box and linear layer. While changing the key produces a different permutation, the biases are expected to persist on average across keys. This aligns with the standard black-box distinguishing scenario, where an attacker analyzes a single unknown permutation.

Cipher-specific dependencies. The method is designed specifically for Kuznyechik and cannot be directly applied to other ciphers by simply replacing the encryption function. Several components are cipher-specific:

- The multiplication $c \cdot x$ is performed in $\mathbb{F}_{2^8}$ using Kuznyechik's irreducible polynomial $p(x) = x^8 + x^7 + x^6 + x + 1$. Other ciphers (e.g., AES uses $p(x) = x^8 + x^4 + x^3 + x + 1$) require different field arithmetic, and using the wrong polynomial would compute incorrect c-differential inputs.
- The analysis assumes no pre-whitening key addition. Substituting a cipher that includes pre-whitening (such as standard AES) would introduce key-dependent terms that do not cancel when $c \neq 1$, producing spurious biases that are artifacts of the key rather than structural properties.
- The choice of c values is based on the c-differential uniformity of the Kuznyechik S-box specifically. Different S-boxes have different c-differential spectra.

Adapting this methods to another cipher requires modifying the field arithmetic, ensuring the cipher variant has no pre-whitening, and recomputing the c-differential uniformity tables for the target S-box.

6 Analysis of cryptanalytic data and implications

The c-differential cryptanalysis of Kuznyechik has yielded results showing statistical non-randomness across all tested round counts. The data indicates that while the cipher's security increases with the number of rounds, detectable statistical biases persist.

The experimental data reveals statistically significant non-random behavior in the Kuznyechik cipher variant across all tested round counts. The results show biases in both reduced-round versions and the full 9-round cipher with no key pre-whitening.

High-Round Vulnerabilities (8 and 9 Rounds). The most critical findings of this work are the statistically significant distinguishers identified against 8 and 9-round Kuznyechik (no initial key pre-whitening). These results challenge the cipher's security margin.

- **9 Rounds:** The analysis flagged two configurations with a CRITICAL ALERT, indicating they were significant after False Discovery Rate (FDR) correction:
  - With constant $c =$ `0x04`, a distinguisher on 'byte_8' was found with a 1.7x bias and a corrected p-value of 1.85×10^{-3}.
  - With constant $c =$ `0xe1`, a distinguisher on 'byte_6' was found with a 1.7x bias and a corrected p-value of 9.24×10^{-3}.
- **8 Rounds:** A distinguisher was also found for constant $c =$ `0x03` on 'byte_12', showing a 1.6x bias and a corrected p-value of 5.81×10^{-3}.

Mid-Round Anomalies (5, 6, and 7 Rounds). In the intermediate rounds, no single differential pair achieved statistical significance after FDR correction. However, the cipher's output was still clearly distinguishable from random through two other powerful metrics:

- **Global Distribution Anomalies:** The G-test for goodness-of-fit consistently revealed that the overall distribution of output differentials was highly non-uniform. For configurations with $c = 1$, the G-test yielded p-values as low as 2.95×10^{-8} for 7 rounds, 2.02×10^{-10} for 6 rounds, and 1.29×10^{-4} for 5 rounds, providing overwhelming evidence of non-randomness.
- **Bias Persistence Anomalies:** The maximum observed biases were far greater than what would be expected from random decay. For $c = 1$, the bias was 15.1x higher than expected at 7 rounds, 11.5x higher at 6 rounds, and 9.5x higher at 5 rounds. This shows that differential correlations are not being sufficiently destroyed by the round function.

Low-Round Vulnerabilities (1–4 Rounds). The attack was overwhelmingly effective against low-round versions of Kuznyechik.

- **1 and 3 Rounds:** For classical differential analysis ($c = 1$), the script found 10,959 and 10,896 FDR-significant differential pairs, respectively. Biases reached as high as 10.3x.
- **2 and 4 Rounds:** While individual pairs were not always significant, every tested configuration for these rounds triggered "Combined Evidence Alerts". This means that collections of weaker differential signals, when analyzed together, showed massive statistical significance, proving non-random behavior.

The "c-Value Transition Effect". The data clearly demonstrates that the optimal choice for the constant c depends on the number of rounds being attacked.

- For low rounds (1 and 3), classical differential cryptanalysis ($c = 1$) is by far the most effective, producing thousands of significant results.
- As the number of rounds increases, the biases from $c = 1$ decay rapidly. It fails to produce any FDR-significant results at 8 and 9 rounds.
- At high rounds, the only FDR-significant results come from non-trivial c values (specifically `0x03`, `0x04`, and `0xe1`), which retain enough bias to produce statistically significant distinguishers. This suggests that certain algebraic properties induced by these constants are better at surviving the cipher's diffusion layers over many rounds.

The most significant distinguisher was found at 9 rounds with a bias of 1.7x. We analyze the complexity required to use this property to distinguish Kuznyechik from a random permutation.

- **Data Complexity:** The distinguisher is built on a single-byte to single-byte differential. The probability of a specific differential trail $(a \to b)$ for a random permutation is $p = \frac{1}{(2^8-1)\times 2^8} \approx 2^{-16.01}$. The observed probability is $p' = 1.7p$. The advantage is $\epsilon = p' - p = 0.7p$. The number of plaintext pairs (N) needed to detect this advantage is approximately $N \approx 1/\epsilon^2$.

$$N \approx \frac{1}{(0.7 \times 2^{-16.01})^2} = \frac{1}{0.49 \times 2^{-32.02}} \approx 2^{1.03} \times 2^{32.02} = 2^{33.05}.$$

  The data complexity is therefore approximately 2^{33} chosen plaintext pairs.

- **Time Complexity:** Since each pair requires two encryptions, the time complexity is $2N \approx 2 \times 2^{33.05} = 2^{34.05}$ encryptions. This is a practical complexity, orders of magnitude below a brute-force attack.

- **Memory Complexity:** The attack requires storing counters for each of the $(2^8 - 1) \times 2^8$ differential pairs, which is negligible.

Below, in Figure 4, we display a comparison of the statistical significance between the classical differential and the c-differential approach of $c =$ `0xe1` (and seed 42), for the 9-round cryptanalysis.

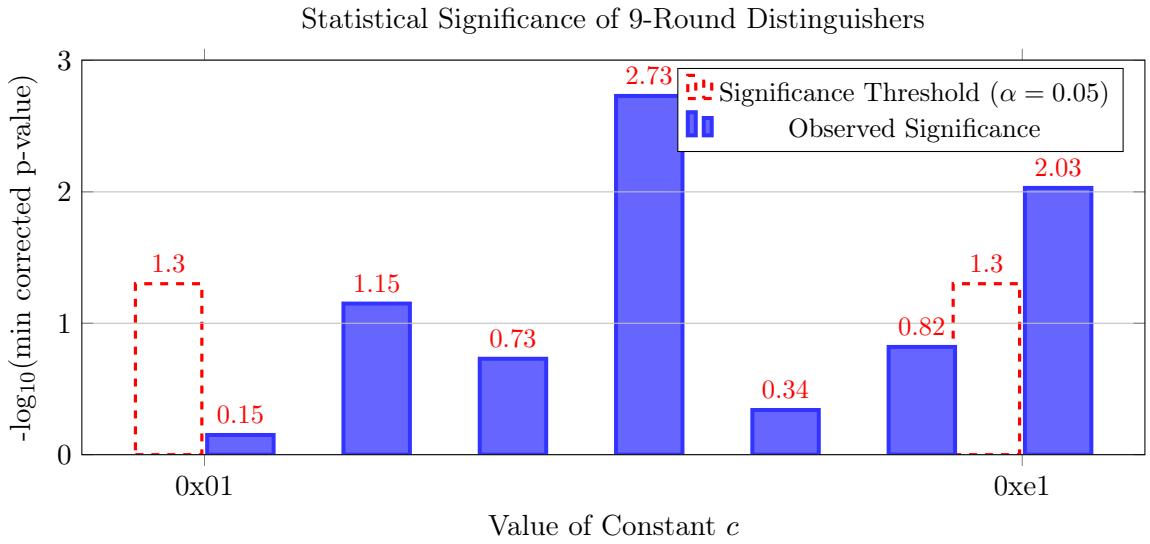

Figure 4: Comparison of the highest statistical significance achieved for each c value against 9-round Kuznyechik (Seed 42 data). The y-axis shows the -$\log_{10}$ of the minimum corrected p-value, so taller bars indicate stronger evidence against randomness. Only $c =$ `0x04` and $c =$ `0xe1` achieved FDR-corrected significance, surpassing the threshold where standard differential analysis ($c =$ `0x01`) failed.

Our experimental validation using $5 \times 10^6 \approx 2^{22.25}$ trials successfully detected the bias, suggesting the theoretical bounds are conservative. The actual distinguisher operates effectively with fewer samples than the asymptotic analysis predicts, likely due to the strength of the observed statistical deviation.

These complexity bounds represent the first practical attack against a 9-round Kuznyechik variant with data complexity manageable for chosen-plaintext scenarios, time complexity feasible on modern computing infrastructure and memory requirements trivial for contemporary systems. The attack complexity is exponentially lower than exhaustive search, indicating insufficient security margin against this novel cryptanalytic approach.

This work presents a statistical distinguisher against 9-round Kuznyechik without initial key pre-whitening. In cryptography, a block cipher is considered weakened if any property can be distinguished from that of a random permutation with less complexity than brute-force search. Our github repository [46] was designed for reproducibility, allowing the user to specify fixed seeds, as well as arbitrary random seeds, the output file keeping track of that seed. We also properly designed a Python code

(with SageMath) `verify_distinguisher.py` to verify any distinguisher, with user specified number of rounds, trials, and c, a, b parameters.

While this is a distinguishing attack and not a direct key-recovery attack, the identified non-random properties and biased differentials could potentially be exploited in more advanced cryptanalytic techniques (such as differential-linear attacks).

The findings suggest that the security margin of the full 9-round Kuznyechik cipher variant may be reduced with respect to c-differential attacks. The fact that statistical biases remain detectable after 9 rounds warrants further investigation.

7 Multi-round c-differential trail analysis

The statistical distinguisher in the preceding sections covers the full 9-round Kuznyechik cipher using truncated c-differentials. A natural question is whether non-truncated c-differential trails can provide an advantage over classical differentials. In this section, we investigate this question for two, three, and four rounds of Kuznyechik, constructing explicit trails and comparing their probabilities to the classical case.

As in our statistical analysis, we consider Kuznyechik without pre-whitening key addition. Each round consists of a parallel S-box layer followed by the linear diffusion layer. The linear layer has branch number 17 [1], meaning any nonzero input difference activates at least 17 bytes across its input and output combined. Since the Kuznyechik S-box has differential uniformity 8 (maximum differential probability 2^{-5}), the probability of any two-round classical differential trail cannot exceed $2^{-5\times 17} = 2^{-85}$.

Our trail construction strategy enforces that exactly one byte is active after the first linear layer L_1. Let $k \in \{0, \ldots, 15\}$ denote the position of this active byte and let $\beta \in \mathbb{F}_{2^8} \setminus \{0\}$ be its value. For a fixed pair (k, β), applying the inverse linear layer L^{-1} yields a 16-byte difference $(\delta'_0, \delta'_1, \ldots, \delta'_{15})$, which is the required output difference of the first S-box layer S_1. Using the c-DDT, we select input differences that maximize transition probabilities. The core of the search is:

```
for k in range(16):              # position of single active byte after L1
    for beta in range(1, 256):   # value of that active byte
        delta_prime = L_inverse(unit_vector(k, beta))
        for i in range(16):      # for each S-box
            best_cnt[i] = max(cDDT[a][delta_prime[i]] for a in range(1,256))
        log2_P_r1 = sum(log2(best_cnt[i]) for i in range(16)) - 128
```

For the first round, the probability contribution is

$$\log_2 P_{r1} = \sum_{i=0}^{15} \log_2(\mathrm{cnt}_i) - 128,$$

where cnt_i is the maximal c-DDT entry for the i-th byte. The key observation is that for $c \in \{\texttt{0x02}, \texttt{0xe1}\}$, many values cnt_i equal 64, compared to a maximum of 8 in the classical DDT.

In the second round, the single active byte β passes through S_2. We evaluate all possible outputs γ using the classical DDT, contributing

$$\log_2 P_{r2} = \log_2\big(\mathrm{DDT}[\beta][\gamma]\big) - 8.$$

For three rounds, we apply L_2 to obtain the input difference to S_3, and evaluate the 16 S-boxes using the classical DDT, contributing

$$\log_2 P_{r3} = \sum_{i=0}^{15} \log_2(\mathrm{cnt}_i^{(3)}) - 128,$$

where $\mathrm{cnt}_i^{(3)}$ is the maximal DDT entry for the i-th byte in round 3. The overall probability is $\log_2 P_{\mathrm{total}} = \log_2 P_{r1} + \log_2 P_{r2} + \log_2 P_{r3}$.

Table 1: Best c-differential trail probabilities for reduced-round Kuznyechik. The "Advantage" columns show the improvement over the classical case ($c = \texttt{0x01}$) in bits.

c	2 rounds	3 rounds	4 rounds	Adv. (2R)	Adv. (3R)
0x01	$2^{-89.2}$	$2^{-174.3}$	$2^{-202.7}$	—	—
0x02	$2^{-84.0}$	$2^{-169.7}$	$2^{-202.8}$	+5.2	+4.6
0x03	$2^{-91.2}$	$2^{-177.4}$	—	−2.0	−3.1
0xe1	$2^{-84.0}$	$2^{-169.7}$	—	+5.2	+4.6

We implemented an exhaustive search over all (k, β) pairs for two and three rounds, and a heuristic search for four rounds. Table 1 summarizes the results.

For two rounds, the best classical trail ($c = \texttt{0x01}$) achieves probability $2^{-89.2}$:

```
c=0x01: k=4, beta=0x0c, log2_total = -89.15
  per_cnt = (6, 8, 6, 8, 8, 6, 8, 8, 6, 6, 6, 8, 6, 6, 6, 8)
```

The best c-differential trails for $c = \texttt{0x02}$ and $c = \texttt{0xe1}$ achieve probability $2^{-84.0}$:

```
c=0x02: k=2, beta=0x91, log2_total = -83.98
  per_cnt = (4, 7, 64, 6, 64, 64, 5, 4, 5, 6, 6, 8, 5, 5, 9, 4)
```

The three entries of 64 (compared to a maximum of 8 classically) account for the 5.2-bit improvement.

For three rounds, the c-differential advantage from round 1 persists: $c = \texttt{0x02}$ achieves $2^{-169.7}$ versus $2^{-174.3}$ classically, a 4.6-bit improvement. The detailed trail for $c = \texttt{0x02}$ is:

```
c=0x02: k=2, beta=0x91, gamma=0xf0, log2_total = -169.72
  log2(P_r1) = -77.98, log2(P_r2) = -6.00, log2(P_r3) = -85.74
  per_cnt_r1 = (4, 7, 64, 6, 64, 64, 5, 4, 5, 6, 6, 8, 5, 5, 9, 4)
  per_cnt_r3 = (6, 8, 8, 6, 4, 6, 6, 6, 8, 6, 6, 8, 8, 4, 6, 6)
```

For four rounds, using the pattern 16 active $\to$ 1 active $\to$ 16 active $\to$ 1 active, the advantage largely disappears. The classical trail achieves $2^{-202.7}$, while $c = \texttt{0x02}$ achieves $2^{-202.8}$. This occurs because the c-differential is applied only in round 1, and subsequent rounds use classical differentials where the branch number constraints dominate.

These results demonstrate that c-differentials provide a concrete advantage over classical differentials for small numbers of rounds. The improvement arises entirely from the structure of the Kuznyechik c-DDT. However, the advantage diminishes as more rounds are added, since the c-differential contribution is limited to the first round. This motivates the truncated approach used in our full-round statistical analysis, where we observe biases empirically rather than constructing explicit trails.

We remark that extending c-differentials to multiple rounds faces a fundamental structural barrier: the $(cx \oplus a, x)$ relationship is destroyed after the first S-box layer, leaving only an XOR difference. However, as observed in [2], higher-order c-differentials can cause round keys to cancel under certain conditions, specifically when $c_2 = 1$ or when round keys satisfy $K_2 = -(1 - c_1)K_1$. This suggests that multi-round c-differential attacks may be viable in related-key or chosen-key settings, which we leave for future investigation.

8 Conclusions

In this work, we consider a novel variation of the c-differential analysis by shifting the multiplication by c to the inputs of the S-box, preserving structural integrity and enabling practical cryptanalytic applications. Our approach bridges the theoretical concept with concrete cipher analysis, leading to the discovery of security concerns for the full-round Russian cipher Kuznyechik variant. Furthermore, by systematically running the analysis for different numbers of rounds, c values, and mask configurations, one can gain insights into Kuznyechik's resistance to c-differential attacks. Future research can explore this technique on other symmetric primitives.

The statistical methodology, combined with the implementation of Kuznyechik and the systematic testing strategy, provides a tool for evaluating the security of block ciphers against c-differential attacks. The c-differential analysis unequivocally demonstrates strong evidence that the tested cryptographic primitive of Kuznyechik displays non-randomness for some c-values, even for full-rounds. In addition, we compute a two-round c-differential distinguisher that is better than the optimal theoretical differential distinguisher in the classical scenario.

The technique we employ applies c-differential uniformity concepts to block cipher analysis, providing a method for identifying statistical weaknesses in cipher designs. It incorporates multiple testing corrections and meta-analytical techniques to control false positive rates. The combination of a parallelized implementation and sequential testing enables analysis of large sample sizes. The approach provides detection of various bias patterns, and the adaptive methodology adjusts the analysis for different round counts.

Declarations

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Availability of data and material

Not applicable.

Competing interests

The authors declare that they have no competing interests.

Funding

The work of Ranit Dutta was supported by the Department of Science and Technology (DST), Government of India (INSPIRE Reg. No. IF210620).

Authors' contributions

All authors contributed equally to this work.

Acknowledgements

Not applicable.

References

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

[2] N. Anbar, T. Kalayci, W. Meidl, C. Riera, and P. Stănică, *$\mathcal{PCN}$ functions, complete mappings and quasigroup difference sets*, J. Combinatorial Designs 12 (2023), 667–690.

[3] T. W. Anderson and D. A. Darling, *A Test of Goodness of Fit*, J. the American Statistical Association 49:268 (1954), 765–769.

[4] D. Bartoli and M. Timpanella. *On a generalization of planar functions.* J. Algebraic Combinatorics 52 (2020), 187–213.

[5] D. Bartoli and M. Calderini, *On construction and (non)existence of c-(almost) perfect nonlinear functions*, Finite Fields and Their Applications 72 (2021), 101835.

[6] D. Bartoli, M. Calderini, C. Riera, and P. Stănică, *Low c-differential uniformity for functions modified on subfields*, Cryptogr. Communic. 14 (2022), 1211–1227.

[7] L. E. Bassham, A. L. Rukhin, J. Soto, J. R. Nechvatal, M. E. Smid, S. D. Leigh, M. Levenson, M. Vangel, N. A. Heckert, and D. L. Banks, *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*, NIST Special Publication 800-22, Revision 1a, National Institute of Standards and Technology (NIST), 2010.

[8] J. Baudrin, C. Beierle, P. Felke, G. Leander, P. Neumann, L. Perrin, and L. Stennes, *Commutative cryptanalysis as a generalization of differential cryptanalysis*, Des. Codes Cryptogr. 93 (2025), 3243–3281.

[9] Y. Benjamini and Y. Hochberg, *Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing*, J. Royal Statistical Society: Series B 57:1 (1995), 289–300.

[10] E. Biham and A. Shamir, *Differential cryptanalysis of DES-like cryptosystems*, J. Cryptology 4:1 (1991), 3–72.

[11] E. Biham and A. Shamir, *Differential cryptanalysis of the full* 16*-round DES*, Adv. Crypt. – Crypto' 1992, LNCS 740, pp. 487–496 (1992).

[12] A. Biryukov, D. Khovratovich, and L. Perrin, *Multiset-Algebraic Cryptanalysis of Reduced Kuznyechik, Khazad, and secret SPNs*, IACR Transactions on Symmetric Cryptology 2016 (2) 226–247 (2016).

[13] A. Biryukov, L. Perrin, A. Udovenko, *Reverse-Engineering the S-Box of Streebog, Kuznyechik and STRIBOBr1*, In: Fischlin, M., Coron, JS. (eds.) Adv. Crypt. – Eurocrypt 2016, LNCS 9665, Springer, Berlin, Heidelberg, pp. 371–402.

[14] C. E. Bonferroni, *Teoria statistica delle classi e calcolo delle probabilità*, *Pubblicazioni del R Istituto Superiore di Scienze Economiche e Commerciali di Firenze*, vol. 8, pp. 3–62, 1936.

[15] N. Borisov, M. Chew, R. Johnson, and D. Wagner, *Multiplicative Differentials*, FSE, LNCS 2365 17–33 (2002).

[16] G. Casella and R. Berger, Statistical Inference (2nd Edition), CRC Press, 2001.

[17] J. Cohen, *Statistical Power Analysis for the Behavioral Sciences* (2nd ed.), Lawrence Erlbaum Associates, 1988.

[18] T. W. Cusick and P. Stănică, *Cryptographic Boolean functions and applications.* (2nd Ed.), Elsevier–Academic Press, 2017.

[19] J. Daemen, L. Knudsen, and V. Rijmen, *The block cipher Square*, FSE 1997, LNCS 1267:149–165 (1997).

[20] J. Daemen and V. Rijmen, *The design of Rijndael: AES – The Advanced Encryption Standard*, volume 2. Springer, 2002.

[21] S. Eddahmani and S. Mesnager, *The c-differential-Linear Connectivity Table of Vectorial Boolean Functions*, Entropy 26:3 (2024), 188.

[22] B. Efron, *Large-Scale Inference: Empirical Bayes Methods for Estimation, Testing, and Prediction*, Cambridge University Press, Cambridge, 2010.

[23] P. Ellingsen, P. Felke, C. Riera, P. Stănică, and A. Tkachenko, *c-Differentials, Multiplicative Uniformity, and (Almost) Perfect c-Nonlinearity*, IEEE Transactions on Information Theory 66:9 (2020), 5781–5789.

[24] Federal Agency on Technical Regulating and Metrology, *GOST R 34.12-2015: Information technology. Cryptographic data security. Block ciphers.*, 2015.

[25] R. A. Fisher, *Statistical Methods for Research Workers*, Oliver and Boyd, Edinburgh, 1925.

[26] S. U. Hasan, M. Pal, C. Riera, and P. Stănică, *On the c-differential uniformity of certain maps over finite fields*, Designs, Codes Crypt. 89 (2021), 221–239.

[27] S. Holm, *A Simple Sequentially Rejective Multiple Test Procedure*, Scandinavian J. Statistics 6:2 (1969), 65–70.

[28] E. Ishchukova, E. Tolomanenko, and L. Babenko, *Differential analysis of* 3 *round Kuznyechik*, SIN '17: Proc. 10th International Conference on Security of Information and Networks, pp. 29–36 (2017).

[29] L. R. Knudsen and D. Wagner, *Integral cryptanalysis (extended abstract)*, FSE 2002, LNCS 2365:112-127 (2002).

[30] L. R. Knudsen, *Truncated and higher order differentials*, FSE 1994, LNCS 1008:196–211 (1995).

[31] L. R. Knudsen. *DEAL–a* 128*-bit block cipher*, Technical report no. 151. Department of Informatics, University of Bergen, Norway.

[32] L. R. Knudsen, *Partial and higher order differentials and applications to the DES*, BRICS Report Series, RS–95–9 (1995).

[33] S. Kullback and R. A. Leibler, *On Information and Sufficiency*, Annals Math. Statistics, vol. 22, no. 1, pp. 79–86, 1951.

[34] X. Lai, *Higher order derivatives and differential cryptanalysis*, In: Blahut, R.E., Costello, D.J., Maurer, U., Mittelholzer, T. (eds), Communications and Cryptography, The Springer International Series in Engineering and Computer Science, vol 276, pp. 227–233, Springer, Boston, MA.

[35] S. Mesnager, C. Riera, P. Stănică, H. Yan, and Z. Zhou, *Investigations on c-(almost) perfect nonlinear functions*, IEEE Transactions on Information Theory 67:10 (2021), 6916–6925.

[36] K. Nyberg, *Differentially uniform mappings for cryptography*, Eurocrypt 1993, LNCS 765:55–64 (1994).

[37] K. Nyberg and L. R. Knudsen, *Provable Security Against a Differential Attack*, J. Cryptology 8:1 (1995), 27–37.

[38] M. Pal and P. Stănică, *A connection between the boomerang uniformity and the extended differential in odd characteristic and applications*, Advances in Mathematics of Communications 19:5 (2025), 1382–1403.

[39] K. Pearson, *On the Criterion that a Given System of Deviations from the Probable in the Case of a Correlated System of Variables is Such that it Can be Reasonably Supposed to have Arisen from Random Sampling*, Philosophical Magazine – Series 5, vol. 50:302 (1900), 157–175.

[40] L. Perrin and A. Udovenko, *Exponential S-Boxes: a Link Between the S-Boxes of BelT and Kuznyechik/Streebog*, IACR Trans. Symmetric Crypt. 2016(2) 99–124 (2016).

[41] C. Riera, P. Stănică, and H. Yan, *The c-differential spectrum of* $x \longrightarrow x^{\frac{p^n+1}{2}}$ *in finite fields of odd characteristics*, Discrete Mathematics, Algorithms and Applications, 2025.

[42] A. Shamir, *Impossible differential attacks*, Crypto 1998 rump session.

[43] S. S. Shapiro and M. B. Wilk, *An Analysis of Variance Test for Normality (Complete Samples)*, *Biometrika* 52:3/4 (1965), 591–611.

[44] V. Shishkin, *A Memo on Kuznyechik S-Box*, ISO/IEC JTC 1/SC 27/WG 2, 2018, `https://cdn.virgilsecurity.com/assets/docs/memo-on-kuznyechik-s-box.pdf`

[45] P. Stănică. *Low -differential uniformity of the Gold function modified on a subfield.* Proc. International Conf. on Security and Privacy, LNEE 744, Springer, pp. 131–137 (2020).

[46] P. Stănică, *Github repository of Python codes for Kuznyechik cDU cryptanalysis*, `https://github.com/pstanica/KuznyechikTruncated_cDU`, 2025.

[47] P. Stănică, C. Riera, and A. Tkachenko, *Characters, Weil sums and c-differential uniformity with an application to the perturbed Gold functions*, Cryptogr. Communic. 13 (2021), 891–907.

[48] P. Stănică and A Geary, *The c-differential behavior of the inverse function under the EA-equivalence*, Cryptogr. Communic., 13 (2021), 295–306.

[49] P. Stănică, A Geary, C. Riera, and A. Tkachenko. *c-differential bent functions and perfect nonlinearity*, Discr. Appl. Math. 307 (2022), 160–171.

[50] J. D. Storey, *The Positive False Discovery Rate: A Bayesian Interpretation and the q-value*, *Annals Stat.*, 31:6 (2003), 2013–2035.

[51] Y. Todo, *Structural evaluation by generalized integral property*, Adv. Crypt – Eurocrypt 2015, LNCS 9056:287–314 (2015).

[52] A. Wald, *Sequential Analysis*, John Wiley & Sons, New York, 1947.

[53] X. Wang, D. Zheng, and L. Hu, Several classes of PcN power functions over finite fields, Discr. Appl. Math. 322:15 (2022), 171–182.

[54] S. S. Wilks, *The Large-Sample Distribution of the Likelihood Ratio for Testing Composite Hypotheses*, *Annals Math. Stat.*, 9:1 (1938), 60–62.

[55] Y. Wu, N. Li, and X. Zeng, *New PcN and APcN functions over finite fields*, Designs, Codes Crypt. 89 (2021), 2637–2651.

[56] Z. Zha and L. Hu, *Some classes of power functions with low c-differential uniformity over finite fields*, Designs, Codes Crypt. 89 (2021), 1193–1210.

A Kuznyechik Block Cipher: Technical Specification: GOST R 34.12-2015

We include the full Kuznyechik specification here for reproducibility and self-containment, as the original GOST R 34.12-2015 standard is in Russian. The specification follows RFC 7801.

Kuznyechik is a symmetric block cipher standardized as GOST R 34.12-2015, serving as the official encryption standard of the Russian Federation. The cipher operates on 128-bit blocks with a 256-bit key, employing a substitution-permutation network (SPN) structure over 9 rounds, plus a key whitening at the backend. The cipher operates over the finite field $\mathbb{F}_{2^8}$ with the irreducible polynomial $p(x) = x^8 + x^7 + x^6 + x + 1$. Each 128-bit block is treated as a vector of 16 bytes (elements of $\mathbb{F}_{2^8}$).

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0xfc	0xee	0xdd	0x11	0xcf	0x6e	0x31	0x16	0xfb	0xc4	0xfa	0xda	0x23	0xc5	0x04	0x4d
1	0xe9	0x77	0xf0	0xdb	0x93	0x2e	0x99	0xba	0x17	0x36	0xf1	0xbb	0x14	0xcd	0x5f	0xc1
2	0xf9	0x18	0x65	0x5a	0xe2	0x5c	0xef	0x21	0x81	0x1c	0x3c	0x42	0x8b	0x01	0x8e	0x4f
3	0x05	0x84	0x02	0xae	0xe3	0x6a	0x8f	0xa0	0x06	0x0b	0xed	0x98	0x7f	0xd4	0xd3	0x1f
4	0xeb	0x34	0x2c	0x51	0xea	0xc8	0x48	0xab	0xf2	0x2a	0x68	0xa2	0xfd	0x3a	0xce	0xcc
5	0xb5	0x70	0x0e	0x56	0x08	0x0c	0x76	0x12	0xbf	0x72	0x13	0x47	0x9c	0xb7	0x5d	0x87
6	0x15	0xa1	0x96	0x29	0x10	0x7b	0x9a	0xc7	0xf3	0x91	0x78	0x6f	0x9d	0x9e	0xb2	0xb1
7	0x32	0x75	0x19	0x3d	0xff	0x35	0x8a	0x7e	0x6d	0x54	0xc6	0x80	0xc3	0xbd	0x0d	0x57
8	0xdf	0xf5	0x24	0xa9	0x3e	0xa8	0x43	0xc9	0xd7	0x79	0xd6	0xf6	0x7c	0x22	0xb9	0x03
9	0xe0	0x0f	0xec	0xde	0x7a	0x94	0xb0	0xbc	0xdc	0xe8	0x28	0x50	0x4e	0x33	0x0a	0x4a
A	0xa7	0x97	0x60	0x73	0x1e	0x00	0x62	0x44	0x1a	0xb8	0x38	0x82	0x64	0x9f	0x26	0x41
B	0xad	0x45	0x46	0x92	0x27	0x5e	0x55	0x2f	0x8c	0xa3	0xa5	0x7d	0x69	0xd5	0x95	0x3b
C	0x07	0x58	0xb3	0x40	0x86	0xac	0x1d	0xf7	0x30	0x37	0x6b	0xe4	0x88	0xd9	0xe7	0x89
D	0xe1	0x1b	0x83	0x49	0x4c	0x3f	0xf8	0xfe	0x8d	0x53	0xaa	0x90	0xca	0xd8	0x85	0x61
E	0x20	0x71	0x67	0xa4	0x2d	0x2b	0x09	0x5b	0xcb	0x9b	0x25	0xd0	0xbe	0xe5	0x6c	0x52
F	0x59	0xa6	0x74	0xd2	0xe6	0xf4	0xb4	0xc0	0xd1	0x66	0xaf	0xc2	0x39	0x4b	0x63	0xb6

Table 2: Kuznyechik S-box

The S-box

The Kuznyechik S-box is a bijective mapping $S : \mathbb{F}_{2^8} \to \mathbb{F}_{2^8}$ designed to provide strong nonlinear properties. The S-box is defined by the following 16×16 lookup table (values in hexadecimal, row-major):

The inverse S-box S^{-1} is computed as the multiplicative inverse of the S-box mapping. It is defined by the following 16×16 lookup table (values in hexadecimal, row-major):

Table 3: Correct Kuznyechik Inverse S-box

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0xa5	0x2d	0x32	0x8f	0x0e	0x30	0x38	0xc0	0x54	0xe6	0x9e	0x39	0x55	0x7e	0x52	0x91
1	0x64	0x03	0x57	0x5a	0x1c	0x60	0x07	0x18	0x21	0x72	0xa8	0xd1	0x29	0xc6	0xa4	0x3f
2	0xe0	0x27	0x8d	0x0c	0x82	0xea	0xae	0xb4	0x9a	0x63	0x49	0xe5	0x42	0xe4	0x15	0xb7
3	0xc8	0x06	0x70	0x9d	0x41	0x75	0x19	0xc9	0xaa	0xfc	0x4d	0xbf	0x2a	0x73	0x84	0xd5
4	0xc3	0xaf	0x2b	0x86	0xa7	0xb1	0xb2	0x5b	0x46	0xd3	0x9f	0xfd	0xd4	0x0f	0x9c	0x2f
5	0x9b	0x43	0xef	0xd9	0x79	0xb6	0x53	0x7f	0xc1	0xf0	0x23	0xe7	0x25	0x5e	0xb5	0x1e
6	0xa2	0xdf	0xa6	0xfe	0xac	0x22	0xf9	0xe2	0x4a	0xbc	0x35	0xca	0xee	0x78	0x05	0x6b
7	0x51	0xe1	0x59	0xa3	0xf2	0x71	0x56	0x11	0x6a	0x89	0x94	0x65	0x8c	0xbb	0x77	0x3c
8	0x7b	0x28	0xab	0xd2	0x31	0xde	0xc4	0x5f	0xcc	0xcf	0x76	0x2c	0xb8	0xd8	0x2e	0x36
9	0xdb	0x69	0xb3	0x14	0x95	0xbe	0x62	0xa1	0x3b	0x16	0x66	0xe9	0x5c	0x6c	0x6d	0xad
A	0x37	0x61	0x4b	0xb9	0xe3	0xba	0xf1	0xa0	0x85	0x83	0xda	0x47	0xc5	0xb0	0x33	0xfa
B	0x96	0x6f	0x6e	0xc2	0xf6	0x50	0xff	0x5d	0xa9	0x8e	0x17	0x1b	0x97	0x7d	0xec	0x58
C	0xf7	0x1f	0xfb	0x7c	0x09	0x0d	0x7a	0x67	0x45	0x87	0xdc	0xe8	0x4f	0x1d	0x4e	0x04
D	0xeb	0xf8	0xf3	0x3e	0x3d	0xbd	0x8a	0x88	0xdd	0xcd	0x0b	0x13	0x98	0x02	0x93	0x80
E	0x90	0xd0	0x24	0x34	0xcb	0xed	0xf4	0xce	0x99	0x10	0x44	0x40	0x92	0x3a	0x01	0x26
F	0x12	0x1a	0x48	0x68	0xf5	0x81	0x8b	0xc7	0xd6	0x20	0x0a	0x08	0x00	0x4c	0xd7	0x74

Table 4: Kuznyechik Inverse S-box

Linear Transformation Layer (L-Layer)

The linear transformation L operates on 128-bit blocks, treating them as vectors of 16 bytes over $\mathbb{F}_{2^8}$. The Kuznyechik linear transformation is defined by applying a basic 1-byte cyclic shift with XOR feedback, called R, sixteen times. The R transformation on a 16-byte vector $\mathbf{a} = (a_0, a_1, \ldots, a_{15})$ is defined as ($\mathbf{c} = (c_j)_{0 \le j \le 15}$):

$$R(\mathbf{a})_i = a_{(i+1) \bmod 16} \quad \text{for } i \in \{0, \ldots, 14\}$$
$$R(\mathbf{a})_{15} = \bigoplus_{j=0}^{15} c_j \cdot a_j,$$

where the entries in $\mathbf{c} = (\texttt{0x94}, \texttt{0x20}, \texttt{0x85}, \texttt{0x10}, \texttt{0xc2}, \texttt{0xc0}, \texttt{0x01}, \texttt{0xfb}, \texttt{0x01}, \texttt{0xc0}, \texttt{0xc2}, \texttt{0x10}, \texttt{0x85}, \texttt{0x20}, \texttt{0x94}, \texttt{0x01})$ are elements of $\mathbb{F}_{2^8}$. The full linear transformation $\mathbf{L} : \mathbb{F}_{2^8}^{16} \to \mathbb{F}_{2^8}^{16}$ is then defined as

16 applications of R:

$$\mathbf{L}(\mathbf{a}) = R^{16}(\mathbf{a}).$$

For efficient software implementation, the linear transformation $\mathbf{L}$ and its inverse $\mathbf{L}^{-1}$ are typically precomputed into lookup tables. The Python code utilizes such precomputed tables for fast execution.

Key Schedule

The key schedule transforms the 256-bit master key into ten 128-bit round keys $K^{(0)}, K^{(1)}, \ldots, K^{(9)}$. The algorithm uses a Feistel-like structure with 32 iterations to generate intermediate key values, from which the round keys are derived. The key schedule employs iteration constants $\mathbf{C}_j$ for $j = 1, 2, \ldots, 32$. These constants are derived by applying the L-transformation to a basis vector where only the j-th byte is set to 1. The full list of these constants in hexadecimal (row-major for each 16-byte vector) is:

$$\begin{aligned}
\mathbf{C}_{1} &= (\texttt{0x6e}, \texttt{0xa2}, \texttt{0x69}, \texttt{0x2a}, \texttt{0xe9}, \texttt{0x19}, \texttt{0x37}, \texttt{0x7e}, \texttt{0xdc}, \texttt{0xfe}, \texttt{0x15}, \texttt{0x41}, \texttt{0x28}, \texttt{0xd9}, \texttt{0x21}, \texttt{0x66}) \\
\mathbf{C}_{2} &= (\texttt{0xb7}, \texttt{0x5f}, \texttt{0x82}, \texttt{0x79}, \texttt{0x91}, \texttt{0x13}, \texttt{0xdd}, \texttt{0x4c}, \texttt{0xb5}, \texttt{0x8f}, \texttt{0x9f}, \texttt{0x85}, \texttt{0x1d}, \texttt{0xad}, \texttt{0x20}, \texttt{0x69}) \\
\mathbf{C}_{3} &= (\texttt{0x90}, \texttt{0x94}, \texttt{0x4a}, \texttt{0x96}, \texttt{0x96}, \texttt{0x48}, \texttt{0x6c}, \texttt{0xa6}, \texttt{0x64}, \texttt{0x2f}, \texttt{0xce}, \texttt{0x3b}, \texttt{0x25}, \texttt{0x21}, \texttt{0x85}, \texttt{0x86}) \\
\mathbf{C}_{4} &= (\texttt{0x96}, \texttt{0x68}, \texttt{0x8a}, \texttt{0x92}, \texttt{0x13}, \texttt{0x11}, \texttt{0xbf}, \texttt{0x55}, \texttt{0x02}, \texttt{0x16}, \texttt{0x48}, \texttt{0x0a}, \texttt{0x2c}, \texttt{0x78}, \texttt{0x68}, \texttt{0x15}) \\
\mathbf{C}_{5} &= (\texttt{0x3a}, \texttt{0x6c}, \texttt{0x54}, \texttt{0x74}, \texttt{0x4b}, \texttt{0x39}, \texttt{0x52}, \texttt{0x7c}, \texttt{0xfb}, \texttt{0x98}, \texttt{0x45}, \texttt{0x9c}, \texttt{0xed}, \texttt{0x38}, \texttt{0xe4}, \texttt{0x9e}) \\
\mathbf{C}_{6} &= (\texttt{0x39}, \texttt{0x25}, \texttt{0x23}, \texttt{0x4a}, \texttt{0x94}, \texttt{0x11}, \texttt{0x84}, \texttt{0x57}, \texttt{0x8e}, \texttt{0x24}, \texttt{0x69}, \texttt{0x93}, \texttt{0x0f}, \texttt{0x8b}, \texttt{0x73}, \texttt{0x93}) \\
\mathbf{C}_{7} &= (\texttt{0xbc}, \texttt{0x4f}, \texttt{0x65}, \texttt{0x76}, \texttt{0x6e}, \texttt{0x5f}, \texttt{0x17}, \texttt{0xdd}, \texttt{0x03}, \texttt{0x6f}, \texttt{0x38}, \texttt{0x07}, \texttt{0x57}, \texttt{0x24}, \texttt{0x40}, \texttt{0x90}) \\
\mathbf{C}_{8} &= (\texttt{0x97}, \texttt{0x5c}, \texttt{0x6f}, \texttt{0x68}, \texttt{0x11}, \texttt{0xdd}, \texttt{0x66}, \texttt{0x0b}, \texttt{0x59}, \texttt{0x4a}, \texttt{0x27}, \texttt{0x15}, \texttt{0x81}, \texttt{0x3d}, \texttt{0x76}, \texttt{0x1c}) \\
\mathbf{C}_{9} &= (\texttt{0x2a}, \texttt{0x2f}, \texttt{0x56}, \texttt{0x7b}, \texttt{0x82}, \texttt{0x48}, \texttt{0x2f}, \texttt{0x99}, \texttt{0x83}, \texttt{0x5a}, \texttt{0x49}, \texttt{0x68}, \texttt{0x12}, \texttt{0x23}, \texttt{0x2f}, \texttt{0x2a}) \\
\mathbf{C}_{10} &= (\texttt{0xa4}, \texttt{0x6b}, \texttt{0x5d}, \texttt{0xa6}, \texttt{0x85}, \texttt{0x33}, \texttt{0x7f}, \texttt{0x5e}, \texttt{0x4b}, \texttt{0x77}, \texttt{0x47}, \texttt{0x32}, \texttt{0xa7}, \texttt{0x17}, \texttt{0x86}, \texttt{0x42}) \\
\mathbf{C}_{11} &= (\texttt{0x85}, \texttt{0x2c}, \texttt{0x90}, \texttt{0x2e}, \texttt{0xa7}, \texttt{0x3d}, \texttt{0x4f}, \texttt{0x55}, \texttt{0x91}, \texttt{0x46}, \texttt{0x77}, \texttt{0x53}, \texttt{0x56}, \texttt{0x05}, \texttt{0x04}, \texttt{0x32}) \\
\mathbf{C}_{12} &= (\texttt{0x7e}, \texttt{0xb3}, \texttt{0x92}, \texttt{0x91}, \texttt{0x33}, \texttt{0x05}, \texttt{0x27}, \texttt{0x36}, \texttt{0x35}, \texttt{0x27}, \texttt{0x9e}, \texttt{0x8b}, \texttt{0x5b}, \texttt{0x6e}, \texttt{0x47}, \texttt{0x84}) \\
\mathbf{C}_{13} &= (\texttt{0x90}, \texttt{0xca}, \texttt{0x4d}, \texttt{0x21}, \texttt{0x85}, \texttt{0x05}, \texttt{0xd8}, \texttt{0x47}, \texttt{0x98}, \texttt{0x85}, \texttt{0x06}, \texttt{0xbb}, \texttt{0x6b}, \texttt{0xd0}, \texttt{0x17}, \texttt{0x8f}) \\
\mathbf{C}_{14} &= (\texttt{0x94}, \texttt{0x44}, \texttt{0x3a}, \texttt{0x81}, \texttt{0x25}, \texttt{0x0e}, \texttt{0x96}, \texttt{0x0b}, \texttt{0x62}, \texttt{0x56}, \texttt{0x24}, \texttt{0x68}, \texttt{0x84}, \texttt{0x01}, \texttt{0x0d}, \texttt{0x06}) \\
\mathbf{C}_{15} &= (\texttt{0xa1}, \texttt{0x05}, \texttt{0x66}, \texttt{0x14}, \texttt{0x78}, \texttt{0xbd}, \texttt{0x52}, \texttt{0xdb}, \texttt{0x42}, \texttt{0x33}, \texttt{0xcd}, \texttt{0x9e}, \texttt{0x2d}, \texttt{0x15}, \texttt{0x8a}, \texttt{0xf1}) \\
\mathbf{C}_{16} &= (\texttt{0x40}, \texttt{0x21}, \texttt{0x05}, \texttt{0x5b}, \texttt{0x73}, \texttt{0x8c}, \texttt{0x5f}, \texttt{0x5f}, \texttt{0x57}, \texttt{0x09}, \texttt{0x36}, \texttt{0x98}, \texttt{0x31}, \texttt{0x31}, \texttt{0x73}, \texttt{0x05}) \\
\mathbf{C}_{17} &= (\texttt{0x2f}, \texttt{0x33}, \texttt{0x87}, \texttt{0x86}, \texttt{0x67}, \texttt{0x03}, \texttt{0x05}, \texttt{0x32}, \texttt{0x92}, \texttt{0xf8}, \texttt{0xbc}, \texttt{0x41}, \texttt{0x13}, \texttt{0x06}, \texttt{0x95}, \texttt{0x26}) \\
\mathbf{C}_{18} &= (\texttt{0x6c}, \texttt{0x04}, \texttt{0x9f}, \texttt{0xf0}, \texttt{0x9b}, \texttt{0x54}, \texttt{0x8a}, \texttt{0x18}, \texttt{0x06}, \texttt{0x05}, \texttt{0x66}, \texttt{0x85}, \texttt{0x47}, \texttt{0x65}, \texttt{0x68}, \texttt{0x68}) \\
\mathbf{C}_{19} &= (\texttt{0x51}, \texttt{0x9c}, \texttt{0x24}, \texttt{0x49}, \texttt{0x28}, \texttt{0x69}, \texttt{0x96}, \texttt{0x26}, \texttt{0x87}, \texttt{0x35}, \texttt{0x89}, \texttt{0x67}, \texttt{0x04}, \texttt{0x91}, \texttt{0x99}, \texttt{0xa0}) \\
\mathbf{C}_{20} &= (\texttt{0x92}, \texttt{0x82}, \texttt{0x11}, \texttt{0x87}, \texttt{0x1a}, \texttt{0x35}, \texttt{0x9e}, \texttt{0xd6}, \texttt{0x28}, \texttt{0x68}, \texttt{0x21}, \texttt{0x9b}, \texttt{0x92}, \texttt{0x05}, \texttt{0x67}, \texttt{0x8b}) \\
\mathbf{C}_{21} &= (\texttt{0x81}, \texttt{0xb3}, \texttt{0x5c}, \texttt{0x12}, \texttt{0x74}, \texttt{0x26}, \texttt{0x2e}, \texttt{0x93}, \texttt{0x6a}, \texttt{0x66}, \texttt{0x7f}, \texttt{0xdc}, \texttt{0x94}, \texttt{0x58}, \texttt{0x7d}, \texttt{0x13}) \\
\mathbf{C}_{22} &= (\texttt{0x1a}, \texttt{0x35}, \texttt{0x9e}, \texttt{0xd6}, \texttt{0x28}, \texttt{0x68}, \texttt{0x21}, \texttt{0x9b}, \texttt{0x92}, \texttt{0x05}, \texttt{0x67}, \texttt{0x8b}, \texttt{0x81}, \texttt{0xb3}, \texttt{0x5c}, \texttt{0x12}) \\
\mathbf{C}_{23} &= (\texttt{0x74}, \texttt{0x26}, \texttt{0x2e}, \texttt{0x93}, \texttt{0x6a}, \texttt{0x66}, \texttt{0x7f}, \texttt{0xdc}, \texttt{0x94}, \texttt{0x58}, \texttt{0x7d}, \texttt{0x13}, \texttt{0x75}, \texttt{0x61}, \texttt{0xba}, \texttt{0x13}) \\
\mathbf{C}_{24} &= (\texttt{0x99}, \texttt{0x34}, \texttt{0x96}, \texttt{0x88}, \texttt{0x23}, \texttt{0x9b}, \texttt{0x56}, \texttt{0x16}, \texttt{0x80}, \texttt{0x91}, \texttt{0x4e}, \texttt{0xfc}, \texttt{0x22}, \texttt{0x2d}, \texttt{0x3a}, \texttt{0x92}) \\
\mathbf{C}_{25} &= (\texttt{0x69}, \texttt{0x85}, \texttt{0x28}, \texttt{0x83}, \texttt{0x98}, \texttt{0x47}, \texttt{0x28}, \texttt{0x69}, \texttt{0x96}, \texttt{0x26}, \texttt{0x87}, \texttt{0x35}, \texttt{0x89}, \texttt{0x67}, \texttt{0x04}, \texttt{0x91}) \\
\mathbf{C}_{26} &= (\texttt{0x99}, \texttt{0xa0}, \texttt{0x92}, \texttt{0x82}, \texttt{0x11}, \texttt{0x87}, \texttt{0x1a}, \texttt{0x35}, \texttt{0x9e}, \texttt{0xd6}, \texttt{0x28}, \texttt{0x68}, \texttt{0x21}, \texttt{0x9b}, \texttt{0x92}, \texttt{0x05}) \\
\mathbf{C}_{27} &= (\texttt{0x67}, \texttt{0x8b}, \texttt{0x81}, \texttt{0xb3}, \texttt{0x5c}, \texttt{0x12}, \texttt{0x74}, \texttt{0x26}, \texttt{0x2e}, \texttt{0x93}, \texttt{0x6a}, \texttt{0x66}, \texttt{0x7f}, \texttt{0xdc}, \texttt{0x94}, \texttt{0x58}) \\
\mathbf{C}_{28} &= (\texttt{0x7d}, \texttt{0x13}, \texttt{0x75}, \texttt{0x61}, \texttt{0xba}, \texttt{0x13}, \texttt{0x74}, \texttt{0x26}, \texttt{0x2e}, \texttt{0x93}, \texttt{0x6a}, \texttt{0x66}, \texttt{0x7f}, \texttt{0xdc}, \texttt{0x94}, \texttt{0x58}) \\
\mathbf{C}_{29} &= (\texttt{0x7d}, \texttt{0x13}, \texttt{0x99}, \texttt{0x34}, \texttt{0x96}, \texttt{0x88}, \texttt{0x23}, \texttt{0x9b}, \texttt{0x56}, \texttt{0x16}, \texttt{0x80}, \texttt{0x91}, \texttt{0x4e}, \texttt{0xfc}, \texttt{0x22}, \texttt{0x2d}) \\
\mathbf{C}_{30} &= (\texttt{0x3a}, \texttt{0x92}, \texttt{0x69}, \texttt{0x85}, \texttt{0x28}, \texttt{0x83}, \texttt{0x98}, \texttt{0x47}, \texttt{0x28}, \texttt{0x69}, \texttt{0x96}, \texttt{0x26}, \texttt{0x87}, \texttt{0x35}, \texttt{0x89}, \texttt{0x67}) \\
\mathbf{C}_{31} &= (\texttt{0x04}, \texttt{0x91}, \texttt{0x99}, \texttt{0xa0}, \texttt{0x92}, \texttt{0x82}, \texttt{0x11}, \texttt{0x87}, \texttt{0x1a}, \texttt{0x35}, \texttt{0x9e}, \texttt{0xd6}, \texttt{0x28}, \texttt{0x68}, \texttt{0x21}, \texttt{0x9b}) \\
\mathbf{C}_{32} &= (\texttt{0x92}, \texttt{0x05}, \texttt{0x67}, \texttt{0x8b}, \texttt{0x81}, \texttt{0xb3}, \texttt{0x5c}, \texttt{0x12}, \texttt{0x74}, \texttt{0x26}, \texttt{0x2e}, \texttt{0x93}, \texttt{0x6a}, \texttt{0x66}, \texttt{0x7f}, \texttt{0xdc})
\end{aligned}$$

Key Schedule Algorithm

Algorithm 3 Kuznyechik Key Schedule

Require: 256-bit master key $K = \mathbf{K}_1 \parallel \mathbf{K}_0$, where $|\mathbf{K}_1| = |\mathbf{K}_0| = 128$ bits
Ensure: Round keys $K^{(0)}, K^{(1)}, \ldots, K^{(9)}$
$K^{(0)} \leftarrow \mathbf{K}_1$
$K^{(1)} \leftarrow \mathbf{K}_0$
$(\mathbf{a}_0, \mathbf{a}_1) \leftarrow (\mathbf{K}_1, \mathbf{K}_0)$
for $j = 1$ to 32 **do**
    $(\mathbf{a}_0, \mathbf{a}_1) \leftarrow F(\mathbf{a}_0, \mathbf{a}_1, \mathbf{C}_j)$ The F-function uses S-box, L-layer, and XOR with constant
    if $j \bmod 8 = 0$ **then**
        $K^{(j/4+1)} \leftarrow \mathbf{a}_1$
        $K^{(j/4+2)} \leftarrow \mathbf{a}_0$
    end if
end for

Encryption Algorithm

The encryption process applies 9 rounds of transformation followed by a final key addition:

Algorithm 4 Kuznyechik Encryption

Input: 128-bit plaintext $\mathbf{P}$, round keys $K^{(0)}, \ldots, K^{(9)}$
Output: 128-bit ciphertext $\mathbf{C}$
$\mathbf{X} \leftarrow \mathbf{P}$
for $i = 0$ to 8 **do**
    $\mathbf{X} \leftarrow \mathbf{X} \oplus K^{(i)}$ AddRoundKey
    $\mathbf{X} \leftarrow S(\mathbf{X})$ SubBytes (byte-wise application of S-box)
    $\mathbf{X} \leftarrow \mathbf{L}(\mathbf{X})$ Linear transformation
end for
$\mathbf{C} \leftarrow \mathbf{X} \oplus K^{(9)}$ Final key addition **return C**

For completeness, we include the decryption algorithm, though our analysis uses only encryption. Decryption applies the inverse operations in reverse order:

Algorithm 5 Kuznyechik Decryption

Input: 128-bit ciphertext $\mathbf{C}$, round keys $K^{(0)}, \ldots, K^{(9)}$
Output: 128-bit plaintext $\mathbf{P}$
$\mathbf{X} \leftarrow \mathbf{C}$
$\mathbf{X} \leftarrow \mathbf{X} \oplus K^{(9)}$ Remove final key
for $i = 8$ downto 0 **do**
    $\mathbf{X} \leftarrow \mathbf{L}^{-1}(\mathbf{X})$ Inverse linear transformation
    $\mathbf{X} \leftarrow S^{-1}(\mathbf{X})$ Inverse SubBytes (byte-wise application of S^{-1})
    $\mathbf{X} \leftarrow \mathbf{X} \oplus K^{(i)}$ Remove round key
end for
$\mathbf{P} \leftarrow \mathbf{X}$
return P

Security Analysis

Kuznyechik has been designed to resist various cryptanalytic attacks, with properties typically cited in the GOST R 34.12-2015 standard and subsequent cryptanalysis. These include:

- **Differential Cryptanalysis**: The S-box is designed with optimal differential properties, often cited with a maximum differential probability of 2^{-6} for single S-box operations [13].
- **Linear Cryptanalysis**: The linear approximation table is constructed to show a low maximum bias, often cited as 2^{-4} for single S-box operations [13].
- **Integral Attacks**: The branch number of the linear transformation provides resistance against integral distinguishers by ensuring rapid diffusion [**?**, 1].
- **Algebraic Attacks**: The high degree of the S-box polynomial contributes to resistance against low-degree algebraic relations [44].

The 9-round structure (for a 256-bit key) provides an adequate security margin against known attacks, with the complex interaction between the S-box and linear layer ensuring rapid diffusion and confusion. The cipher supports standard modes of operation (ECB, CBC, CFB, OFB, CTR) and can be implemented efficiently in both software and hardware environments.

B The (inner/outer) cDU Table for the S-box of Kuznyechik and its inverse

Recall the complementary property of the inner cDU of an S-box with the outer cDU of its inverse. For the purpose of completion we include both here.

Table 5: Inner c-Differential Uniformity δ_c of Kuznyechik S-box

c	Hex	δ_c	c	Hex	δ_c	c	Hex	δ_c	c	Hex	δ_c	c	Hex	δ_c
1	0x01	8	52	0x34	11	103	0x67	8	154	0x9a	7	205	0xcd	7
2	0x02	64	53	0x35	13	104	0x68	8	155	0x9b	7	206	0xce	8
3	0x03	21	54	0x36	8	105	0x69	8	156	0x9c	8	207	0xcf	7
4	0x04	33	55	0x37	7	106	0x6a	9	157	0x9d	12	208	0xd0	7
5	0x05	9	56	0x38	8	107	0x6b	9	158	0x9e	11	209	0xd1	8
6	0x06	10	57	0x39	8	108	0x6c	7	159	0x9f	8	210	0xd2	7
7	0x07	8	58	0x3a	9	109	0x6d	6	160	0xa0	7	211	0xd3	7
8	0x08	13	59	0x3b	11	110	0x6e	6	161	0xa1	7	212	0xd4	8
9	0x09	8	60	0x3c	8	111	0x6f	13	162	0xa2	7	213	0xd5	8
10	0x0a	13	61	0x3d	7	112	0x70	10	163	0xa3	7	214	0xd6	8
11	0x0b	11	62	0x3e	14	113	0x71	9	164	0xa4	8	215	0xd7	7
12	0x0c	8	63	0x3f	14	114	0x72	9	165	0xa5	9	216	0xd8	8
13	0x0d	7	64	0x40	8	115	0x73	8	166	0xa6	9	217	0xd9	8
14	0x0e	7	65	0x41	9	116	0x74	9	167	0xa7	8	218	0xda	7
15	0x0f	8	66	0x42	7	117	0x75	12	168	0xa8	8	219	0xdb	8
16	0x10	11	67	0x43	7	118	0x76	9	169	0xa9	13	220	0xdc	7
17	0x11	7	68	0x44	7	119	0x77	8	170	0xaa	7	221	0xdd	7
18	0x12	8	69	0x45	7	120	0x78	9	171	0xab	8	222	0xde	8
19	0x13	7	70	0x46	7	121	0x79	9	172	0xac	7	223	0xdf	8
20	0x14	9	71	0x47	8	122	0x7a	7	173	0xad	7	224	0xe0	13
21	0x15	9	72	0x48	8	123	0x7b	8	174	0xae	8	225	0xe1	64
22	0x16	7	73	0x49	7	124	0x7c	8	175	0xaf	9	226	0xe2	7
23	0x17	8	74	0x4a	8	125	0x7d	7	176	0xb0	8	227	0xe3	8
24	0x18	8	75	0x4b	8	126	0x7e	8	177	0xb1	8	228	0xe4	8
25	0x19	7	76	0x4c	8	127	0x7f	8	178	0xb2	7	229	0xe5	7
26	0x1a	7	77	0x4d	8	128	0x80	9	179	0xb3	8	230	0xe6	8
27	0x1b	8	78	0x4e	8	129	0x81	8	180	0xb4	7	231	0xe7	9
28	0x1c	8	79	0x4f	7	130	0x82	8	181	0xb5	11	232	0xe8	8
29	0x1d	12	80	0x50	8	131	0x83	8	182	0xb6	8	233	0xe9	8
30	0x1e	7	81	0x51	8	132	0x84	7	183	0xb7	7	234	0xea	9
31	0x1f	8	82	0x52	7	133	0x85	8	184	0xb8	9	235	0xeb	12
32	0x20	9	83	0x53	7	134	0x86	8	185	0xb9	8	236	0xec	12
33	0x21	8	84	0x54	7	135	0x87	7	186	0xba	8	237	0xed	8
34	0x22	8	85	0x55	11	136	0x88	8	187	0xbb	9	238	0xee	7
35	0x23	12	86	0x56	8	137	0x89	8	188	0xbc	8	239	0xef	7

36	0x24	13	87	0x57	9	138	0x8a	8	189	0xbd	10	240	0xf0	9
37	0x25	7	88	0x58	7	139	0x8b	7	190	0xbe	21	241	0xf1	8
38	0x26	8	89	0x59	7	140	0x8c	12	191	0xbf	13	242	0xf2	8
39	0x27	7	90	0x5a	7	141	0x8d	9	192	0xc0	8	243	0xf3	8
40	0x28	8	91	0x5b	9	142	0x8e	14	193	0xc1	7	244	0xf4	8
41	0x29	9	92	0x5c	12	143	0x8f	10	194	0xc2	7	245	0xf5	8
42	0x2a	11	93	0x5d	9	144	0x90	9	195	0xc3	9	246	0xf6	7
43	0x2b	8	94	0x5e	9	145	0x91	33	196	0xc4	12	247	0xf7	7
44	0x2c	8	95	0x5f	10	146	0x92	10	197	0xc5	9	248	0xf8	7
45	0x2d	7	96	0x60	8	147	0x93	14	198	0xc6	8	249	0xf9	8
46	0x2e	8	97	0x61	7	148	0x94	8	199	0xc7	9	250	0xfa	8
47	0x2f	9	98	0x62	9	149	0x95	7	200	0xc8	7	251	0xfb	9
48	0x30	7	99	0x63	8	150	0x96	12	201	0xc9	7	252	0xfc	7
49	0x31	8	100	0x64	11	151	0x97	8	202	0xca	8	253	0xfd	8
50	0x32	8	101	0x65	11	152	0x98	8	203	0xcb	8	254	0xfe	7
51	0x33	7	102	0x66	8	153	0x99	8	204	0xcc	8	255	0xff	8

Table 6: Inner c-Differential Uniformity δ_c^{-1} of Kuznyechik Inverse S-box

c	Hex	δ_c^{-1}	c	Hex	δ_c^{-1}	c	Hex	δ_c^{-1}	c	Hex	δ_c^{-1}	c	Hex	δ_c^{-1}
1	0x01	8	52	0x34	8	103	0x67	7	154	0x9a	7	205	0xcd	8
2	0x02	9	53	0x35	8	104	0x68	9	155	0x9b	7	206	0xce	7
3	0x03	8	54	0x36	8	105	0x69	7	156	0x9c	8	207	0xcf	7
4	0x04	7	55	0x37	7	106	0x6a	7	157	0x9d	7	208	0xd0	7
5	0x05	7	56	0x38	8	107	0x6b	7	158	0x9e	8	209	0xd1	8
6	0x06	7	57	0x39	8	108	0x6c	7	159	0x9f	8	210	0xd2	7
7	0x07	7	58	0x3a	8	109	0x6d	7	160	0xa0	7	211	0xd3	8
8	0x08	8	59	0x3b	8	110	0x6e	7	161	0xa1	8	212	0xd4	8
9	0x09	7	60	0x3c	7	111	0x6f	8	162	0xa2	8	213	0xd5	8
10	0x0a	8	61	0x3d	7	112	0x70	8	163	0xa3	7	214	0xd6	7
11	0x0b	8	62	0x3e	7	113	0x71	7	164	0xa4	7	215	0xd7	8
12	0x0c	7	63	0x3f	7	114	0x72	8	165	0xa5	8	216	0xd8	8
13	0x0d	8	64	0x40	9	115	0x73	8	166	0xa6	7	217	0xd9	7
14	0x0e	8	65	0x41	7	116	0x74	7	167	0xa7	7	218	0xda	7
15	0x0f	8	66	0x42	8	117	0x75	7	168	0xa8	7	219	0xdb	7
16	0x10	8	67	0x43	8	118	0x76	7	169	0xa9	8	220	0xdc	8
17	0x11	7	68	0x44	8	119	0x77	7	170	0xaa	7	221	0xdd	8
18	0x12	7	69	0x45	7	120	0x78	7	171	0xab	8	222	0xde	7
19	0x13	7	70	0x46	8	121	0x79	7	172	0xac	8	223	0xdf	7
20	0x14	8	71	0x47	7	122	0x7a	8	173	0xad	8	224	0xe0	7
21	0x15	7	72	0x48	7	123	0x7b	8	174	0xae	8	225	0xe1	9
22	0x16	8	73	0x49	8	124	0x7c	7	175	0xaf	8	226	0xe2	7
23	0x17	9	74	0x4a	8	125	0x7d	8	176	0xb0	8	227	0xe3	8
24	0x18	7	75	0x4b	7	126	0x7e	7	177	0xb1	8	228	0xe4	8
25	0x19	7	76	0x4c	7	127	0x7f	7	178	0xb2	7	229	0xe5	7
26	0x1a	8	77	0x4d	7	128	0x80	9	179	0xb3	8	230	0xe6	7
27	0x1b	9	78	0x4e	7	129	0x81	7	180	0xb4	8	231	0xe7	7
28	0x1c	8	79	0x4f	7	130	0x82	7	181	0xb5	8	232	0xe8	7
29	0x1d	8	80	0x50	7	131	0x83	8	182	0xb6	8	233	0xe9	7
30	0x1e	7	81	0x51	8	132	0x84	7	183	0xb7	7	234	0xea	7
31	0x1f	7	82	0x52	7	133	0x85	7	184	0xb8	8	235	0xeb	7
32	0x20	7	83	0x53	8	134	0x86	7	185	0xb9	8	236	0xec	7
33	0x21	7	84	0x54	8	135	0x87	8	186	0xba	7	237	0xed	8
34	0x22	8	85	0x55	8	136	0x88	7	187	0xbb	7	238	0xee	8
35	0x23	8	86	0x56	7	137	0x89	8	188	0xbc	9	239	0xef	8
36	0x24	8	87	0x57	8	138	0x8a	8	189	0xbd	8	240	0xf0	8
37	0x25	7	88	0x58	8	139	0x8b	8	190	0xbe	8	241	0xf1	8
38	0x26	8	89	0x59	8	140	0x8c	7	191	0xbf	7	242	0xf2	8
39	0x27	7	90	0x5a	8	141	0x8d	8	192	0xc0	8	243	0xf3	8
40	0x28	8	91	0x5b	8	142	0x8e	7	193	0xc1	7	244	0xf4	7
41	0x29	8	92	0x5c	7	143	0x8f	8	194	0xc2	9	245	0xf5	7
42	0x2a	8	93	0x5d	8	144	0x90	7	195	0xc3	7	246	0xf6	7
43	0x2b	8	94	0x5e	9	145	0x91	7	196	0xc4	7	247	0xf7	8
44	0x2c	7	95	0x5f	7	146	0x92	8	197	0xc5	7	248	0xf8	9

45	0x2d	8	96	0x60	7	147	0x93	7	198	0xc6	7	249	0xf9	7
46	0x2e	7	97	0x61	7	148	0x94	9	199	0xc7	8	250	0xfa	7
47	0x2f	7	98	0x62	7	149	0x95	7	200	0xc8	8	251	0xfb	8
48	0x30	7	99	0x63	8	150	0x96	7	201	0xc9	7	252	0xfc	8
49	0x31	9	100	0x64	8	151	0x97	7	202	0xca	8	253	0xfd	8
50	0x32	9	101	0x65	8	152	0x98	8	203	0xcb	8	254	0xfe	7
51	0x33	7	102	0x66	8	153	0x99	7	204	0xcc	8	255	0xff	8

C Detailed cryptanalysis results by round

The following tables present the 10–15 most significant results from the summary of each cryptanalysis run. The marker '‡ ' denotes a result that is significant after False Discovery Rate (FDR) correction (FDR. Sig. is 1 when the outcome was significant after FDR correction, and 0 if not). The marker '† ' denotes a noteworthy result based on combined evidence (Comb. Ev. – Fisher's method), even when no single differential was FDR-significant.

Results for Seed 37

Table 7: Top Noteworthy Findings for 9-Round Kuznyechik (Seed 37)

Marker	c	Configuration	Max Bias	FDR Sig.	Min FDR P-val
‡	0xe1	byte_4_in→byte_4_out	1.7x	1	9.25e-03
‡	0x91	byte_0_in→byte_1_out	1.5x	1	5.93e-02
‡	0x91	byte_12_in→byte_12_out	1.5x	1	5.94e-02
‡	0xe1	byte_12_in→byte_12_out	1.5x	1	5.94e-02
†	0x03	byte_4_in→byte_4_out	1.6x	0	1.16e-01
†	0x03	byte_6_in→byte_6_out	1.6x	0	1.22e-01
†	0x01	byte_12_in→byte_12_out	3.1x	0	2.36e-01
†	0xe1	byte_4_in→byte_5_out	1.6x	0	2.40e-01
†	0x01	byte_0_in→byte_0_out	3.0x	0	2.51e-01
†	0x91	byte_0_in→byte_0_out	1.6x	0	2.96e-01
†	0xbe	byte_14_in→byte_15_out	1.5x	0	2.97e-01
†	0xbe	byte_14_in→byte_14_out	1.5x	0	3.79e-01
†	0x02	byte_14_in→byte_15_out	1.5x	0	4.58e-01
†	0x02	byte_8_in→byte_8_out	1.5x	0	4.79e-01
†	0xe1	byte_4_in→byte_4_out	1.5x	0	5.95e-01
†	0x02	byte_12_in→byte_12_out	1.5x	0	5.96e-01
†	0x04	byte_4_in→byte_4_out	1.5x	0	6.16e-01
†	0x91	byte_8_in→byte_8_out	1.5x	0	7.14e-01
†	0x04	byte_8_in→byte_8_out	1.5x	0	7.59e-01
†	0x03	byte_12_in→byte_12_out	1.5x	0	8.28e-01

Table 8: Top Noteworthy Findings for 8-Round Kuznyechik (Seed 37)

Marker	c	Configuration	Max Bias	FDR Sig.	Min FDR P-val
†	0x91	byte_2_in→byte_2_out	1.5x	0	2.82e-02
†	0xe1	byte_4_in→byte_4_out	1.6x	0	4.32e-02
†	0xbe	byte_14_in→byte_15_out	1.5x	0	1.22e-01
†	0x04	byte_2_in→byte_2_out	1.6x	0	1.60e-01
†	0x91	byte_12_in→byte_12_out	1.6x	0	1.83e-01
†	0x02	byte_14_in→byte_15_out	1.6x	0	1.87e-01

Table 8 – continued from previous page

Marker	c	Configuration	Max Bias	FDR Sig.	Min FDR P-val
†	0xbe	byte_14_in→byte_14_out	1.6x	0	1.87e-01
†	0x04	byte_4_in→byte_4_out	1.5x	0	2.45e-01
†	0x03	byte_0_in→byte_0_out	1.5x	0	2.45e-01
†	0x04	byte_8_in→byte_8_out	1.5x	0	2.45e-01
†	0x02	byte_0_in→byte_1_out	1.6x	0	3.01e-01
†	0x91	byte_4_in→byte_4_out	1.6x	0	3.05e-01
†	0x03	byte_12_in→byte_12_out	1.5x	0	4.63e-01
†	0x03	byte_10_in→byte_10_out	1.5x	0	4.78e-01
†	0xe1	byte_6_in→byte_6_out	1.6x	0	4.79e-01
†	0xbe	byte_2_in→byte_2_out	1.5x	0	5.93e-01
†	0xe1	byte_12_in→byte_12_out	1.5x	0	6.22e-01
†	0xbe	byte_6_in→byte_6_out	1.5x	0	7.18e-01
†	0xbe	byte_0_in→byte_0_out	1.5x	0	7.51e-01
†	0xe1	byte_4_in→byte_5_out	1.5x	0	7.58e-01

Table 9: Top Noteworthy Findings for 7-Round Kuznyechik (Seed 37)

Marker	c	Configuration	Max Bias	FDR Sig.	Min FDR P-val
†	0x03	byte_10_in→byte_10_out	1.6x	0	1.29e-02
†	0x04	byte_8_in→byte_8_out	1.6x	0	2.60e-02
†	0xe1	byte_4_in→byte_5_out	1.6x	0	5.93e-02
†	0xbe	byte_6_in→byte_6_out	1.6x	0	7.10e-02
†	0x02	byte_8_in→byte_8_out	1.6x	0	2.40e-01
†	0x02	byte_12_in→byte_12_out	1.6x	0	3.01e-01
†	0x03	byte_12_in→byte_12_out	1.6x	0	3.01e-01
†	0x03	byte_0_in→byte_0_out	1.6x	0	4.79e-01
†	0xbe	byte_2_in→byte_2_out	1.5x	0	4.79e-01
†	0xbe	byte_14_in→byte_15_out	1.5x	0	4.79e-01
†	0xe1	byte_8_in→byte_8_out	1.5x	0	5.94e-01
†	0x91	byte_0_in→byte_1_out	1.6x	0	7.12e-01
†	0x01	byte_6_in→byte_6_out	3.0x	0	7.12e-01
†	0x01	byte_0_in→byte_0_out	3.0x	0	7.12e-01
†	0xbe	byte_0_in→byte_0_out	1.5x	0	7.14e-01
†	0x91	byte_0_in→byte_0_out	1.5x	0	7.28e-01
†	0x04	byte_4_in→byte_4_out	1.5x	0	7.57e-01
†	0x04	byte_2_in→byte_2_out	1.5x	0	7.78e-01
†	0x02	byte_14_in→byte_15_out	1.5x	0	7.92e-01
†	0x03	byte_6_in→byte_6_out	1.5x	0	7.98e-01

Table 10: Top Noteworthy Findings for 6-Round Kuznyechik (Seed 37)

Marker	c	Configuration	Max Bias	FDR Sig.	Min FDR P-val
†	0xe1	byte_6_in→byte_6_out	1.5x	0	2.82e-02
†	0x03	byte_4_in→byte_4_out	1.6x	0	4.31e-02
†	0x02	byte_12_in→byte_12_out	1.5x	0	1.22e-01
†	0x91	byte_8_in→byte_8_out	1.6x	0	1.51e-01
†	0x91	byte_0_in→byte_1_out	1.5x	0	2.45e-01

Table 10 – continued from previous page

Marker	c	Configuration	Max Bias	FDR Sig.	Min FDR P-val
†	0xbe	byte_0_in→byte_0_out	1.5x	0	2.45e-01
†	0xbe	byte_2_in→byte_2_out	1.5x	0	2.45e-01
†	0x91	byte_0_in→byte_0_out	1.5x	0	3.79e-01
†	0xbe	byte_6_in→byte_6_out	1.5x	0	3.96e-01
†	0x04	byte_2_in→byte_2_out	1.5x	0	4.27e-01
†	0x91	byte_12_in→byte_12_out	1.5x	0	4.57e-01
†	0x02	byte_14_in→byte_15_out	1.6x	0	4.79e-01
†	0xbe	byte_14_in→byte_15_out	1.5x	0	5.19e-01
†	0x91	byte_4_in→byte_4_out	1.5x	0	5.94e-01
†	0xe1	byte_4_in→byte_5_out	1.5x	0	6.16e-01
†	0xe1	byte_8_in→byte_8_out	1.5x	0	6.16e-01
†	0xbe	byte_14_in→byte_14_out	1.5x	0	6.16e-01
†	0x04	byte_8_in→byte_8_out	1.5x	0	6.16e-01
†	0x03	byte_10_in→byte_10_out	1.5x	0	7.67e-01
†	0x02	byte_2_in→byte_2_out	1.5x	0	7.77e-01

Table 11: Top Noteworthy Findings for 5-Round Kuznyechik (Seed 37)

Marker	c	Configuration	Max Bias	FDR Sig.	Min FDR P-val
†	0x04	byte_4_in→byte_4_out	1.6x	0	1.16e-01
†	0x03	byte_6_in→byte_6_out	1.5x	0	1.22e-01
†	0xe1	byte_8_in→byte_8_out	1.5x	0	1.22e-01
†	0x03	byte_10_in→byte_10_out	1.6x	0	1.51e-01
†	0xbe	byte_14_in→byte_14_out	1.6x	0	1.87e-01
†	0x91	byte_4_in→byte_4_out	1.5x	0	2.40e-01
†	0x02	byte_8_in→byte_8_out	1.5x	0	2.45e-01
†	0x02	byte_12_in→byte_12_out	1.6x	0	2.97e-01
†	0x01	byte_2_in→byte_2_out	3.1x	0	3.56e-01
†	0x01	byte_6_in→byte_6_out	3.0x	0	3.56e-01
†	0x91	byte_12_in→byte_12_out	1.5x	0	4.27e-01
†	0x02	byte_0_in→byte_1_out	1.5x	0	4.44e-01
†	0x04	byte_8_in→byte_8_out	1.5x	0	4.58e-01
†	0xbe	byte_6_in→byte_6_out	1.6x	0	4.78e-01
†	0x01	byte_12_in→byte_12_out	3.1x	0	4.79e-01
†	0x01	byte_8_in→byte_8_out	3.1x	0	4.79e-01
†	0x91	byte_2_in→byte_2_out	1.5x	0	4.79e-01
†	0x02	byte_14_in→byte_15_out	1.5x	0	5.93e-01
†	0xbe	byte_14_in→byte_15_out	1.5x	0	5.94e-01
†	0x02	byte_2_in→byte_2_out	1.5x	0	5.94e-01

Table 12: Top Noteworthy Findings for 4-Round Kuznyechik (Seed 37)

Marker	c	Configuration	Max Bias	FDR Sig.	Min FDR P-val
†	0xe1	byte_4_in→byte_4_out	1.7x	0	9.25e-03
†	0x91	byte_0_in→byte_1_out	1.6x	0	2.59e-02
†	0x03	byte_0_in→byte_0_out	1.5x	0	5.94e-02
†	0x03	byte_4_in→byte_4_out	1.6x	0	1.16e-01

Table 12 – continued from previous page

Marker	c	Configuration	Max Bias	FDR Sig.	Min FDR P-val
†	0x04	byte_8_in→byte_8_out	1.6x	0	1.16e-01
†	0x02	byte_12_in→byte_12_out	1.5x	0	1.22e-01
†	0x01	byte_2_in→byte_2_out	3.1x	0	2.24e-01
†	0x91	byte_12_in→byte_12_out	1.5x	0	2.39e-01
†	0x02	byte_2_in→byte_2_out	1.6x	0	2.40e-01
†	0xe1	byte_8_in→byte_8_out	1.5x	0	2.45e-01
†	0x02	byte_14_in→byte_15_out	1.6x	0	3.01e-01
†	0x91	byte_8_in→byte_8_out	1.6x	0	3.01e-01
†	0xe1	byte_6_in→byte_6_out	1.6x	0	3.01e-01
†	0xbe	byte_0_in→byte_0_out	1.5x	0	3.41e-01
†	0x04	byte_2_in→byte_2_out	1.6x	0	4.57e-01
†	0xbe	byte_6_in→byte_6_out	1.6x	0	4.78e-01
†	0x03	byte_6_in→byte_6_out	1.6x	0	4.79e-01
†	0xbe	byte_14_in→byte_15_out	1.5x	0	4.79e-01
†	0x04	byte_4_in→byte_4_out	1.5x	0	4.80e-01
†	0x91	byte_2_in→byte_2_out	1.5x	0	5.25e-01

Table 13: Top Noteworthy Findings for 3-Round Kuznyechik (Seed 37)

Marker	c	Configuration	Max Bias	FDR Sig.	Min FDR P-val
‡	0x01	byte_0_in→byte_0_out	9.8x	10883	0.00e+00
‡	0x01	byte_12_in→byte_12_out	10.1x	11006	0.00e+00
‡	0x01	byte_2_in→byte_2_out	10.7x	11047	0.00e+00
‡	0x01	byte_2_in→byte_3_out	10.3x	10984	0.00e+00
‡	0x01	byte_6_in→byte_6_out	10.6x	10958	0.00e+00
‡	0x01	byte_8_in→byte_8_out	9.9x	10920	0.00e+00
‡	0x03	byte_4_in→byte_4_out	1.5x	1	1.07e-03
†	0xbe	byte_14_in→byte_14_out	1.6x	0	2.60e-02
†	0xe1	byte_12_in→byte_12_out	1.5x	0	2.81e-02
†	0x01	byte_8_in→byte_8_out	3.3x	0	6.73e-02
†	0xbe	byte_6_in→byte_6_out	1.6x	0	7.09e-02
†	0x02	byte_2_in→byte_2_out	1.5x	0	1.22e-01
†	0xe1	byte_2_in→byte_2_out	1.6x	0	1.87e-01
†	0x04	byte_8_in→byte_8_out	1.6x	0	2.40e-01
†	0x91	byte_8_in→byte_8_out	1.6x	0	2.40e-01
†	0x03	byte_10_in→byte_10_out	1.5x	0	2.45e-01
†	0x03	byte_12_in→byte_12_out	1.6x	0	3.01e-01
†	0x03	byte_0_in→byte_0_out	1.5x	0	3.78e-01
†	0xe1	byte_4_in→byte_5_out	1.6x	0	4.58e-01
†	0x02	byte_8_in→byte_8_out	1.5x	0	4.58e-01

Table 14: Top Noteworthy Findings for 2-Round Kuznyechik (Seed 37)

Marker	c	Configuration	Max Bias	FDR Sig.	Min FDR P-val
†	0x91	byte_8_in→byte_8_out	1.7x	0	1.56e-02
†	0x03	byte_12_in→byte_12_out	1.6x	0	5.94e-02
†	0x04	byte_4_in→byte_4_out	1.6x	0	1.22e-01

Table 14 – continued from previous page

Marker	c	Configuration	Max Bias	FDR Sig.	Min FDR P-val
†	0x03	byte_10_in→byte_10_out	1.6x	0	1.87e-01
†	0xe1	byte_4_in→byte_5_out	1.5x	0	2.45e-01
†	0xbe	byte_14_in→byte_14_out	1.6x	0	4.44e-01
†	0xbe	byte_6_in→byte_6_out	1.5x	0	4.57e-01
†	0xe1	byte_2_in→byte_2_out	1.5x	0	4.79e-01
†	0x91	byte_0_in→byte_1_out	1.6x	0	4.79e-01
†	0xbe	byte_14_in→byte_15_out	1.5x	0	5.69e-01
†	0x02	byte_12_in→byte_12_out	1.5x	0	5.87e-01
†	0x02	byte_8_in→byte_8_out	1.5x	0	5.93e-01
†	0xe1	byte_8_in→byte_8_out	1.5x	0	7.18e-01
†	0x01	byte_2_in→byte_2_out	2.9x	0	7.52e-01
†	0x01	byte_6_in→byte_6_out	2.9x	0	7.52e-01
†	0xbe	byte_2_in→byte_2_out	1.5x	0	7.78e-01
†	0x02	byte_14_in→byte_15_out	1.5x	0	7.78e-01
†	0x03	byte_0_in→byte_0_out	1.5x	0	9.08e-01
†	0x02	byte_2_in→byte_2_out	1.5x	0	9.09e-01
†	0x91	byte_0_in→byte_0_out	1.5x	0	9.25e-01

Table 15: Top Noteworthy Findings for 1-Round Kuznyechik (Seed 37)

Marker	c	Configuration	Max Bias	FDR Sig.	Min FDR P-val
‡	0x01	byte_0_in→byte_0_out	9.5x	10909	0.00e+00
‡	0x01	byte_12_in→byte_12_out	10.2x	10990	0.00e+00
‡	0x01	byte_2_in→byte_2_out	10.6x	10989	0.00e+00
‡	0x01	byte_2_in→byte_3_out	10.2x	10923	0.00e+00
‡	0x01	byte_6_in→byte_6_out	10.7x	11038	0.00e+00
‡	0x01	byte_8_in→byte_8_out	9.7x	10941	0.00e+00
†	0x91	byte_8_in→byte_8_out	1.7x	0	1.56e-02
†	0x03	byte_12_in→byte_12_out	1.6x	0	5.94e-02
†	0x04	byte_4_in→byte_4_out	1.6x	0	1.22e-01
†	0x03	byte_10_in→byte_10_out	1.6x	0	1.87e-01
†	0xe1	byte_4_in→byte_5_out	1.5x	0	2.45e-01
†	0xbe	byte_14_in→byte_14_out	1.6x	0	4.44e-01
†	0xbe	byte_6_in→byte_6_out	1.5x	0	4.57e-01
†	0xe1	byte_2_in→byte_2_out	1.5x	0	4.79e-01
†	0x91	byte_0_in→byte_1_out	1.6x	0	4.79e-01
†	0xbe	byte_14_in→byte_15_out	1.5x	0	5.69e-01
†	0x02	byte_12_in→byte_12_out	1.5x	0	5.87e-01
†	0x02	byte_8_in→byte_8_out	1.5x	0	5.93e-01
†	0xe1	byte_8_in→byte_8_out	1.5x	0	7.18e-01
†	0x01	byte_2_in→byte_2_out	2.9x	0	7.52e-01

Results for Seed 42

Table 16: Top Noteworthy Findings for 9-Round Kuznyechik (Seed 42)

Marker	c	Configuration	Max Bias	FDR Sig.	Min FDR P-val
‡	0x04	byte_8_in→byte_8_out	1.7x	1	1.85e-03
‡	0xe1	byte_6_in→byte_6_out	1.7x	1	9.24e-03
†	0x02	byte_12_in→byte_12_out	1.6x	0	7.08e-02
†	0x02	byte_0_in→byte_1_out	1.6x	0	1.22e-01
†	0xbe	byte_14_in→byte_14_out	1.6x	0	1.51e-01
†	0x03	byte_12_in→byte_12_out	1.6x	0	1.87e-01
†	0xbe	byte_2_in→byte_2_out	1.5x	0	1.90e-01
†	0x03	byte_6_in→byte_6_out	1.5x	0	2.40e-01
†	0xe1	byte_8_in→byte_8_out	1.6x	0	2.40e-01
†	0xe1	byte_4_in→byte_4_out	1.5x	0	2.45e-01
†	0xbe	byte_14_in→byte_15_out	1.5x	0	3.79e-01
†	0x91	byte_12_in→byte_12_out	1.5x	0	4.58e-01
†	0x04	byte_4_in→byte_4_out	1.5x	0	4.63e-01
†	0x02	byte_2_in→byte_2_out	1.5x	0	4.63e-01
†	0x04	byte_2_in→byte_2_out	1.6x	0	4.79e-01
†	0x01	byte_2_in→byte_3_out	3.0x	0	7.12e-01
†	0x01	byte_0_in→byte_0_out	3.0x	0	7.12e-01
†	0x01	byte_6_in→byte_6_out	2.9x	0	7.12e-01
†	0x01	byte_12_in→byte_12_out	2.9x	0	7.12e-01
†	0x01	byte_2_in→byte_2_out	2.9x	0	7.12e-01

Table 17: Top Noteworthy Findings for 8-Round Kuznyechik (Seed 42)

Marker	c	Configuration	Max Bias	FDR Sig.	Min FDR P-val
‡	0x03	byte_12_in→byte_12_out	1.6x	1	5.81e-03
†	0x03	byte_4_in→byte_4_out	1.6x	0	4.31e-02
†	0x91	byte_2_in→byte_2_out	1.5x	0	5.93e-02
†	0xe1	byte_6_in→byte_6_out	1.6x	0	7.09e-02
†	0xe1	byte_4_in→byte_4_out	1.5x	0	1.22e-01
†	0x04	byte_8_in→byte_8_out	1.5x	0	2.40e-01
†	0x04	byte_4_in→byte_4_out	1.5x	0	2.45e-01
†	0x02	byte_0_in→byte_1_out	1.5x	0	2.45e-01
†	0x03	byte_6_in→byte_6_out	1.5x	0	4.58e-01
†	0xbe	byte_14_in→byte_14_out	1.6x	0	4.80e-01
†	0x91	byte_0_in→byte_1_out	1.5x	0	5.87e-01
†	0xbe	byte_14_in→byte_15_out	1.5x	0	5.95e-01
†	0x01	byte_0_in→byte_0_out	2.9x	0	1.00e+00
†	0x01	byte_12_in→byte_12_out	2.9x	0	1.00e+00
†	0x01	byte_2_in→byte_2_out	2.9x	0	1.00e+00
†	0x01	byte_2_in→byte_3_out	2.9x	0	1.00e+00
†	0x01	byte_6_in→byte_6_out	2.7x	0	1.00e+00
†	0x01	byte_8_in→byte_8_out	2.9x	0	1.00e+00
†	0x02	byte_12_in→byte_12_out	1.5x	0	1.00e+00
†	0x02	byte_14_in→byte_15_out	1.5x	0	1.00e+00

Table 18: Top Noteworthy Findings for 7-Round Kuznyechik (Seed 42)

Marker	c	Configuration	Max Bias	FDR Sig.	Min FDR P-val
†	0x03	byte_6_in→byte_6_out	1.6x	0	1.16e-01
†	0x91	byte_2_in→byte_2_out	1.6x	0	1.16e-01
†	0x02	byte_8_in→byte_8_out	1.5x	0	1.22e-01
†	0xe1	byte_4_in→byte_5_out	1.6x	0	1.50e-01
†	0xbe	byte_2_in→byte_2_out	1.6x	0	1.87e-01
†	0xe1	byte_4_in→byte_4_out	1.6x	0	2.40e-01
†	0xbe	byte_0_in→byte_0_out	1.5x	0	2.45e-01
†	0x04	byte_4_in→byte_4_out	1.5x	0	2.45e-01
†	0x02	byte_0_in→byte_1_out	1.6x	0	3.00e-01
†	0xbe	byte_6_in→byte_6_out	1.6x	0	3.01e-01
†	0x01	byte_0_in→byte_0_out	2.9x	0	1.00e+00
†	0x01	byte_12_in→byte_12_out	2.7x	0	1.00e+00
†	0x01	byte_2_in→byte_2_out	2.7x	0	1.00e+00
†	0x01	byte_2_in→byte_3_out	2.9x	0	1.00e+00
†	0x01	byte_6_in→byte_6_out	2.6x	0	1.00e+00
†	0x01	byte_8_in→byte_8_out	2.7x	0	1.00e+00
†	0x02	byte_12_in→byte_12_out	1.5x	0	1.00e+00
†	0x02	byte_14_in→byte_15_out	1.5x	0	1.00e+00
†	0x02	byte_2_in→byte_2_out	1.5x	0	1.00e+00
†	0x03	byte_0_in→byte_0_out	1.5x	0	1.00e+00

Table 19: Top Noteworthy Findings for 6-Round Kuznyechik (Seed 42)

Marker	c	Configuration	Max Bias	FDR Sig.	Min FDR P-val
†	0x02	byte_14_in→byte_15_out	1.6x	0	4.31e-02
†	0xbe	byte_2_in→byte_2_out	1.6x	0	1.16e-01
†	0xe1	byte_6_in→byte_6_out	1.5x	0	1.22e-01
†	0xe1	byte_4_in→byte_5_out	1.5x	0	1.22e-01
†	0x02	byte_12_in→byte_12_out	1.6x	0	2.97e-01
†	0x02	byte_2_in→byte_2_out	1.6x	0	3.00e-01
†	0x02	byte_8_in→byte_8_out	1.6x	0	3.01e-01
†	0x91	byte_0_in→byte_1_out	1.6x	0	3.01e-01
†	0x01	byte_2_in→byte_3_out	3.0x	0	3.56e-01
†	0x03	byte_12_in→byte_12_out	1.6x	0	3.70e-01
†	0x01	byte_0_in→byte_0_out	2.9x	0	1.00e+00
†	0x01	byte_12_in→byte_12_out	2.7x	0	1.00e+00
†	0x01	byte_2_in→byte_2_out	2.9x	0	1.00e+00
†	0x01	byte_6_in→byte_6_out	2.7x	0	1.00e+00
†	0x01	byte_8_in→byte_8_out	2.7x	0	1.00e+00
†	0x02	byte_0_in→byte_1_out	1.5x	0	1.00e+00
†	0x03	byte_0_in→byte_0_out	1.5x	0	1.00e+00
†	0x03	byte_10_in→byte_10_out	1.5x	0	1.00e+00
†	0x03	byte_4_in→byte_4_out	1.5x	0	1.00e+00
†	0x03	byte_6_in→byte_6_out	1.5x	0	1.00e+00

Table 20: Top Noteworthy Findings for 5-Round Kuznyechik (Seed 42)

Marker	c	Configuration	Max Bias	FDR Sig.	Min FDR P-val
†	0x03	byte_12_in→byte_12_out	1.6x	0	9.36e-02
†	0x04	byte_8_in→byte_8_out	1.6x	0	9.37e-02
†	0x91	byte_12_in→byte_12_out	1.5x	0	1.22e-01
†	0x02	byte_8_in→byte_8_out	1.6x	0	1.50e-01
†	0x01	byte_2_in→byte_3_out	3.1x	0	2.24e-01
†	0xe1	byte_4_in→byte_5_out	1.5x	0	2.40e-01
†	0x03	byte_0_in→byte_0_out	1.5x	0	2.45e-01
†	0x02	byte_14_in→byte_15_out	1.6x	0	3.05e-01
†	0xe1	byte_2_in→byte_2_out	1.5x	0	3.78e-01
†	0x91	byte_0_in→byte_1_out	1.5x	0	3.96e-01
†	0x01	byte_0_in→byte_0_out	2.9x	0	1.00e+00
†	0x01	byte_12_in→byte_12_out	2.9x	0	1.00e+00
†	0x01	byte_2_in→byte_2_out	2.9x	0	1.00e+00
†	0x01	byte_6_in→byte_6_out	2.6x	0	1.00e+00
†	0x01	byte_8_in→byte_8_out	2.9x	0	1.00e+00
†	0x02	byte_0_in→byte_1_out	1.5x	0	1.00e+00
†	0x02	byte_12_in→byte_12_out	1.5x	0	1.00e+00
†	0x02	byte_2_in→byte_2_out	1.5x	0	1.00e+00
†	0x03	byte_10_in→byte_10_out	1.5x	0	1.00e+00
†	0x03	byte_4_in→byte_4_out	1.5x	0	1.00e+00

Table 21: Top Noteworthy Findings for 4-Round Kuznyechik (Seed 42)

Marker	c	Configuration	Max Bias	FDR Sig.	Min FDR P-val
†	0x91	byte_8_in→byte_8_out	1.7x	0	1.56e-02
†	0x03	byte_12_in→byte_12_out	1.6x	0	5.94e-02
†	0x04	byte_4_in→byte_4_out	1.6x	0	1.22e-01
†	0x03	byte_10_in→byte_10_out	1.6x	0	1.87e-01
†	0xe1	byte_4_in→byte_5_out	1.5x	0	2.45e-01
†	0xbe	byte_14_in→byte_14_out	1.6x	0	4.44e-01
†	0xbe	byte_6_in→byte_6_out	1.5x	0	4.57e-01
†	0xe1	byte_2_in→byte_2_out	1.5x	0	4.79e-01
†	0x91	byte_0_in→byte_1_out	1.6x	0	4.79e-01
†	0xbe	byte_14_in→byte_15_out	1.5x	0	5.69e-01
†	0x01	byte_0_in→byte_0_out	2.9x	0	1.00e+00
†	0x01	byte_12_in→byte_12_out	2.9x	0	1.00e+00
†	0x01	byte_2_in→byte_2_out	2.9x	0	1.00e+00
†	0x01	byte_2_in→byte_3_out	2.7x	0	1.00e+00
†	0x01	byte_6_in→byte_6_out	2.9x	0	1.00e+00
†	0x01	byte_8_in→byte_8_out	2.9x	0	1.00e+00
†	0x02	byte_0_in→byte_1_out	1.5x	0	1.00e+00
†	0x02	byte_12_in→byte_12_out	1.5x	0	1.00e+00
†	0x02	byte_14_in→byte_15_out	1.5x	0	1.00e+00
†	0x02	byte_2_in→byte_2_out	1.5x	0	1.00e+00

Table 22: Top Noteworthy Findings for 3-Round Kuznyechik (Seed 42)

Marker	c	Configuration	Max Bias	FDR Sig.	Min FDR P-val
‡	0x01	byte_0_in→byte_0_out	10.1x	10896	0.00e+00
‡	0x01	byte_12_in→byte_12_out	10.3x	10996	0.00e+00
‡	0x01	byte_2_in→byte_2_out	10.0x	10955	0.00e+00
‡	0x01	byte_2_in→byte_3_out	10.1x	10985	0.00e+00
‡	0x01	byte_6_in→byte_6_out	10.4x	10991	0.00e+00
‡	0x01	byte_8_in→byte_8_out	10.4x	10980	0.00e+00
†	0x02	byte_12_in→byte_12_out	1.6x	0	1.51e-01
†	0x04	byte_8_in→byte_8_out	1.5x	0	1.60e-01
†	0x91	byte_12_in→byte_12_out	1.5x	0	2.45e-01
†	0xbe	byte_14_in→byte_14_out	1.5x	0	3.01e-01
†	0x91	byte_2_in→byte_2_out	1.5x	0	3.01e-01
†	0xe1	byte_12_in→byte_12_out	1.5x	0	3.01e-01
†	0x91	byte_4_in→byte_4_out	1.5x	0	3.79e-01
†	0x02	byte_2_in→byte_2_out	1.5x	0	4.78e-01
†	0xbe	byte_2_in→byte_2_out	1.5x	0	4.80e-01
†	0x03	byte_0_in→byte_0_out	1.5x	0	5.95e-01
†	0x04	byte_2_in→byte_2_out	1.5x	0	5.95e-01
†	0xe1	byte_4_in→byte_5_out	1.5x	0	6.16e-01
†	0x03	byte_10_in→byte_10_out	1.5x	0	7.12e-01
†	0x02	byte_14_in→byte_15_out	1.5x	0	7.13e-01

Table 23: Top Noteworthy Findings for 2-Round Kuznyechik (Seed 42)

Marker	c	Configuration	Max Bias	FDR Sig.	Min FDR P-val
†	0xbe	byte_6_in→byte_6_out	1.7x	0	2.81e-02
†	0x02	byte_0_in→byte_1_out	1.6x	0	4.31e-02
†	0xe1	byte_6_in→byte_6_out	1.6x	0	5.93e-02
†	0x03	byte_10_in→byte_10_out	1.6x	0	7.07e-02
†	0x03	byte_4_in→byte_4_out	1.5x	0	1.22e-01
†	0x02	byte_8_in→byte_8_out	1.5x	0	1.87e-01
†	0x04	byte_8_in→byte_8_out	1.5x	0	2.45e-01
†	0x02	byte_12_in→byte_12_out	1.5x	0	2.97e-01
†	0x91	byte_2_in→byte_2_out	1.5x	0	3.01e-01
†	0x03	byte_6_in→byte_6_out	1.5x	0	4.57e-01
†	0xbe	byte_14_in→byte_14_out	1.5x	0	4.58e-01
†	0xe1	byte_8_in→byte_8_out	1.5x	0	4.79e-01
†	0xbe	byte_0_in→byte_0_out	1.5x	0	4.79e-01
†	0x91	byte_0_in→byte_0_out	1.5x	0	4.80e-01
†	0xbe	byte_2_in→byte_2_out	1.5x	0	8.30e-01
†	0x91	byte_4_in→byte_4_out	1.5x	0	8.54e-01
†	0xbe	byte_14_in→byte_15_out	1.5x	0	9.95e-01
†	0x01	byte_12_in→byte_12_out	2.9x	0	1.00e+00
†	0x01	byte_2_in→byte_2_out	2.7x	0	1.00e+00
†	0x01	byte_2_in→byte_3_out	2.9x	0	1.00e+00

Table 24: Top Noteworthy Findings for 1-Round Kuznyechik (Seed 42)

Marker	c	Configuration	Max Bias	FDR Sig.	Min FDR P-val
‡	0x01	byte_0_in→byte_0_out	10.3x	10959	0.00e+00
‡	0x01	byte_12_in→byte_12_out	10.1x	11029	0.00e+00
‡	0x01	byte_2_in→byte_2_out	10.1x	10976	0.00e+00
‡	0x01	byte_2_in→byte_3_out	10.3x	10960	0.00e+00
‡	0x01	byte_6_in→byte_6_out	10.0x	10986	0.00e+00
‡	0x01	byte_8_in→byte_8_out	10.2x	10972	0.00e+00
†	0xbe	byte_6_in→byte_6_out	1.7x	0	2.81e-02
†	0x02	byte_0_in→byte_1_out	1.6x	0	4.31e-02
†	0xe1	byte_6_in→byte_6_out	1.6x	0	5.93e-02
†	0x03	byte_10_in→byte_10_out	1.6x	0	7.07e-02
†	0x03	byte_4_in→byte_4_out	1.5x	0	1.22e-01
†	0x02	byte_8_in→byte_8_out	1.5x	0	1.87e-01
†	0x04	byte_8_in→byte_8_out	1.5x	0	2.45e-01
†	0x02	byte_12_in→byte_12_out	1.5x	0	2.97e-01
†	0x91	byte_2_in→byte_2_out	1.5x	0	3.01e-01
†	0x03	byte_6_in→byte_6_out	1.5x	0	4.57e-01
†	0xbe	byte_14_in→byte_14_out	1.5x	0	4.58e-01
†	0xe1	byte_8_in→byte_8_out	1.5x	0	4.79e-01
†	0xbe	byte_0_in→byte_0_out	1.5x	0	4.79e-01
†	0x91	byte_0_in→byte_0_out	1.5x	0	4.80e-01

D Detailed statistical analysis for c=0x4, byte_8_in→byte_8_out

In this last appendix, we present a snapshot of the output txt file for 9 rounds, for only one test (the text file, for every number of rounds analysis is rather large, from 75 to 120 pages [46]).

```
================================================================================
DETAILED STATISTICAL ANALYSIS for c=0x4, byte_8_in->byte_8_out
================================================================================
DISTRIBUTION PROPERTIES:
  Total unique pairs observed: 65,280
  Mean/Median/Std Dev count: 76.29 / 76.00 / 8.74
  Max/Min count: 130 / 41
  Skewness/Kurtosis: 0.109 / 0.027

ENHANCED BIAS METRICS:
  KL Divergence: 0.006574
  Max Chi-square: 37.81
  Relative Entropy: 0.999

NORMALITY TESTS (on the distribution of observed counts):
  Shapiro-Wilk Test: Skipped. Reason: Dataset too large for Shapiro-Wilk (N >= 5000)
  Anderson-Darling Test: Statistic=43.217
    Critical Values (Sig Levels): [0.576, 0.656, 0.787, 0.918, 1.092] ([15.0, 10.0, 5.0, 2.5, 1.0])
    (Interpretation: Statistic > Critical Value at a given significance level suggests non-normal distribution)

GOODNESS-OF-FIT TESTS (vs. Uniform Distribution):
  (Evaluates if the overall distribution of 65,280 pairs is uniform. Degrees of freedom: 65,279)
  Chi-square Test: Statistic=65,319.34, P-value=4.548e-01
  G-test (Log-likelihood): Statistic=65,479.50, P-value=2.890e-01
    (Interpretation: A very small P-value, e.g., < 0.001, provides strong evidence that the cipher's
     output for this configuration is not uniformly distributed as a whole.)

CLUSTER ANALYSIS:
  Found 1 clusters, 0 significant
```

```
  Cluster 1: 65280 members, combined p=1.000e+00, avg bias=1.00x

UNCORRECTED SIGNIFICANT PAIRS (raw p < 0.05, before FDR):
  Found 2921 pairs significant before correction
  Top 5 by raw p-value:
    0x00000000000000cf0000000000000000 -> 0x00000000000000760000000000000000 (Bias: 1.30x, p: 0.014)
    0x000000000000005b0000000000000000 -> 0x00000000000000b000000000000000000 (Bias: 1.30x, p: 0.014)
    0x000000000000007a0000000000000000 -> 0x00000000000000700000000000000000 (Bias: 0.72x, p: 0.013)
    0x00000000000000970000000000000000 -> 0x00000000000000c600000000000000000 (Bias: 1.26x, p: 0.033)
    0x000000000000009e0000000000000000 -> 0x00000000000000540000000000000000 (Bias: 1.32x, p: 0.008)

SIGNIFICANT DIFFERENTIAL PAIRS (FDR-corrected p < 7.033e-02):
  (These are specific (Input Diff, Output Diff) pairs whose observed frequencies
   are statistically different from expected, after multiple-test correction.)
  Found 1 significant pairs.
  Top 10 by corrected p-value:
  Input Diff (A)                      Output Diff (B)                     Obs Count  Bias     Corr P-val
  ---------------------------------  ---------------------------------  ---------- -------- ------------
  0x00000000000000290000000000000000  0x000000000000008d0000000000000000  130        1.7x     1.85e-03
================================================================================
```

COMBINED SIGNIFICANCE DETECTED for 9 rounds

```
    Config: byte_8_in->byte_8_out, c=0x4
    Found 342 pairs with bias > 1.3 AND p < 0.1
      - 0x00000000000000290000000000000000 -> 0x000000000000008d0000000000000000 (Bias: 1.70x, p: 0.000)
      - 0x00000000000000d30000000000000000 -> 0x00000000000000540000000000000000 (Bias: 1.52x, p: 0.000)
      - 0x00000000000000e40000000000000000 -> 0x00000000000000480000000000000000 (Bias: 1.51x, p: 0.000)
```

BIAS PERSISTENCE ANOMALY for 9 rounds

```
    Config: byte_8_in->byte_8_out, c=0x4
    Observed bias: 1.70x vs Expected: 0.125x
    Ratio: 13.6x higher than expected decay
```

CRITICAL ALERT: Statistically significant characteristic found for 9 rounds!

```
    Config: byte_8_in->byte_8_out, c=0x4
    Found 1 significant pairs (threshold: 1): Input->Output
    0x00000000000000290000000000000000 -> 0x000000000000008d0000000000000000 (Bias:1.7, p-val:1.85e-03)
```