IACR Transactions on Symmetric Cryptology
ISSN 2519-173X, Vol. 2020, No. 3, pp. 46–86. DOI:10.13154/tosc.v2020.i3.46-86

# Dasta – Alternative Linear Layer for Rasta

Phil Hebborn and Gregor Leander

Ruhr University Bochum, Bochum, Germany
phil.hebborn@rub.de, gregor.leander@rub.de

**Abstract.**
Progress in the areas of multi-party computation (MPC) and fully homomorphic encryption (FHE) caused the demand of new design strategies, that minimize the number of multiplications in symmetric primitives. Rasta is an approach for a family of stream ciphers with an exceptional low AND depth, which equals the number of ANDs per encrypted bit. This is achieved in particular by randomizing parts of the computation with the help of a PRNG, implying that the security arguments rely on the provided randomness and the encryption/ decryption is potentially slowed down by this generation.
In this paper we propose a variant of Rasta that achieves the same performance with respect to the AND depth and the number of ANDs per encrypted bit, but does not rely on a PRNG, i.e. is based on fixed linear layers.

**Keywords:** Symmetric encryption, FHE, multiplicative complexity, multiplicative depth, Rasta

## 1 Introduction

While classical block ciphers are designed, besides being secure, to perform efficiently in hardware and software, recent schemes for FHE require new tradeoffs.

Important FHE schemes like the BGV scheme [BGV11] come with a large ciphertext expansion. To prevent the transfer of large ciphertexts, it makes sense to encrypt them symmetrically, send them to the other party, let the other party double encrypt the ciphertexts under FHE and then reverse the symmetric encryption through homomorphic evaluation. Typically an AND operation, and in particular a high AND depth, is much more expensive than a XOR operation in the FHE context. Therefore classical symmetric primitives are not suitable for this scenario and ciphers which lower the amount of ANDs at the price of increasing the number of XORs are needed to increase the performance of decryption under FHE.

Motivated by this observation, several approaches have been considered to either lower the AND depth or the number of ANDs per encrypted bit. In a nutshell, LowMC [Alb+15] and Kreyvium [Can+15] optimize the number of ANDs per bit (which can be as low as 3 to 4 ANDs per bit) while the AND depth is at least 11, depending on the targeted security level. FLIP [Méa+16] on the other hand achieves a very low AND depth of 4, while the number of ANDs per bit is significantly higher (1072 ANDs per bit). Rasta [Dob+18] finally is an approach for a family of stream ciphers based on the ASASA design [BBK14], having an AND depth of only 4 to 6 and the same number of ANDs per encrypted bit. That is, it aims at being optimal with respect to both main metrics simultaneously.

The core idea of Rasta, initiated already in the design of FLIP, is to make large parts of the computation nonce-dependent but key-independent. The advantage of key-independent variations is that, on the one hand, the (key-dependent) AND-depth can be kept very low,


Received: 2019-06-01 Revised: 2020-06-01 Accepted: 2020-08-01 Published: 2020-09-28

while at the same time many standard attacks are not applicable to Rasta due to the nonce-dependent variations.

More specifically, Rasta interleaves a very simple non-linear function with (pseudo) randomly generated affine bijections. The non-linear layer is a generalized variant of the $\chi$ function used in Keccak [Ber+13] and already studied in [Dae95]. The affine layers are generated using a PRNG together with a nonce and a counter.

While this design strategy is very suitable to achieve the above mentioned goals, it has several important drawbacks. First, fundamentally, Rasta relies on a pseudo random function (to generate the matrices) to construct a pseudo random function (the Rasta key stream). While this does not reduce the applicability in practice, it does make us curious if one can create a framework for Rasta variants that avoids the use of a PRNG, without sacrificing efficiency with respect to the AND depth and the number of AND gates per encrypted bit.

Moreover, this process slows down the implementation. This slowdown happens in the pre-computation phase, where the generation of random layers is handled (with the costly restriction of checking invertibility) as well as in the actual generation of the key stream, where one has to perform multiplication with random matrices. This practically disallows any possible optimizations. The only optimizations that are possible are generic speed-ups by improving the vector-matrix multiplication, with the natural limitations.

For LowMC, on the other hand, further optimizations have been applied in [Din+19]. The main idea is to join several matrices from consecutive rounds to speed-up the computation. This is possible mainly due to the partial non-linear layer of LowMC. As Rasta does not use a partial non-linear layer, but a block-wide version of the Keccack S-box, those are not applicable to Rasta. The main other approach, given in [Per+17], is to use matrices that are not random but random looking (see the Fibonacci Feistel Network approach from [Per+17]) has the inherent drawback that any security argument based on the randomness of the linear layer breaks down completely. It is exactly this drawback that we resolve for Rasta.

## Our Contribution

Replacing the random generation of the linear layers with a pure nonce-based approach has to be treated with care. There are many choices for the linear mappings that, when being used, would immediately render the scheme insecure. In Rasta an attacker can modify the nonce, but as this is input to a PRNG, being used to generate the linear layer, the attacker has virtually no control about the resulting linear mappings. This renders the probability of choosing such a weak instance negligible. By removing the PRNG in this process, the attacker has direct influence on the specific linear layers being used. This in turn means that we have to restrict the choice of possible linear layers to a subset that excludes all weak instances.

Our procedure to accomplish this task is a modular approach. Instead of randomly generated linear layers, that is random invertible binary matrices, we consider linear layers that are split in two parts, (i) a variable bit permutation and (ii) a fixed linear transformation. This modular approach has several advantages. First, as the linear layers are, up to a permutation of their inputs, fixed, their implementation can now be optimized. Second, and more importantly, the modular approach can be lifted to a sound modular approach for the security analysis, as follows.

In a first step, we derive conditions on how to construct a fixed linear layer that allows to lift the randomness-based security arguments for Rasta to arguments for a fixed linear layer. This is in particular of interest for linear cryptanalysis and truncated differentials. For linear cryptanalysis this technically involves an in-depth study of the non-linear layer of Rasta, i.e. the generalized Keccak Sbox that was already introduced in [Dae95] and might be of independent interest. We aim at criteria for the linear layers that are invariant

under composition with permutation matrices. Thus the criteria are automatically fulfilled by all members of the family as soon as they are fulfilled by a single member.

In a second step we discuss criteria for a suitable family of bit permutations. As we cannot make use of any PRNG here, the attacker, by choosing the nonce, is able to directly choose any member of the specified family to be used in the generation of the key-stream block. Here, in a nutshell, one has in particular to avoid that the specified family contains bit permutations that are very similar and would reveal information about the key. We developed the necessary tools to formalize precise criteria that ensure that these attacks are not a threat to our new design.

Based on those considerations we are able to present a new variant of Rasta, called Dasta[1], that uses the same parameters, i. e. block length and number of rounds as the original but avoids the random generation of linear layers. In an offline key stream generation comparison, Dasta is about 200 to 400 times faster than Rasta (See Table 3). This is due to the fact that we can generate the linear layer of Dasta much more efficiently. In the FHE setting with an implementation in the HElib [HS13], Dasta achieves runtime improvements compared to Rasta of about 15% to 20% (see Table 4). This improvement basically relies on the reuse of matrices in Dasta within the generation of one key stream block.

### Related Work

For LowMC, the paper by Albrecht *et al.* [Alb+15] presents alternative reduced linear layers called Fibonacci Feistel Networks. It can lower the size of the representation of the linear layer and improve the performance of computing the linear layer depending on the implementation. While this approach might also be possible for Dasta, we do not see how security arguments could be developed for such a variant.

### Organisation

We start by recalling basic definitions and the design of Rasta in Section 2. Next, in Section 3, we describe our nonce-based variant. Section 4 contains our findings on suitable families of bit permutations while Sections 5 and 6 deal with establishing the criteria for the fixed linear layer. We conclude the paper with a focus on possible future topics in Section 8.

## 2 Preliminaries

We denote by $\mathbb{F}_2$ the field with two elements and by $\mathbb{F}_2^n$ the $n$-dimensional vector space over $\mathbb{F}_2$. By $\mathbf{0}$ we denote the zero vector, $\mathbf{1}$ is the vector containing all 1s. We refer to the $i$-th unit vector in $\mathbb{F}_2^n$ for $0 \leq i < n$ by $e_i$, i. e. the $i$-th coordinate of $e_i$ is 1, while all other coordinates are 0. All indices are considered modulo $n$.

With $\mathbf{hw}(x)$ we denote the Hamming weight of $x \in \mathbb{F}_2^n$, i.e.

$$\mathbf{hw}(x) := |\{i \in \{0, 1, \cdots, n-1\} \mid x_i = 1\}|.$$

The elements of a vector $x$ of length $n$ can be addressed by $x_i$ or $x[i]$, where $0 \leq i < n$. Another meaning for the square brackets is given by $[i] = \{0, 1, \cdots, i-1\}$ for $i > 0$.

The modulo operator is defined in two ways: Let $a, b, c \in \mathbb{Z}$, then $a \equiv b \bmod c$ iff $c \mid (a - b)$. Secondly we have $a \mod b = a - b \cdot \left\lfloor \frac{a}{b} \right\rfloor$.

We denote the addition in $\mathbb{F}_2^n$ by $+$, which is equivalent to bit-wise XOR, while the element-wise multiplication in $\mathbb{F}_2^n$ is denoted by $\odot$. This is equivalent to the bit-wise AND:

$$(x \odot y)_i = x_i y_i.$$

[1] For **D**eterministic R**asta**

The function $S^a : \mathbb{F}_2^n \to \mathbb{F}_2^n$ denotes the rotation by $a$ positions to the right, if we write $x$ as a column vector:

$$S^a(x)_i = x_{i-a}.$$

We denote the Walsh-Hadamard transformation of $f : \mathbb{F}_2^n \to \mathbb{F}_2^n$ with $\alpha, \beta \in \mathbb{F}_2^n$ by

$$\hat{f}(\alpha, \beta) \coloneqq \sum_{x \in \mathbb{F}_2^n} (-1)^{\langle \beta, f(x) \rangle + \langle \alpha, x \rangle} \tag{1}$$

where $\langle a, b \rangle$ is the dot product.

The linear correlation of a function $f : \mathbb{F}_2^n \to \mathbb{F}_2^n$ with the input mask $\alpha \in \mathbb{F}_2^n$ and the output mask $\beta \in \mathbb{F}_2^n$ is defined by

$$C_f(\alpha \to \beta) \coloneqq \frac{\hat{f}(\alpha, \beta)}{2^n}.$$

Let $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$. The derivative of $F$ in direction $\alpha$ is defined as

$$\begin{aligned} \Delta : \mathbb{F}_2^n \times (\mathbb{F}_2^n \to \mathbb{F}_2^n) &\to (\mathbb{F}_2^n \to \mathbb{F}_2^n) \\ \Delta_\alpha(F)(x) &\coloneqq F(x) + F(x + \alpha). \end{aligned} \tag{2}$$

The probability of a differential over $F$ with input mask $\alpha \in \mathbb{F}_2^n$ and output mask $\beta \in \mathbb{F}_2^n$ is defined by

$$\Pr\left[\alpha \xrightarrow{F} \beta\right] \coloneqq \frac{|\{x \mid \Delta_\alpha(F)(x) = \beta\}|}{2^n}.$$

The linear branch number of a matrix $L \in \mathbb{F}_2^{n \times n}$ is defined as

$$\mathcal{B}_\ell \coloneqq \min_{x \in \mathbb{F}_2^n \setminus \{\mathbf{0}\}} \left(\mathbf{hw}(x) + \mathbf{hw}(L^\top x)\right).$$

Its importance for linear cryptanalysis originates from [Dae95], see also [DR01].

We denote by $S_n$ the symmetric group of degree $n$, i. e. $S_n$ contains all permutations on $n$ elements.

**Constructing Binary Linear Codes** A linear binary $[n, k, d]$ code $C$ is a subspace of dimension $k$ in $\mathbb{F}_2^n$ with code words of length $n$, and a minimum distance between two code words of $d$. The generator matrix $G$ of a code $C$ has the format $k \times n$ and the rows of $G$ build a basis of $C$, so that

$$C = \{x \cdot G \mid x \in \mathbb{F}_2^k\}.$$

For a given length $n$ and a minimum distance $d$, we can compute an $[n, k, \geq d]$ narrow-sense BCH code [MS78] efficiently. We write the narrow-sense BCH code $C$ with given $n, d$ and a maximal dimension $k$ as

$$\mathbf{BCH}(n, d) \coloneqq C.$$

We can shorten an $[n, k, d]$ code $C$ in a coordinate $i \in [k]$, where we assume that $\exists c \in C \colon c_i = 1$, which means that exactly half of the code words are 1 in the $i$-th coordinate. This lowers the dimension of the code by 1 by removing all code words $c \in C$ where $c_i = 1$. Now we remove the $i$-th coordinate of the remaining code words, which is 0 for all of them. This leads to a $[n - 1,\ k - 1,\ \geq d]$ code $C'$, since

$$\min_{c \in C' \setminus \{\mathbf{0}\}} \mathbf{hw}(c) \geq \min_{c \in C \setminus \{\mathbf{0}\}} \mathbf{hw}(c).$$

In the same way the code $C$ can be shortened in multiple coordinates $I \subseteq [n]$ to construct the $[n - |I|, k - |I|, \geq d]$ code $C_s$. We write the shortening as

$$\textbf{Shorten}(C, I) \coloneqq C_s.$$

To extend an $[n, k, d]$ code $C$ with the generator matrix $G \in \mathbb{F}_2^{k \times n}$ we append an extension matrix $E \in \mathbb{F}_2^{k \times e}$. The generator matrix $G' = (G|E)$ defines an $[n + e, k, \geq d]$ code $C'$. We write the extension as

$$\textbf{Extension}(C, E) \coloneqq C'.$$

## 2.1 Rasta

Rasta [Dob+18] is a family of stream ciphers with a low amount of ANDs per encrypted bit and a low AND depth. It works on $n$-bit blocks, where the key size matches the block size. A block of the key stream is generated by

$$T_{N,i}(k) + k$$

with the key $k$, the nonce $N$ and the block counter $i$ as depicted in Figure 1. For every encryption a unique pair of $(N, i)$ has to be used.

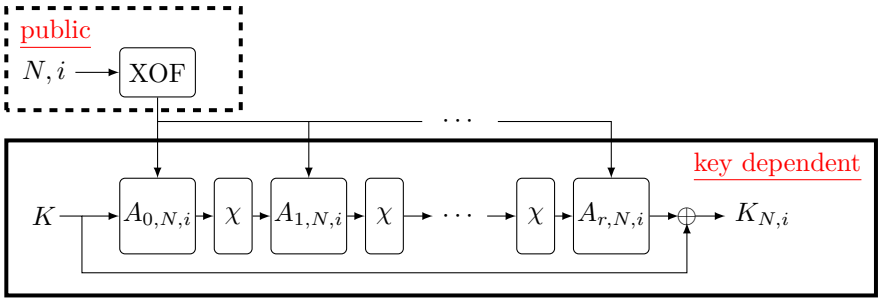


**Figure 1:** The $r$-round Rasta construction to generate the key stream $K_{N,i}$ for block $i$ under nonce $N$ with affine layers $A_{j,N,i}$.

The permutation has $r$ rounds, which contain an affine layer $A_{j,N,i}$ and a non-linear layer $\chi$, and another affine layer after the last round:

$$T_{N,i} = A_{r,N,i} \circ \chi \circ A_{r-1,N,i} \circ \cdots \circ \chi \circ A_{1,N,i} \circ \chi \circ A_{0,N,i}.$$

The non-linear layer is the $\chi$-transformation of Keccak[Ber+13], which is bijective for an odd block length $n$:

$$\chi(x) = x + S^{-2}(x) + S^{-1}(x) \odot S^{-2}(x).$$

Each affine layer $A_{j,N,i}$ consists of the multiplication with an invertible matrix $M_{j,N,r} \in \mathbb{F}_2^{n \times n}$ and the addition of a round constant $c_{j,N,i} \in \mathbb{F}_2^n$:

$$A_{j,N,i}(x) = M_{j,N,r} \cdot x + c_{j,N,i}.$$

The affine layer is generated randomly using a PRNG, or rather an extendable-output function (XOF) (see for more details [Nat15]), which takes as inputs $N$ and $i$. Since a pair of nonce and block counter should not be reused with the same key, every permutation is only evaluated once. Table 1 shows a subset of the proposed versions of Rasta, which we examine in the following.

In a chosen-nonce setting the attacker cannot control the affine layers directly due to the XOF used to generate them. Therefore the attacker has to make about $p^{-1}$ calls to the XOF on average, to get an affine layer which has a characteristic that occurs with probability $p$. For the claimed security level $s$ the data complexity of an attack is limited to $D = \frac{2^{s/2}}{n}$.

**Table 1:** RASTA versions.

| RASTA version | security level | block/ key size | rounds |
|---|---|---|---|
| 80-6 | 80 | 219 | 6 |
| 80-4 | 80 | 327 | 4 |
| 128-6 | 128 | 351 | 6 |
| 128-5 | 128 | 525 | 5 |
| 128-4 | 128 | 1877 | 4 |
| 256-6 | 256 | 703 | 6 |
| 256-5 | 256 | 3545 | 5 |

# 3 Specification of DASTA

In this section we define DASTA exactly. However, the motivation for those choices will only become clear in the following sections that discuss the security arguments.

We define the blockwise key stream generation of DASTA (see Algorithm 1 and Fig. 2) by

$$B_i(k) = L \circ P_{r,i} \circ \chi \circ L \circ P_{r-1,i} \circ \cdots \circ \chi \circ L \circ P_{1,i} \circ \chi \circ L \circ P_{0,i}(k) + k, \tag{3}$$

where $k$ is the secret key, $0 \leq i < D$ is the block counter, and $D = \left\lceil \frac{2^{s/2}}{n} \right\rceil$ is the data limit. $L$ is a fixed linear transformation, which depends on the DASTA version. The block counter determines a $(r+1)$-tuple of bit permutations $(P_{0,i}, \ldots, P_{r,i})$, which we call instance in the following. In Section 4 we discuss the requirements for the set of instances. We define the block lengths, number of rounds, and security parameters for the different versions of DASTA analogous to the RASTA versions in Table 1. This leads in particular to the same AND depth and AND count as RASTA.

We do not use round constants due to the results of Appendix B. Without round constants the key $\mathbf{0}$ produces as key stream $\mathbf{0}$ for all blocks. Therefore we define this key as invalid for DASTA.

**Algorithm 1** $B_i(k)$ (Generation of key stream block)

**Input:** Key $k$, Block counter $i$
**Output:** Key stream block $y$

1: $x \leftarrow k$
2: **for** $m \leftarrow 0$ **to** $r-1$ **do**
3: $\quad x \leftarrow P_{m,i}(x)$
4: $\quad x \leftarrow L(x)$
5: $\quad x \leftarrow \chi(x)$
6: $x \leftarrow P_{r,i}(x)$
7: $x \leftarrow L(x)$
8: **return** $x + k$

**The Linear Transformation $L$.** The linear transformation is chosen as $L = A^\top$, where $G = (I_n|A)$ is the generator matrix from a linear $[2n,\ n,\ d]$ code $C$ and $I_n$ is the $n \times n$ identity matrix. This means that we have a linear branch number $B = d$. In Section 3.1 we list the constructions of the codes for the different DASTA versions, the actual linear layers are provided online due to their size.[2]. The linear branch number of all versions is large enough to prevent attacks based on linear approximations (as shown in Section 5).

[2] https://github.com/phil-hebborn/Dasta

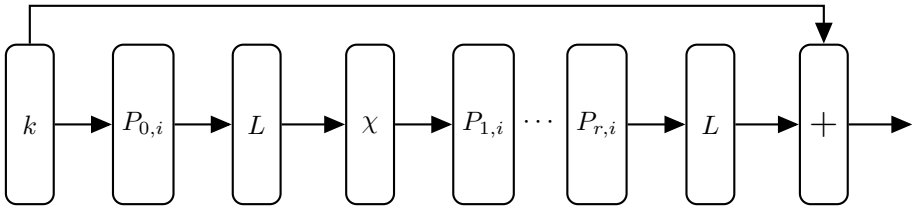


**Figure 2:** Generation of $i$-th block of Dasta.

We create these linear codes by choosing BCH codes that have a dimension and length close to $n$ and $2n$ while the distance is large. To get a code which has exactly dimension $n$ and length $2n$, we use the shortening and extension for codes described in Section 2. Any used extension matrix $E$ is chosen randomly and then fixed with the restriction that for the systematic generator matrix $G = (I_n|A|E)$ of an extended code $(A|E)$ has full rank.

**The Bit Permutations $P_{j,i}$.** Under abuse of notation, we apply a permutation $\pi \in S_n$ to an element $x \in \mathbb{F}_2^n$ to denote the application of the corresponding permutation matrix. That is $\pi(x)$ corresponds to the vector whose positions are permuted according to $\pi$. More precisely,

$$\pi \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{pmatrix} = \begin{pmatrix} x_{\pi(0)} \\ x_{\pi(1)} \\ \vdots \\ x_{\pi(n-1)} \end{pmatrix}.$$

The block counter $0 \leq i < D$ determines the used bit permutation. In other words every block counter corresponds to an $(r+1)$-tuple of bit permutations.

Before we can define how the block counter is mapped to a specific $(r+1)$-tuple of bit permutations, we need some auxiliary notation.

As we will see in Section 4, for our analysis it is the cycle structure of the permutations used that matters most. Among all permutations of a given cycle structure, we choose the simplest one. For example if we have to choose a permutation of 5 elements with one cycle of length 2 and one cycle of length 3 we choose

$$\pi = (0\ 1)(2\ 3\ 4)$$

in the cycle notation, which is

| $x$ | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| $\pi(x)$ | 1 | 0 | 3 | 4 | 2 |

More generally, given cycle length $c_0, \ldots, c_{m-1}$ we define the permutation

$$g(c_0, \ldots, c_{m-1}) = (0\ 1 \cdots c_0 - 1)(c_0 \cdots c_0 + c_1 - 1) \cdots (n - c_{m-1} \cdots n - 1).$$

as the specific permutation to use. When viewed as a permutation of bits, the structure of these permutations is shown in Figure 3.

The basic idea is we fix a permutation $\pi$ having a suitable cycle structure first. For a block counter $i$, we ideally choose

$$(P_{i,0}, \ldots, P_{i,r+1}) = (\pi^i, \ldots, \pi^i),$$

that is we chose the same bit permutation in all rounds. However, for most parameters and the given data limit, constructing suitable permutations $\pi$ that allow strong security arguments was not possible. Therefore, we have to use a slightly more involved construction

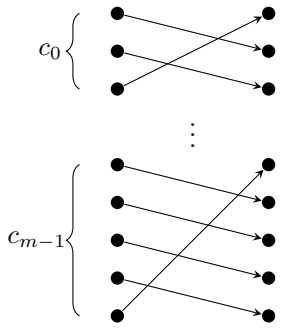


**Figure 3:** Permutation generated by function $g$.

where we fix up to two permutations and use different powers of those, determined again by the block counter. This construction gives more freedom in the choice of the permutations and finally improves the security bounds (see Section 4). The explicit choice we use is given in Section 3.1.

## 3.1 Defining the Instance

In the following we give the exact definition of the linear layers and the set of bit permutations for all versions of Dasta. Recall that $n$ is the block size, $r$ the number of rounds, and $s$ the security parameter.

### Dasta **80-6: n = 219, r = 6, s = 80**

For the definition of the linear layer we start with a BCH code of length 455 and dimension 239. This BCH code is shortened in the first 20 coordinates and then extended by $E_{80\text{-}6}$, so that the resulting code has length $455 - 20 + 3 = 2n$ and dimension $239 - 20 = 219 = n$ as required. The minimal distance is at least 45 by construction. In short, we use

$$C_{80\text{-}6} = \mathbf{Extension}(\mathbf{Shorten}(\mathbf{BCH}(455, 45), [20]), E_{80\text{-}6}), \quad E_{80\text{-}6} \in \mathbb{F}_2^{219\times 3}.$$

We define one base permutation

$$\pi = g(5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 27)$$

as discussed above. Note that the order of elements within $g$ does not matter for our purpose and we decided to list first the primes in ascending order, followed by the non-primes.

Given the block counter $i$ expressed as

$$i = \left\lceil \sqrt{D} \right\rceil i_1 + i_0 \text{ with } 0 \le i_0, i_1 < \sqrt{D}$$

we define the round permutations as

$$(P_{0,i}, \ldots, P_{6,i}) = \left(\pi^i, \pi^{i_0}, \pi^{i_1}, \pi^{i_0}, \pi^{i_1}, \pi^{i_0}, \pi^{i_1}\right).$$

### Dasta **80-4: n = 327, r = 4, s = 80**

With the same notation as above we construct the linear layer using the following code

$$C_{80\text{-}4} = \mathbf{Extension}(\mathbf{Shorten}(\mathbf{BCH}(657, 42), [6]), E_{80\text{-}4}), \quad E_{80\text{-}4} \in \mathbb{F}_2^{327\times 3}.$$

We use two base permutations

$$\pi_0 = g(13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 27)$$
$$\pi_1 = g(17, 19, 23, 29, 31, 37, 41, 43, 47, 40)$$

and, given the block counter $i = \left\lceil \sqrt{D} \right\rceil i_1 + i_0$ we define

$$(P_{0,i}, P_{1,i}, P_{2,i}, P_{3,i}, P_{4,i}) = \left(\pi_0^i, \pi_1^{i_0}, \pi_1^{i_1}, \pi_1^{i_0}, \pi_1^{i_1}\right).$$

### Dasta **128-6: n = 351, r = 6, s = 128**

With the same notation as above we construct the linear layer using the following code

$$C_{128\text{-}6} = \mathbf{Extension}(\mathbf{Shorten}(\mathbf{BCH}(771, 45), [84]), E_{128\text{-}6}), \quad E_{128\text{-}6} \in \mathbb{F}_2^{351\times 15}.$$

We use the base permutation

$$\pi = g(3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 25)$$

and, given the block counter $i = \left\lceil \sqrt{D} \right\rceil i_1 + i_0$ we define

$$(P_{0,i}, \ldots, P_{6,i}) = \left(\pi^i, \pi^{i_0}, \pi^{i_1}, \pi^{i_0}, \pi^{i_1}, \pi^{i_0}, \pi^{i_1}\right).$$

### Dasta **128-5: n = 525, r = 5, s = 128**

With the same notation as above we construct the linear layer using the following code

$$C_{128\text{-}5} = \mathbf{Shorten}(\mathbf{BCH}(1057, 85), [7]).$$

We use two base permutations

$$\pi_0 = g(2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 24)$$
$$\pi_1 = g(29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 57)$$

and, given the block counter $i = \left\lceil \sqrt{D} \right\rceil i_1 + i_0$ we define

$$(P_{0,i}, P_{1,i}, P_{2,i}, P_{3,i}, P_{4,i}, P_{5,i}) = \left(\pi_0^i, \pi_0^i, \pi_1^{i_0}, \pi_1^{i_1}, \pi_1^{i_0}, \pi_1^{i_1}\right).$$

### Dasta **128-4: n = 1877, r = 4, s = 128**

With the same notation as above we construct the linear layer using the following code

$$C_{128\text{-}4} = \mathbf{Extension}(\mathbf{Shorten}(\mathbf{BCH}(3741, 173), [2]), E_{128\text{-}4}), \quad E_{128\text{-}4} \in \mathbb{F}_2^{1877\times 15}.$$

We use the base permutation

$$\pi = g(19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71,$$
$$73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 84)$$

and, given the block counter $i$ we define

$$(P_{0,i}, P_{1,i}, P_{2,i}, P_{3,i}, P_{4,i}) = \left(\pi^i, \pi^i, \pi^i, \pi^i, \pi^i\right).$$

### Dasta **256-6: n = 703, r = 6, s = 256**

With the same notation as above we construct the linear layer using the following code

$$C_{256\text{-}6} = \mathbf{Extension}(\mathbf{Shorten}(\mathbf{BCH}(1365, 125), [4]), E_{256\text{-}6}), \quad E_{256\text{-}6} \in \mathbb{F}_2^{703\times 45}.$$

We use two base permutations

$$\pi_0 = g(2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 64)$$
$$\pi_1 = g(19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 49)$$

and, given the block counter $i = \left\lceil \sqrt{D} \right\rceil i_1 + i_0 = \left\lceil \sqrt[3]{D} \right\rceil^2 j_2 + \left\lceil \sqrt[3]{D} \right\rceil j_1 + j_0$ we define

$$(P_{0,i}, \ldots, P_{6,i}) = \left(\pi_0^{i_0}, \pi_0^{i_1}, \pi_1^{j_0}, \pi_1^{j_1}, \pi_1^{j_2}, \pi_0^{i_0}, \pi_0^{i_1}\right).$$

DASTA **256-5: n = 3545, r = 5, s = 256**

With the same notation as above we construct the linear layer using the following code

$$C_{256\text{-}5} = \mathbf{Extension}(\mathbf{Shorten}(\mathbf{BCH}(8191, 621), [1110]), E_{256\text{-}5}), \quad E_{256\text{-}5} \in \mathbb{F}_2^{3545\times 9}.$$

We use the base permutation

$$\begin{aligned}\pi = g(&71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127,\\ &131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 85)\end{aligned}$$

and, given the block counter $i$ we define

$$(P_{0,i}, P_{1,i}, P_{2,i}, P_{3,i}, P_{4,i}, P_{5,i}) = \left(\pi^i, \pi^i, \pi^i, \pi^i, \pi^i, \pi^i\right).$$

**Security claims** We claim that the advantage of any attacker $A$ spending $T$ operations and querying at most $D$ key stream blocks

$$c_j = B_{i_j}(k)$$

with counters $0 \leq i_j < D$ of her choice, in distinguishing the output of DASTA from a truly random sequence is bounded by

$$\mathrm{Adv}(A) \leq \frac{T}{2^s}.$$

where $s$ is the security parameter and

$$D = \frac{2^{s/2}}{n}$$

is the data limit.

**Alternative Choices for the Linear Layer** Our choice of codes is probably among the most straight forward ways to achieve the required results and thus in particular a good choice for a proof of concept. BCH codes are well understood and their main advantage for our purpose is the explicitly computable minimal distance. That said, we like to mention that other choices for the linear layers, based on codes with sufficient distance are certainly possible. Binary Goppa codes, as used also in the McEliece crypto system [McE78], could be a promising alternative. They also have an explicitly computable minimal distance and are hard to distinguish from random codes. The later suggests that using Goppa codes does not introduce structural weaknesses. Actually random codes of the required dimensions have, for most dimensions, with very high probability a minimal distance that is sufficient. While it is in general hard to compute the minimal distance, and indeed already hard for some of our DASTA versions, it might be of interest, depending on the application and platform, to check for variants that allow for an improved performance.

## 3.2 Overview of the Security Analysis

The change from the random linear layers in original RASTA to our instance, where the linear layers only differ by a sequence of bit permutations, has of course significant impact on the security analysis.

**Linear Cryptanalysis** For linear cryptanalysis, the analysis in the original Rasta paper was based on estimating the probability that a randomly chosen matrix would allow a two round trail with high correlation, an analysis similar to LowMC. Clearly, this analysis is not applicable to Dasta, as the choice of the linear layer is not random and actually partially attacker controlled. Instead we base our analysis in Section 5 on the branch number of the linear layer and a more detailed analysis of the non-linear layer, that might be of independent interest. For each of the original parameter sets, we were able to generate linear layers with a sufficiently large branch number, based on known binary codes or minor modification of those. As the branch number is invariant under bit permutations of the input and output values, the choice of bit permutations as the source of variation for the different rounds does not complicate the analysis.

Note that, compared to e.g. block ciphers, the situation for Dasta (and Rasta) is quite different with respect to the applicability of linear cryptanalysis. For a block cipher, by varying the message, an attacker can continuously generate samples for a fixed linear approximation. For Dasta, as the key is fixed and the linear layers vary, the attacker will get changing linear approximations evaluated at the same point. Thus, we do not see a concrete way to mount attacks based on linear approximations.

**Truncated Differential Analysis** One attack vector, that was not detailed for the original Rasta cipher, was the existence of probability one truncated differentials. Those naturally translate to key stream bits (or linear combinations of those) that would not depend on all key bits. Thus, the existence of such truncated differentials with probability one might lead to guess-and-determine key recovery attacks. For Rasta truncated differentials are highly unlikely due to the random selection of the linear layers. Using the notation of subspace trails [GRR16] and the results of Section 6, we are able to prove that for our choice of linear layers there does not exist any non-trivial subspace trail for Dasta.

We experimentally verified, using a few block counters for all versions, that any output bit depends on every input bit already after two rounds of Dasta. In particular, we experimentally could not find sufficiently large subspace trails to mount attacks even on Dasta reduced to two rounds.

**Linearization Attacks** Our analysis of the complexity of linearization attacks depends on heuristics. This was already the case for original Rasta. Here, in contrast to linear attacks, additional assumptions on the distribution of monomials in the ANF of the key stream were used to estimate the security level. We follow the same approach here and additionally back up the heuristics with significant experimental results.

As a result, the estimation of the resistance of Dasta against linearization attacks is the same as of Rasta. Moreover, the original security estimates in [Dob+18] are very conservative as it assumes linear running time of the linear algebra steps involved. Using more realistic assumptions, i.e. sub-cubic running time, on the complexity of the linear algebra involved, we estimate the maximal number of rounds that can actually be attacked in Table 2 given the estimate in Equation 13 given in Section 7.

**Similarity Attacks** As the computations for two different counters only differ by the bit permutations used, an attacker could try to gain information on the key by comparing the output for different counter values. Thus, not only has any instance to be secure, but the instances should also not be related in a way that an attacker could exploit. To make this more concrete, consider the simplest case where two instances only differ in a transposition of the initial input, i.e. the key. In this case, the attacker could easily learn if the corresponding key bits are the same (in which case the key stream blocks are identical) or different (in which case the key stream blocks differ with high probability). In order to avoid those attacks, that we call similarity attacks, we have to ensure that different

**Table 2:** Linearization attack

| Dasta version | rounds | complexity |
|---|---|---|
| 80-6 | 2 | $2^{62.8}$ |
| 80-4 | 2 | $2^{68.3}$ |
| 128-6 | 3 | $2^{123.9}$ |
| 128-5 | 2 | $2^{74.8}$ |
| 128-4 | 2 | $2^{92.2}$ |
| 256-6 | 4 | $2^{253.2}$ |
| 256-5 | 3 | $2^{187}$ |

counters lead to significantly different outputs. The details of our considerations can be found in Section 4.2.

As we defined the set of the bit permutations to be very strong against this type of attack, we do not see how to attack any of the proposed variants even for two rounds.

Dasta **256-6** Finally, we like to point out that the instance Dasta 256-6 requires some additional attention. Here, in contrast to the other instances, by choosing counters at distance $\left\lceil \sqrt{D} \right\rceil$ the first bit permutation and thus the entire first round are identical for all those counters. In particular with respect to linear attacks and linearization, one could then consider the output of the first round as the actual input and get a version that is de facto reduced by one round. However, this is compensated by the reduced data that will be available ($\sqrt{D}$ instead of $D$ blocks) when only counters at distance $\left\lceil \sqrt{D} \right\rceil$ are considered.

### 3.3 Implementation

We implemented Rasta and Dasta as normal stream ciphers(offline key stream generation) as well as in the FHE setting with the HElib [HS13]. It turns out that Dasta is about 200 to 400 times faster than Rasta in the offline settings(see Table 4) since the very costly overhead of executing SHAKE256 as XOF to generate the random matrices with the algorithm of Randall [Ran93] is not needed. In comparison, for Dasta just the powers of bit permutations have to be computed which is much more efficient.

In the FHE setting Dasta achieves runtime improvements compared to Rasta of about 15% to 20% (see Table 4). Here the (offline) generation of the random matrices is negligible since the block evaluation under FHE dominated the execution time. We evaluate the bit permutation and matrix multiplication in Dasta as one linear layer. Since we use for each version less than $r + 1$ different bit permutation, we use the same linear layer in one block evaluation twice. This leads to the runtime improvements.

For Dasta with an 80 bit security level, the version 80-4 is more efficient than 80-6 in terms of latency and throughput in our HElib implementation. In contrast, for Dasta with a 128 bit security level, the version with more rounds (128-6) and a smaller state is more efficient. Remember, that the different number of rounds for the same security level are the trade-offs between the number of AND and XOR operations. For an implementation in another FHE library or in a MPC setting, the performance of the different versions of Dasta can behave different, so that the definition of multiple versions for one security level is useful.

## 4 Bit Permutation Analysis

In this section we justify our choice for the bit permutation layer. Recall that the bit-permutations used in Dasta are powers of a small set of base permutations, that is for a

**Table 3:** Performance comparison of Rasta and Dasta in the offline key stream generation on Intel(R) Xeon(R) CPU E3-1230 v6 @ 3.50GHz with 32GB of RAM.

| | Throughput in kB/s | |
|---|---|---|
| Version | Rasta | Dasta |
| 80-6 | 6.6 | 1291.3 |
| 80-4 | 6.3 | 1615.6 |
| 128-6 | 4.2 | 1325.5 |
| 128-5 | 3.3 | 1458.3 |

**Table 4:** Performance comparison of Rasta and Dasta in the homomorphic key stream generation with the HElib [HS13] on Intel(R) Xeon(R) CPU E3-1230 v6 @ 3.50GHz with 32GB of RAM.

| | Encryption of one Key Stream Block in Seconds | | Milliseconds per Encrypted Bit (amortized) | |
|---|---|---|---|---|
| Version | Rasta | Dasta | Rasta | Dasta |
| 80-6 | 10.83 | 8.47 | 49.45 | 38.68 |
| 80-4 | 7.44 | 6.49 | 22.75 | 19.85 |
| 128-6 | 15.41 | 12.07 | 43.90 | 34.39 |
| 128-5 | 32.05 | 27.06 | 61.05 | 51.54 |

given round, the set of allowed bit permutations can be described as

$$\{\pi^i\}_{i\leq l} := \{\pi^i \mid 0 \leq i < l\}$$

where $l \in \mathbb{N}$ is bounded by the data complexity. Recall furthermore, that the permutation $\pi$ we are using has a specified structure, which is parameterized by the cycle lengths.

That choice is motivated by the results in this section were we analyze the success probability of the similarity attack sketched above.

#### Similarity for Two Bit Permutations

If we look at the generation of two different key stream blocks, the only differences in the generation algorithm are the used bit permutations since the key, linear layer, and non-linear layer are fixed.

In order to avoid that two block counters result in the same key stream block, we have to ensure that any two possible bit permutations are sufficiently different.

We start by looking at the probability that two different bit permutations $\pi, \pi' \in S_n$ map an $x \in \mathbb{F}_2^n$ to the same value, i. e.

$$p = \Pr_{x\in\mathbb{F}_2^n}[\pi(x) = \pi'(x)]$$

where $\pi$ corresponds to a bit permutation in block one and $\pi'$ to a bit permutation in block two.

If we denote by $S$ the set of possible bit permutations, computing the difference between two permutations is not sufficient, but we rather have to upper bound this probability when $\pi, \pi'$ range within a set $S$ of allowed permutations. That is we want to bound

$$\min_{\pi,\pi'\in S} \Pr_{x\in\mathbb{F}_2^n}[\pi(x) = \pi'(x)].$$

At this point, our choice of $S$ as $\{\pi^i\}_{i\leq l}$ turns out very handy.

To compute the probability for $\pi_0, \pi_1 \in \{\pi^i\}_{i\leq l}$, we have a look at the identity $\pi_0(x) = \pi_1(x)$, or equivalently at $\pi_1^{-1} \circ \pi_0(x) = x$.

Taking into account that $\pi_0 = \pi^i$ and $\pi_1 = \pi^j$ for some $0 \leq i, j < l$ where $i \neq j$, we can conclude

$$\pi_0(x) = \pi_1(x) \Leftrightarrow \pi^{|i-j|}(x) = x.$$

Thus, instead of comparing arbitrary pairs of permutations, we are left with studying the number of fixed points for powers of a given bit permutation. This reduces the complexity of the problem significantly. Moreover, this property is obviously related to the cycle structure of $\pi$. Again, this is reflected in our definition of the bit permutations based on special choices for the cycle length.

## 4.1 Lower Bounding the Difference

The probability for an $x \in \mathbb{F}_2^n$ to be a fixed point for a bit permutation depends only on the cycle structure and is shown in the following lemma.

**Lemma 1.** *Let $\pi \in S_n$ be a permutation with $m$ cycles, then we have*

$$\Pr_{x\in\mathbb{F}_2^n}[\pi(x) = x] = 2^{-(n-m)}.$$

*Proof.* It holds that $\pi(x) = x$ if and only all elements in one cycle are equal, i. e. they are all zeros or all ones. We have in total $2^m$ possible inputs which fulfill this property and $2^n$ elements in $\mathbb{F}_2^2$. This leads to an overall probability of $2^m/2^n = 2^{-(n-m)}$. □

To simplify the notation, we introduce the following values.

**Definition 1** (Measure of Similarity of Bit Permutations)**.** Let $\pi \in S_n$ be a permutation with $m$ cycles. Then we define

$$\psi(\pi) = n - m.$$

Furthermore, for a set $S \subseteq S_n$ of permutations, we define

$$\Psi(S) \coloneqq \min\{\psi(s) \mid s \in S, s \neq \mathrm{id}\}.$$

With the result of Lemma 1 and the notation just introduced, we conclude that for all $\pi_0, \pi_1 \in \{\pi^i\}_{i\leq l}, \pi_0 \neq \pi_1$ it holds that

$$p = \Pr_{x\in\mathbb{F}_2^n}[\pi_0(x) = \pi_1(x)] = 2^{-\psi(\pi_1^{-1}\circ\pi_0)} \leq 2^{-\Psi(\{\pi^i\}_{i\leq l})}.$$

In the next step we need an efficient way to compute the $\Psi(\{\pi^i\}_{i\leq l})$ above, which we present in Algorithm 2.

But first, we derive in Lemma 2 the observation, that computing $\psi(\pi^i)$ just depends on $i$ and the cycle lengths of $\pi$.

**Lemma 2.** *Let $\pi \in S_n$ be a permutation with $t$ cycles of the lengths $c_0, \ldots, c_{t-1}$ where $n = \sum_{j=0}^{t-1} c_j$, then for all $i \in \mathbb{N}$ it holds that*

$$\psi(\pi^i) = \sum_{m=0}^{t-1} (c_m - \gcd(i, c_m)).$$

*Proof.* For a power of a permutation $\pi$, the cycles are independent of each other and can be considered as an own permutation. Let $\pi_m \in S_{c_m}$ be an arbitrary but fixed cycle of $\pi$ with $\pi_m = (x_0\, x_1 \cdots x_{c_m-1})$. For $(\pi_m)^i$, $x_0$ is mapped to $x_i$, $x_1$ is mapped to $x_{1+i}$ and so on with all indices modulo $c_m$. Keep in mind that a cycle of length $b$ has $b$ different representations in the cycle notation. Each $x_k$ is now part of a cycle $(x_k\ x_{k+i}\ \cdots\ x_{k+(b-1)\cdot i})$, so that $b$ is the smallest positive integer that fulfills

$$\begin{aligned} & k + b \cdot i \equiv k \bmod c_m \\ \Leftrightarrow \quad & b \cdot i \equiv 0 \bmod c_m \\ \Leftrightarrow \quad & c_m \mid b \cdot i \end{aligned}$$

This implies that $b = \frac{\operatorname{lcm}(i,c_m)}{i} = \frac{c_m}{\gcd(i,c_m)}$. Thus $(\pi_m)^i$ has $\frac{c_m}{b}$ cycles of length $b$. This leads to

$$\begin{aligned} \psi(\pi_m^i) &= \frac{c_m}{b} \cdot (b-1) \\ &= \gcd(i,c_m) \cdot \left( \frac{c_m}{\gcd(i,c_m)} - 1 \right) \\ &= c_m - \gcd(i,c_m). \end{aligned}$$

□

Lemma 2 shows, that a cycle of prime length $p$ in permutation $\pi$ keeps the length $p$ in $\pi^i$ for $p \nmid i$ and decomposes in fixed points for $p \mid i$. Therefore, it seems that prime cycle lengths are a good choice.

As the naive computation of

$$\Psi(\{\pi^i\}_{i\le l}) = \min\{\psi(\pi^i) \mid 1 \le i \le l\}$$

by iterating over all permutations in the set, is too slow for our purpose, we developed Algorithm 2, that is much more efficient especially when many cycle lengths are prime. On a standard PC, the algorithm terminates for all versions of Dasta in less than one minute. The correctness of the algorithm is shown in Theorem 1.

**Theorem 1.** *Algorithm 2 computes* $\Psi(\{\pi^i\}_{i\le l})$.

*Proof.* Let $\pi \in S_n$ be a permutation with the cycle lengths $c_0, \ldots, c_{t-1}$. Then we have

$$\begin{aligned} \Psi(\{\pi^i\}_{i\le l}) &= \min\left\{\ \psi(\pi^i) \mid 1 \le i < l\ \right\} \\ &= \min\left\{ \sum_{j=0}^{t-1} (c_j - \gcd(c_j, i)) \;\middle|\; 1 \le i < l \right\} \\ &= \sum_{j=0}^{m-1} c_j - \max\left\{ \sum_{j=0}^{t-1} \gcd(c_j, i) \;\middle|\; 1 \le i < l \right\}. \end{aligned}$$

In the following we determine the value of $M \coloneqq \max\left\{\ \sum_{j=0}^{t-1} \gcd(c_j,i) \mid 1 \le i < l\ \right\}$. It is easy to see that

$$\{(\gcd(c_0,i), \ldots, \gcd(c_{t-1},i)) \mid \forall i \in \mathbb{N}\} = D_0 \times \cdots \times D_{t-1}$$

for the sets $D_j, j \in [t]$ as defined in line 3. In the next step we will show that

$$T \coloneqq \{(d_0, \ldots, d_{t-1}) \in D_0 \times \cdots \times D_{t-1} \mid e < l \wedge \forall i \in [t] \colon\ \gcd(e, d_i) = d_i\}$$

**Algorithm 2** Computation of $\Psi$

**Input:** cycle lengths $C = \{c_0, \cdots, c_{t-1}\}$, cardinality $l$
**Output:** $\Psi(\{\pi^i\}_{i \leq l})$

$n \leftarrow \sum_{i=0}^{t-1} c_i$
**for** $j \leftarrow 0$ **to** $t-1$ **do**
    $D_j \leftarrow \left\{k \in \mathbb{N} \;\middle|\; k \mid c_j\right\}$
**return** $n - \textsc{Rec}(\emptyset, 0)$

**function** $\textsc{Rec}(\text{List}, i)$
    **if** $\text{lcm}(\text{List}) < l$ and $\gcd(x, lcm(\text{List})) = x,\ \forall x \in \text{List}$ **then**
        **if** $i \geq t$ **then**
            **return** $\sum_{d \in \text{List}} d$
        **else**
            $m \leftarrow 0$
            **for all** $d \in D_i$ **do**
                $v \leftarrow \textsc{Rec}(\text{List} \cup \{d\}, i+1)$
                **if** $v > m$ **then**
                    $m \leftarrow v$
            **return** $m$
    **else**
        **return** $0$

where $e = \text{lcm}(d_0, \ldots, d_{t-1})$, is equal to

$$T' \coloneqq \{(\gcd(c_0, i), \ldots, \gcd(c_{t-1}, i)) \mid 1 \leq i < l\}\,.$$

It is easy to see that $T \subseteq T'$. Now let $i \in \mathbb{N}$ where $1 \leq i < l$ be arbitrary chosen but fixed. Then we have $(d_0, \ldots, d_{t-1}) = (\gcd(c_0, i), \ldots, \gcd(c_{t-1}, i)) \in D_0 \times \cdots \times D_{t-1}$ and $\text{lcm}(d_0, \ldots, d_{t-1}) \cdot x = i$ where $x \in \mathbb{N}$. Further holds $\gcd(d_i, \text{lcm}(d_0, \ldots, d_{t-1})) = d_i$ so that $T' \subseteq T$ and therefore $T = T'$.

This leads to

$$M = \max\left\{\sum_{d \in t} d \mid t \in T\right\}$$

which is calculated in the function REC of the algorithm. To speed up the algorithm we make use of the identity $\text{lcm}(x, y) \leq \text{lcm}(x, y, z),\ x, y, z \in \mathbb{N}$. Overall the algorithm computes $n - M$.

□

## 4.2 Similarity Attacks

In the last section, we investigated the probabilities for two bit permutations to map an $x \in \mathbb{F}_2^n$ to the same value. When an adversary can find two counters where this happens for all bit permutations in all rounds, the resulting two blocks will be equal and the adversary can learn something about the key. Our –very conservative– aim is to bound the resulting probability (using a strong independence assumption) over all rounds by at most $2^{-n}$ where $n$ is the block length.

The previous analysis presumes that the input of the bit permutation is uniformly drawn. For the bit permutation in the first round, this is clearly the case, since the input is the (uniformly drawn) key. In the later rounds, we assume that the states behave like uniformly drawn to simplify our analysis. We name the type of attacks which are based on similar bit permutations *similarity attacks*.

Let us consider the general case of two arbitrary Dasta executions using bit-permutations $I_s = (P_{0,s}, P_{1,s}, \cdots, P_{r,s})$ and $I_t = (P_{0,t}, P_{1,t}, \cdots, P_{r,t})$ for block counters $s$ and $t$. Further let $\Delta_i = P_{i,s}^{-1} \circ P_{i,t},\ i \in [r+1]$ be the difference of these permutations.

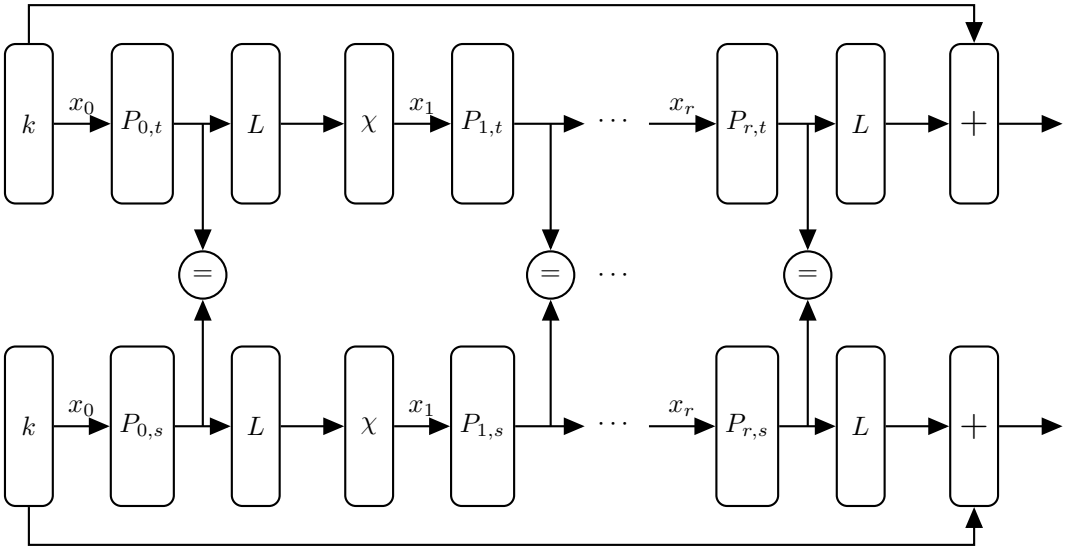


**Figure 4:** Similarity Attack on Dasta.

If the state of the two blocks is equal after each application of the bit permutations, that is if

$$\Delta_i(x_i) = x_i \ \forall i \in [r+1] \tag{4}$$

then it holds that the two blocks are identical, i.e. $B_s(k) = B_t(k)$ (see Figure 4).

Obviously, $B_s(k) = B_t(k)$ does not imply equality after each bit permutation for all rounds. However, this situation makes the attack based on the ideas here significantly more complicated and we focus on the former case only.

Let us denote the probability of Equation (4) happening for independent states and all rounds by $p$. Following Lemma 1, the probability $p$ is given by

$$p = 2^{-(\psi(\Delta_0)+\psi(\Delta_1)+\cdots+\psi(\Delta_r))}.$$

To give an upper bound for the probability of a set of instances we will simply bound each component in the equation above by the corresponding minimum $\Psi$ over the instance. By slight abuse of notation, we will denote the resulting bound for an entire instance $S$ by $\Psi(S)$.

## 4.3 Design Choice for Dasta

For simplicity, we explain the approach to compute the desired bounds for the explicit example of Dasta 80-6. Here, the bit permutation along the rounds are chosen as

$$(P_{0,t}, \ldots, P_{6,t}) = \left(\pi^t, \pi^{t_0}, \pi^{t_1}, \pi^{t_0}, \pi^{t_1}, \pi^{t_0}, \pi^{t_1}\right).$$

with

$$t = \left\lceil \sqrt{D} \right\rceil t_1 + t_0 \text{ with } 0 \leq t_0, t_1 < \sqrt{D}$$

as the block counter. The permutation is given by

$$\pi = g(5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 27).$$

Now with the use of Algorithm 2 we are able to easily compute

$$\Psi(\{\pi^t\}_{t\leq D}) = 38 \text{ and } \Psi(\{\pi^t\}_{t\leq\sqrt{D}}) = 114.$$

For the entire set this implies a difference of

$$\Psi(\text{DASTA80-6}) \leq \Psi(\{\pi^t\}_{t\leq D}) + 3\cdot\Psi(\{\pi^t\}_{t\leq\sqrt{D}}) = 38 + 3\cdot 114 = 380$$

Here we have taken into account that the attacker is able to choose counters

$$t = \left\lceil\sqrt{D}\right\rceil t_1 + t_0 \text{ and } s = \left\lceil\sqrt{D}\right\rceil s_1 + s_0$$

such that $t_0 = s_0$ or $t_1 = s_1$ but not both by counting $\Psi(\{\pi^t\}_{t\leq\sqrt{D}})$ only 3 times even so it appears in six bitpermutations.

**Table 5:** $\Psi$ for DASTA versions.

| DASTA version | $\Psi(S)$ |
|---|---|
| 80-6 | 380 |
| 80-4 | 500 |
| 128-6 | 438 |
| 128-5 | 724 |
| 128-4 | 5125 |
| 256-6 | 596 |
| 256-5 | 6156 |

The values for $\Psi$ of all DASTA versions can be found in Table 5. As can be seen all values of $\Psi$ are well above the block size implying that the probability considered in the similarity attack is below $2^{-n}$. We clearly simplified by assuming independence. However, as the security claim is significantly below the block size, we feel that our construction provides a large security margin against those attacks.

We like to add that more general versions of the attack are imaginable. Instead of asking for equal output blocks, permuted, rotated, or otherwise related blocks could be treated. Due to the choice of the linear layers, we do not think those attacks pose a threat. However, giving good arguments why they are impossible seems very challenging.

## 4.4 Alternative Construction for the Bit-Permutations

Finally, we like to mention another possibility for choosing the bit permutation as shown in Figure 5. In this approach the state of DASTA is split into $t$ parts of the same length, inspired by the S-box layer design of a block cipher.

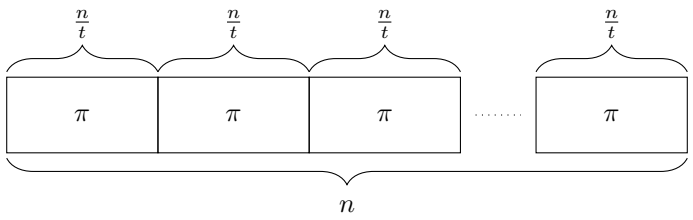


**Figure 5:** Subpermutations

The advantage of this approach is, that it is even more simple to analyze (as two different instances differ in each of the parts) and, with a suitable choice of the base permutations, potentially easier to implement. However, the bounds on the similarity are less strong and we therefore decided to not deploy this approach for DASTA.

# 5 Linear Properties

Recall that $s$ denotes the security level and $D = 2^{s/2}/n$ is the data limit. The data complexity for linear cryptanalysis using a linear approximation with correlation $\delta$ is about $\delta^{-2}$. Thus, given a data limit of $D$, linear approximations with a correlation of $D^{-1/2} = \frac{2^{-s/4}}{\sqrt{n}}$ or lower are hard to exploit and usually only if many of those are available.

Consequently, and slightly more conservative, the designers of Rasta chose the parameters so that no linear trail with correlation $\delta > 2^{-s/4}$ exists. This bound was derived by bounding the correlation of any two round trail. This in turn was based on two ingredients, the probability of matching an input and output mask through a linear layer and bounding the correlation of an approximation of the non-linear layer $\chi$. More precisely, when denoting two rounds of Rasta by

$$\chi \circ L \circ \chi.$$

and considering a fixed two round trail given by

$$\alpha' \xrightarrow{\chi} \alpha \xrightarrow{L} \beta \xrightarrow{\chi} \beta',$$

where $\alpha$ is the output mask of the first non-linear layer and $\beta$ is the input mask of the second non-linear layer. Its correlation is given by

$$C_\chi(\alpha' \to \alpha) \cdot C_L(\alpha \to \beta) \cdot C_\chi(\beta \to \beta').$$

In order to have non-zero correlations, it holds that

$$\alpha = L^\top(\beta),$$

as $L$ is linear and thus

$$C_L(\alpha \to \beta) = \begin{cases} 1 & \text{if } \alpha = L^\top(\beta) \\ 0 & \text{else} \end{cases}.$$

**Arguments for** Rasta

As shown in [Dob+18] the probability for a random invertible binary matrix $L$ to allow a certain fixed linear characteristic is

$$\frac{\prod_{i=1}^{n-1}(2^n - 2^i)}{\prod_{i=0}^{n-1}(2^n - 2^i)} = (2^n - 1)^{-1}.$$

It remains to bound the correlation of $\chi$. For the given output mask, the (in general tight) bound given in [Dae95; KLT15], can be used to bound the correlation $C_\chi(\alpha' \to \alpha)$ by

$$\max_{\alpha'} |C_\chi(\alpha' \to \alpha)| \leq 2^{-\lceil \mathbf{hw}(\alpha)/2 \rceil}.$$

However, the case of bounding the correlation $C_\chi(\beta \to \beta')$ for a given weight of an input mask $\beta$ was not studied in depth before. Thus, no tight bound is known in this case. Instead, Dobraunig *et al.* [Dob+18] used the bound

$$\max_{\beta'} |C_\chi(\beta \to \beta')| \leq 2^{-\lceil \mathbf{hw}(\beta)/6 \rceil}$$

to bound the correlation of the two round trail by

$$|C_\chi(\alpha' \to \alpha) \cdot C_L(\alpha \to \beta) \cdot C_\chi(\beta \to \beta')| \leq 2^{-\lceil \mathbf{hw}(\alpha)/2 \rceil - \lceil \mathbf{hw}(\beta)/6 \rceil}$$

in the case that the trail has non-zero correlation, i.e. is valid through the linear layer.

The block sizes of Rasta are then chosen such that the probability that a suitable trail for all rounds exists is at most $2^{-s}$.

**Arguments for** Dasta

For Dasta the security against linear cryptanalysis cannot rely on the probability of matching input and output mask, as the liner layer is fixed. Instead we rely on the linear branch number of the fixed linear layer for our argumentation. Given two rounds of Dasta as

$$\chi \circ P_{j,i} \circ L \circ \chi,$$

first note that the linear branch number of $P_{j,i} \circ L$ and $L$ are identical and thus the bit permutation $P_{i,j}$ can be ignored in the following.

Using the linear branch number $b$ of $L$, the minimum joint weight of $\alpha$ and $\beta = \left(L^\top\right)^{-1}(\alpha)$ can be lower bounded as

$$\mathbf{hw}(\alpha) + \mathbf{hw}(\beta) \geq b,$$

and using the bounds from above, the maximal correlation could be bounded by

$$|C_\chi(\alpha' \to \alpha) \cdot C_L(\alpha \to \beta) \cdot C_\chi(\beta \to \beta')| \leq 2^{-\left(\left\lceil \frac{b-2}{6} \right\rceil + 1\right)}.$$

In order to ensure that for Dasta no exploitable linear trail with correlation $c \geq 2^{-s/4}$ exist over $r = 2t$ rounds, the branch number $b$ has to be sufficiently large. More precisely,

$$2^{-\left(\left\lceil \frac{b-2}{6} \right\rceil + 1\right) \cdot t} \leq 2^{-(s/4)}.$$

This is equivalent to the following condition

$$b \quad \geq \quad 6 \cdot \left\lceil \frac{s}{4t} \right\rceil - 9. \tag{5}$$

The corresponding results for all parameters of Dasta are shown in Table 6 as $b'_{\text{min}}$. Unfortunately, in several cases, $b'_{\text{min}}$, the minimal required branch number following the known results, is higher than the branch numbers B of our constructions for the corresponding parameters (see Section 3.1).

In order to overcome this problem, we next derive a better bound in Theorem 2 for the correlation of $\chi$ with a given weight of the input mask, which might be of independent interest. With this result, we compute the smaller $b_{\text{min}}$, which justifies our used linear layers.

**Table 6:** Linear branch numbers of Dasta. $b_{min}$ corresponds to Eq. 6, while $b'_{min}$ corresponds to the previous known bound in Eq. 5

| Dasta version | block size | $b_{\text{min}}$ | $b'_{\text{min}}$ | $B$ |
|---|---|---|---|---|
| 80-6 | 219 | 23 | 33 | 45 |
| 80-4 | 327 | 35 | 51 | 42 |
| 128-6 | 351 | 39 | 57 | 45 |
| 128-5 | 525 | 59 | 87 | 85 |
| 128-4 | 1877 | 59 | 87 | 173 |
| 256-6 | 703 | 83 | 123 | 125 |
| 256-5 | 3545 | 123 | 183 | 621 |

## 5.1 Improved Analysis of $\chi$

In this section we take a closer look at the bounds on the absolute linear correlation of the non-linear layer of Dasta, i. e. $\chi$. Especially, we derive new bounds for the case of the

correlation $C_\chi(\alpha \to \beta)$ as

$$\max_\beta |C_\chi(\alpha \to \beta)| \leq 2^{-\lceil \mathbf{hw}(\alpha)/4 \rceil}$$

involving the weight of an input mask $\alpha$. Those bounds are tight in the sense that there exist input masks $\alpha$ for which we have equality.

Based on this improvement, the main result of this section is the following theorem.

**Theorem 2.** *Let $b$ be the linear branch number of the linear layer $L$ of* Dasta*. Then*

$$C_{2,max} \leq 2^{-\left(\left\lceil \frac{b-2}{4} \right\rceil + 1\right)}$$

*where*

$$C_{2,max} \coloneqq \max_{\alpha,\beta,\gamma \in \mathbb{F}_2^n \setminus \{0\}} |C_{L\circ\chi}(\alpha \to \beta) C_\chi(\beta \to \gamma)|$$

*is the maximal absolute correlation of a linear characteristic for two rounds.*

The maximal correlation of a linear characteristic for $r$ rounds of Dasta can thus be upper-bounded by $c = (C_{2,\max})^t$ with $t = \left\lfloor \frac{r}{2} \right\rfloor$. To be resistant against linear approximations we claim $c \leq 2^{-(s/4)}$ for the correlation $c$ of any linear characteristic. This leads to an improved lower bound on the branch number

$$b \geq 4 \cdot \left\lceil \frac{s}{4t} \right\rceil - 5. \tag{6}$$

Most importantly, for all parameters this bound is lower than the branch number of our constructions and thus justifies the instantiations of Dasta with the linear layers as given in Section 2.1.

#### Known Results

Before proving Theorem 2, we start by recalling and generalizing known results. The main ideas were already presented in the thesis of J. Daemen [Dae95]. As the non-linear layer is similar to the round function of the block cipher Simon [Bea+13], we also generalize some cryptanalytic results from [KLT15]. We have

$$\begin{aligned} \chi(x) &= x + S^{-2}(x) + S^{-1}(x) \odot S^{-2}(x) \\ &= S^{-2}(x + S^2(x) + x \odot S^1(x)). \end{aligned}$$

If we substitute $\chi'(x) = x + S^2(x) + x \odot S^1(x)$, we can write the correlation as

$$C_\chi(\alpha \to \beta) = C_{\chi'}(S^{-2}(\alpha) \to \beta)$$

since for every linear mapping $G\colon \mathbb{F}_2^n \to \mathbb{F}_2^n$ and any function $F\colon \mathbb{F}_2^n \to \mathbb{F}_2^n$ it holds that $C_F(\alpha \to \beta) = C_{F\circ G}(G^\top(\alpha) \to \beta)$. Further we can write the correlation of $F'(x) = F(x) + S^a(x)$ as $C_{F'}(\alpha \to \beta) = C_F(\alpha + S^{-a}(\beta) \to \beta)$. This leads to

$$C_\chi(\alpha \to \beta) = C_f\Big(\big(S^{-2}(\alpha) + \beta + S^{-2}(\beta)\big) \to \beta\Big). \tag{7}$$

where

$$f(x) = x \odot S^1(x).$$

The transformation in Equation (7) shows that it is sufficient to determine the correlation of $f$. This was done in [KLT15] for an even block length $n$. Here, we adapt the work to an odd $n$.

First, we define a function $\theta : \mathbb{F}_2^n \to \mathbb{N}$ as follows: Let $c_0, \ldots, c_m$ be the lengths of all blocks of consecutive ones of an input $x$, i. e.

$$x = 0 \ldots 0 \underbrace{1 \ldots 1}_{c_0} 0 \ldots 0 \underbrace{1 \ldots 1}_{c_1} 0 \ldots \ldots 0 \underbrace{1 \ldots 1}_{c_m} 0 \ldots 0.$$

For $x = \mathbf{1}$ the ones are not covered with zeros, but in this case we have $m = 0$ and $c_0 = n$. If $x_0 = x_{n-1} = 1$, the ones belong to the same block. Now we define

$$\theta(x) := \sum_{i=0}^{m} \left\lceil \frac{c_i}{2} \right\rceil .$$

For example: $\theta(1110111) = 3$ and $\theta(1001100) = 2$.

With this definition we can state the next theorem. The proof closely follows the approach used in [KLT15] and can be found in Appendix A.1 mainly for completeness. We also point out some flaws in the argumentation given in [KLT15].

**Theorem 3.** *Let $f(x) = x \odot S^1(x)$, where $x \in \mathbb{F}_2^n$ and $n$ is odd. Then*

$$|C_f(\alpha \to \beta)| = \begin{cases} 2^{-(n+1)/2} & \text{if } \beta = \mathbf{1} \text{ and } \boldsymbol{hw}(\alpha) \equiv 1 \bmod 2 \\ 2^{-\theta(\beta)} & \text{if } \beta \neq \mathbf{1} \text{ and } \alpha \in U_\beta^\perp \\ 0 & \text{else} \end{cases}$$

*where $U_\beta = \{y \mid \beta \odot S^1(y) + S^{-1}(\beta \odot y) = \mathbf{0}\}$*

*Proof.* See Appendix A.1. □

This theorem already hints at why it is easier to bound the correlation for a given output mask: there are only two possible correlation values for any given output mask. This is not the case for a given input mask.

### Improvements

Theorem 3 shows that for a fixed output mask $\beta$ the correlation can only be 0 or $2^{-\theta(\beta)}$, depending on the input mask $\alpha$. Therefore the absolute correlation can easily be upper-bounded for a fixed $\beta$ by $2^{-\theta(\beta)}$. To upper-bound the correlation for a fixed $\alpha$, we have to find the $\beta$ so that $2^{-\theta(\beta)}$ is maximal and $\alpha \in U_\beta$ (resp. $\mathbf{hw}(\alpha) \equiv 1 \bmod 2$ for $\beta = \mathbf{1}$). In the following lemma we derive conditions at the bit level for an $\alpha$ to be in $U_\beta$.

**Lemma 3.** *Let $\alpha, \beta \in \mathbb{F}_2^n$ and $U_\beta = \{y \mid \beta \odot S^1(y) + S^{-1}(\beta \odot y) = \mathbf{0}\}$, then $\alpha \in U_\beta^\perp$ if and only if*

*(i)* $\alpha_i = 1 \Rightarrow (\beta_i = 1 \vee \beta_{i+1} = 1) \quad \forall i \in [n]$ *and*

*(ii)* $\alpha_{i-1} + \alpha_{i+1} + \cdots + \alpha_{i+l-1} = 0$ *for any block of $l$ ones in $\beta$ with $\beta_i = \beta_{i+1} = \cdots = \beta_{i+l-1} = 1$ and $\beta_{i-1} = \beta_{i+l} = 0$ and $l$ even.*

*Proof.* First note that an element $y$ is in $U_\beta$ if and only if

$$\begin{aligned} y \in U_\beta &\Leftrightarrow \beta_i y_{i-1} + \beta_{i+1} y_{i+1} = 0 \quad \forall i \in [n] \\ &\Leftrightarrow y_i \beta_{i+1} = y_{i+2} \beta_{i+2} \quad \forall i \in [n] \\ &\Leftrightarrow y_i \beta_i = y_{i-2} \beta_{i-1} \quad \forall i \in [n] \end{aligned}$$

Furthermore, recall that $U_\beta^\perp$ is defined as

$$U_\beta^\perp = \{x \in \mathbb{F}_2^n \mid \langle x, y \rangle = 0 \; \forall y \in U_\beta\}.$$

$\Rightarrow$ We first show that any element $\alpha \in U_\beta^\perp$ has to fulfill the conditions (i) and (ii).

The conditions for $y \in U_\beta$ show that, if two consecutive bits of $\beta$ are zero ($\beta_i = 0$ and $\beta_{i+1} = 0$) then the unit vector $e_i$ is contained in $U_\beta$. This in turn implies that for $\alpha \in U_\beta^\perp$ it follows that

$$\langle \alpha, e_i \rangle = \alpha_i = 0$$

and thus condition (i) is fulfilled.

For (ii), let $\beta$ have a block of $l$ ones with $\beta_i = \beta_{i+1} = \cdots = \beta_{i+l-1} = 1$ and $\beta_{i-1} = \beta_{i+l} = 0$. Then $y_{i-1}, y_i, \ldots, y_{i+l-1}$ are only determined by the mentioned block in $\beta$. For $y_{i-1}$ holds that $y_{i-1} = y_{i+1}$ when $l > 1$. For $y_i$ and $y_{i+l-2}$ holds that $y_i = y_{i+l-2} = 0$. The entire constraints for the block in $y$ are:

$$\begin{aligned} y_i &= 0 \\ y_{i+l-2} &= 0 \\ y_{i+k} &= y_{i+k+2},\ k \in [l-2] \end{aligned}$$

If $l$ is odd, then the constraints are equivalent to $y_{i-1} = y_i = \cdots = y_{i+l-1} = 0$, because of $y_i = y_{i+2} = \cdots = y_{l-1} = 0$ and $y_{i+l-2} = y_{i+l-4} = \cdots = y_{i-1} = 0$.

If $l$ is even, then the constraints are equivalent to $y_i = y_{i+2} = \cdots = y_{i+l-2} = 0$ and $y_{i-1} = y_{i+1} = \cdots = y_{i+l-1}$. Let

$$s = \alpha_{i-1} + \alpha_{i+2} + \cdots + \alpha_{i+l-1}.$$

Assume that $s = 1$ for any block of $l$ ones with $l$ even in $\beta$. Look at $y'$ with $y'_{i-1} = y'_{i+1} = \cdots = y'_{i+l-1} = 1$ and the rest all zero. It clearly holds that $y' \in U_\beta$ and $\langle y', \alpha \rangle = 1$. That is a contradiction to $\alpha \in U_\beta^\perp$ and leads to: $s = 0$ for all blocks in $\beta$ with $l$ ones, where $l$ is even.

$\Leftarrow$ Now we show that any $\alpha$ fulfilling conditions (i) and (ii) actually belongs to $U_\beta^\perp$.

Because of $\alpha_i = 1 \Rightarrow (\beta_i = 1 \vee \beta_{i+1} = 1)$ we only have to look at $\alpha_{i-1}, \cdots, \alpha_{i+l-1}$ for every block in $\beta$. Since $y_{i-1} = y_i = \cdots = y_{i+l-1} = 0$ for an odd $l$, no $\alpha_j$ with $j \in \{i-1, i, \cdots, i+l-1\}$ affects the scalar product of $\alpha$ and $y$. For an even $l$, only the sum $\alpha_{i-1} + \alpha_{i+2} + \cdots + \alpha_{i+l-1}$ is important for this block. Because of $y_{i-1} = y_{i+1} = \cdots = y_{i+l-1}$ either all or none of them are used in the scalar product. Since $s = 0$ for any even $l$, $\langle \alpha, y \rangle = 0$.

It follows that $\alpha \in U_\beta^\perp$.

□

Now we can give an upper bound for the correlation of the non-linear layer of Dasta based on the Hamming weight of the input mask.

**Theorem 4.** *Let $\alpha$ be an input mask for the non-linear Layer $\chi$ of* Dasta*. Then for every $\beta$ it holds that $|C_\chi(\alpha \to \beta)| \leq 2^{-\left\lceil \frac{hw(\alpha)}{4} \right\rceil}$.*

*Proof.* Equation (7) shows that $|C_\chi(\alpha \to \beta)| = |C_f(\alpha' \to \beta)|$, where

$$\alpha' = S^{-2}(\alpha) + S^{-2}(\beta) + \beta$$

is the input mask for $f(x) = x \odot S^1(x)$ as shown in Figure 6. Now the goal is to construct a $\beta$ for a given $\alpha$, such that $|C_f(\alpha' \to \beta)|$ is maximal. This is equivalent to minimizing $\theta(\beta)$ with respect to $\alpha' \in U_\beta^\perp$.

The constraints for $\alpha' \in U_\beta^\perp$ are defined in Lemma 3. The first constraint is given by

$$\alpha'_i = 1 \Rightarrow \beta_i = 1 \vee \beta_{i+1} = 1 \quad \forall i \in [n].$$

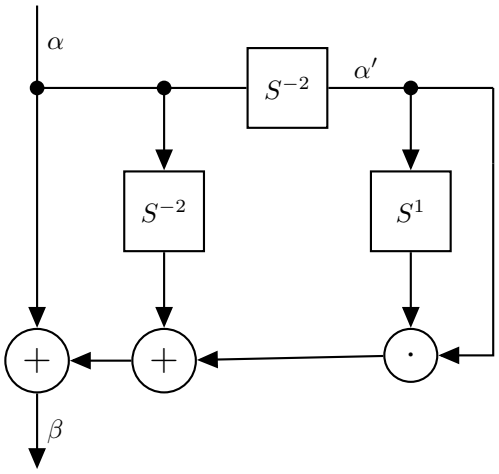


**Figure 6:** Non-linear Layer of DASTA.

This can be transformed to a constrained for $\alpha$ as follows.

$$\begin{aligned}
& \alpha'_i = 1 \Rightarrow \beta_i = 1 \vee \beta_{i+1} = 1 \quad \forall i \in [n] \\
\Leftrightarrow \quad & \alpha'_i = 0 \vee \beta_i = 1 \vee \beta_{i+1} = 1 \quad \forall i \in [n] \\
\Leftrightarrow \quad & \alpha_{i+2} + \beta_{i+2} + \beta_i = 0 \vee \beta_i = 1 \vee \beta_{i+1} = 1 \quad \forall i \in [n] \\
\Leftrightarrow \quad & \alpha_{i+2} = \beta_{i+2} \vee \beta_i = 1 \vee \beta_{i+1} = 1 \quad \forall i \in [n] \\
\Leftrightarrow \quad & \alpha_i = \beta_i \vee \beta_{i-1} = 1 \vee \beta_{i-2} = 1 \quad \forall i \in [n]
\end{aligned}$$

Now, the above constraint implies that a single 1 in $\beta$ can cover at most three 1s in $\alpha$. Furthermore with a block of two consecutive 1s in $\beta$ at most four 1s in $\alpha$ can be covered. This implies that $\theta(\beta) \geq \mathbf{hw}(\alpha)/4$. It follows that

$$|C_\chi(\alpha \to \beta)| \leq 2^{-\theta(\beta)} \leq 2^{-\left\lceil \frac{\mathbf{hw}(\alpha)}{4} \right\rceil},$$

as claimed.

□

We now can finally prove the main result of this section, i. e. Theorem 2.

*Proof.* Let $\beta$ be an output mask for $\chi$ with $\mathbf{hw}(\beta) = w$. Then for every $\alpha$ it holds that

$$|C_\chi(\alpha \to \beta)| \leq 2^{-\left\lceil \frac{w}{2} \right\rceil}$$

since $\theta(\beta) \geq \left\lceil \frac{w}{2} \right\rceil$.

Let $\beta$ now be the output mask after the first application of $\chi$ and $\alpha$ be the input mask of the second $\chi$. Then $\beta$ is also the input mask and $\alpha$ the output mask of the linear layer $L$. If $\beta = \mathbf{0}$ or $\alpha = \mathbf{0}$, the correlation is 0. Thus we assume that both masks are not the zero vector. Then follows $\mathbf{hw}(\beta) \geq 1$ and $\mathbf{hw}(\alpha) \geq 1$. Since the binary branch number of $L$ is $b$ it holds that $\mathbf{hw}(\beta) + \mathbf{hw}(\alpha) \geq b$.

For $\mathbf{hw}(\beta) = 1 \vee \mathbf{hw}(\beta) = 2$ the correlation of $\chi$ is upper-bounded by $2^{-1}$. So the correlation for two rounds is maximal for $\mathbf{hw}(\beta) = 2$ and $\mathbf{hw}(\alpha) = b - 2$, because for a given Hamming weight $w$ of the input mask, the correlation is upper-bounded by $2^{-\left\lceil \frac{w}{4} \right\rceil}$,

while for a given Hamming weight $w$ of the output mask it is $2^{-\lceil \frac{w}{2} \rceil}$. This leads to a maximal absolute two round correlation of

$$2^{-\left(\lceil \frac{b-2}{4} \rceil + 1\right)}.$$

□

# 6 Subspace Trails

Subspace trails have been introduced in [GRR16] as an alternative view on truncated differentials with probability one. The definition of a (one round) subspace trail is given below, we refer to [GRR16] for more details.

**Definition 2.** Let $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$. Linear subspaces $U, V \subseteq \mathbb{F}_2^n$ are called a subspace trail, if

$$\forall a \colon \exists b \colon F(U + a) \subseteq V + b.$$

We denote this by $U \xrightarrow{F} V$. We call the subspace trail trivial, when $U = \{\mathbf{0}\}$ or $V = \mathbb{F}_2^n$.

In this section we show that no non-trivial subspace trail, that is no non-trivial truncated differential with probability one, exists for Dasta.

While differential cryptanalysis in general does not seem applicable neither to Rasta nor to Dasta, truncated differentials with probability one would translate into linear combinations of key-stream bits that would be independent of linear combinations of key bits. Such properties, depending on the exact dimensions, could potentially imply guess-and-determine attacks. An attacker could use the independence of linear combinations of key-stream bits to verify partial key-guesses instead of having to guess the entire key. It should be noted that as the key-size in Rasta and Dasta are significantly larger than the security parameter, not every truncated differential would translate into a valid attack on the cipher.

Subspace trail cryptanalysis was not treated explicitly for Rasta. Indeed, it seems (i) rather unlikely to apply due to the randomness of the liner layers and (ii) – for the same reason– hard to explicitly exclude. As we will show in this section, the situation is different for Dasta. Before going into the technical details of the argument why no subspace trail exists for Dasta, we sketch the high level ideas.

It might also be of interest to consider subspace trails (or truncated differentials) of probability less than one. However, we think (i) it is very hard to analyze and (ii) unlikely to be a problem.

## 6.1 Idea

In the following we introduce properties for $L$, which can be computed efficiently and ensure the absence of non-trivial subspace trails over the full cipher. We show for every version of Dasta that any subspace trail over 3 or 4 rounds has full dimension. This means that for any trail $U \xrightarrow{4R} V$ where $U \neq \{\mathbf{0}\}$ we have $V = \mathbb{F}_2^n$. Two instances only differ in the bit permutation. When we consider the linear layer $L$ as a matrix in $\mathbb{F}_2^{n \times n}$, the multiplication with $L$ together with the bit permutation before is the same as the multiplication with a matrix $L'$, where the columns of $L$ are permuted. The following proof holds for all instances since it is invariant over the permutation of the columns in $L$.

We show in Lemma 5 that any one round subspace trail $U \xrightarrow{\chi} V$ contains at least one unit vector. For the versions 128-5, 128-4, 256-6, and 256-5 we compute the minimum

number $k$ of unit vectors in $V$ for all $U$ which contain just one unit vector for a one round subspace trail $U \xrightarrow{\chi\circ L} V$. That is

$$k = \min_{0\leq i<n} \left\{ |\{v \in V \mid v \text{ is unit vector}\}| \mid \{e_i\} \xrightarrow{\chi\circ L} V \right\}.$$

Then we know that for any non-trivial subspace trail $U \xrightarrow{\chi\circ L\circ\chi} V$, $V$ contains at least $k$ unit vectors. Since $L$ satisfies the conditions of Theorem 6 with the computed $k$, it is shown that any subspace trail over 3 rounds has full dimension.

The argumentation for the versions 80-6, 80-4, and 128-6 is slightly more complicated. We computed the $k$ as defined above, which is greater or equal to 2 for the three versions. Then we compute

$$k' = \min_{\substack{0\leq i,j<n \\ i\neq j}} \{|\{v \in V \mid v \text{ is unit vector}\}| \mid \{e_i, e_j\} \xrightarrow{\chi\circ L} V\}.$$

which is the minimum number of unit vectors in $V$, starting with two unit vectors. We know that for any non-trivial subspace trail $U \xrightarrow{\chi\circ L\circ\chi\circ L\circ\chi} V$, $V$ contains at least $k'$ unit vectors. Since $L$ satisfies the conditions of Theorem 6 with the computed $k'$, it is shown that any subspace trail over 4 rounds has full dimension. The values of $k$ and $k'$ are invariant over all column permutations of $L$.

## 6.2 Technical Details

First we look at a method to compute a minimal subspace trail $U \xrightarrow{f} V$ for a given $U \subseteq \mathbb{F}_2^n$, where $f\colon \mathbb{F}_2^n \to \mathbb{F}_2^n$. The subspace trail is minimal when

$$\forall V'\colon U \xrightarrow{f} V' \Rightarrow V \subseteq V'.$$

Given $U$ and $f$ it is shown in [LTW18, Lemma 1, 2] that $V$ for the corresponding minimal subspace trail $U \xrightarrow{f} V$ can be computed by

$$V = \operatorname{span}\left( \bigcup_{1\leq i\leq k} \operatorname{Im}(\Delta_{b_i}(f)) \right)$$

where $b_1, b_2, \ldots, b_k$ is a basis of $U$ and $\Delta_{b_i}$ is the derivative in direction $b_i$ as defined in Equation (2). This means for the linear layer $L$ that any minimal subspace trail has the form $U \xrightarrow{L} L(U)$. The linear layer alone does not increase the dimension since $\dim U = \dim L(U)$.

For the non-linear layer $\chi$ we need to compute the image of a derivative in point $b$ $\operatorname{Im}(\Delta_b(\chi))$. The degree of $\chi$ is 2 so that the degree of the derivative is at most 1. This leads to

$$\operatorname{Im}(\Delta_b(\chi)) = \operatorname{span}\left( \{\Delta_b(\chi)(\mathbf{0})\} \cup \bigcup_{1\leq i\leq n} \Delta_b(\chi)(e_i) \right)$$

since the image of an affine function can be computed by the span of the unit vectors and the zero vector.

Further an element $x \in \mathbb{F}_2^n$ is in $\operatorname{Im}(\Delta_b(\chi))$ iff the probability of the differential $\Pr\left[b \xrightarrow{\chi} x\right]$ is greater than 0. We can use Theorem 5 to compute these probabilities of differentials, which is based on [KLT15, Theorem 2] and adapted for an odd block length $n$.

The proof can be found in Appendix A.2. In the following the $\vee$ operator is also defined as element-wise or, i. e.

$$(x \vee y)_i \coloneqq x_i + y_i + x_i y_i$$

where $x, y \in \mathbb{F}_2^n$.

**Theorem 5.** *Let $f\colon \mathbb{F}_2^n \to \mathbb{F}_2^n, f(x) = x \odot S^1(x)$, $\alpha \in \mathbb{F}_2^n$ an input difference, and $\beta \in \mathbb{F}_2^n$ an output difference where $n$ is odd. Then*

$$Pr\left[\alpha \xrightarrow{f} \beta\right] = \begin{cases} 2^{-n+1} & \text{if } \alpha = \mathbf{1} \text{ and } \mathbf{hw}(\beta) \equiv 1 \bmod 2 \\ 2^{-\mathbf{hw}(\text{varibits}+\text{doublebits})} & \text{if } \alpha \neq \mathbf{1} \text{ and } \beta \odot \overline{\text{varibits}} = \mathbf{0} \\ & \text{and } (\beta + S^1(\beta)) \odot \text{doublebits} = \mathbf{0} \\ 0 & \text{else} \end{cases}$$

*where*

$$\text{varibits} = S^1(\alpha) \vee \alpha$$

*and*

$$\text{doublebits} = \alpha \odot \overline{S^1(\alpha)} \odot S^2(\alpha).$$

*Proof.* See Appendix A.2. □

Now we have

$$\begin{aligned} \Pr\left[\alpha \xrightarrow{\chi} \beta\right] &= \frac{|\{x \in \mathbb{F}_2^n \mid \chi(x) + \chi(x+\alpha) = \beta\}|}{2^n} \\ &= \frac{|\{x \in \mathbb{F}_2^n \mid S^{-1}(x) \odot S^{-2}(x) + S^{-1}(x+\alpha) \odot S^{-2}(x+\alpha) = \beta + \alpha + S^{-2}(\alpha)\}|}{2^n} \\ &= \Pr\left[S^{-2}(\alpha) \xrightarrow{f} \beta + \alpha + S^{-2}(\alpha)\right] \end{aligned}$$

where $f(x) = x \odot S^1(x)$ so that we can compute the probability of differentials over $\chi$ with Theorem 5. The next lemma gives conditions on bit level whether a vector is part of $\text{Im}(\Delta_b(\chi))$.

**Lemma 4.** *Let $b \in \mathbb{F}_2^n$, then $y \in \text{Im}(\Delta_b(\chi))$ for $b \neq \mathbf{1}$ iff*

1. $$\forall i \in [n] : \; y_i \in \begin{cases} \{0,1\} & \text{if } b_{i+1} = 1 \text{ or } b_{i+2} = 1 \\ \{1\} & \text{if } b_i = 1 \text{ and } b_{i+1} = b_{i+2} = 0 \\ \{0\} & \text{else} \end{cases}$$

2. *and* $\forall i \in [n] : \; (b_i = b_{i+2} = 1 \wedge b_{i+1} = 0) \Rightarrow y_i + y_{i-1} + b_{i-1} = 0$.

*For $b = \mathbf{1}$ holds that $\text{Im}(\Delta_b(\chi)) = \{y \mid \mathbf{hw}(y) \equiv 1 \bmod 2\}$.*

*Proof.* We have

$$\begin{aligned} y \in \text{Im}(\Delta_b(\chi)) &\Leftrightarrow \Pr\left[b \xrightarrow{\chi} y\right] > 0 \\ &\Leftrightarrow \Pr\left[S^{-2}(b) \xrightarrow{f} y + b + S^{-2}(b)\right] > 0. \end{aligned}$$

The case for $b = \mathbf{1}$ can be directly concluded from Theorem 5. When $b \neq \mathbf{1}$, $y \in \text{Im}(\Delta_b(\chi))$ iff

$$(y + b + S^{-2}(b)) \odot \overline{(S^{-1}(b) \vee S^{-2}(b))} = \mathbf{0} \tag{8}$$

$$\text{and } (y + b + S^{-2}(b) + S^1(y) + S^1(b) + S^{-1}(b)) \odot S^{-2}(b) \odot \overline{S^{-1}(b)} \odot b = \mathbf{0} \tag{9}$$

according to Theorem 5. Now Equation (8) is equivalent to

$$(y_i + b_i + b_{i+2}) \cdot ((b_{i+1} + 1) \cdot (b_{i+2} + 1)) = 0, \quad \forall i \in [n]$$

which is the same as the first condition. Further Equation (9) can be simplified to

$$(y + S^1(y) + S^1(b)) \odot S^{-2}(b) \odot \overline{S^{-1}(b)} \odot b = \mathbf{0} \tag{10}$$

$$\Leftrightarrow \quad (y_i + y_{i-1} + b_{i-1}) \cdot b_{i+2} \cdot (b_{i+1} + 1) \cdot b_i = 0, \quad \forall i \in [n] \tag{11}$$

which is the same as the second condition.

□

Lemma 5 is needed to compute the lower bound of unit vectors after a non-trivial subspace trail over 2 rounds.

**Lemma 5.** *Let $U \xrightarrow{\chi} V$ be an one round subspace trail, where $U \neq \{\mathbf{0}\}$. Then $V$ contains at least one unit vector.*

*Proof.* Let $b \in U, b \neq \mathbf{0}$ be arbitrarily chosen but fixed. The case $b = \mathbf{1}$ is trivial, since

$$V = \mathrm{span}(\mathrm{Im}(\Delta_b(\chi))) = \mathrm{span}(\{y \mid \mathbf{hw}(y) \equiv 1 \bmod 2\}).$$

Now let further $b \neq \mathbf{1}$. Since $n$ is odd, there are at least two equal adjacent positions in $b$. This means that

$$\exists i \in [n] \colon (b_i = b_{i+1} = 0 \ \wedge \ b_{i+2} = 1) \ \vee \ (b_{i+1} = b_{i+2} = 1 \wedge b_i = 0),$$

which implies that there are no restrictions for a $y \in \mathrm{Im}(\Delta_b(\chi))$ at position $i$ according to Lemma 4. This means for an arbitrary but fixed $y \in \mathrm{Im}(\Delta_b(\chi))$ the vector $y'$ which equals $y$ in all positions except $i$, i. e. $y' = y + e_i$, is also in the image of $\Delta_b(\chi)$. Since $V$ is the span of the image, it holds that $e_i \in V$.

□

In the following the linear layer $L$ is represented as a matrix in $\mathbb{F}_2^{n \times n}$. Further $L[*, i]$ denotes the $i$-th column of $L$ and $L[i, *]$ the $i$-th row of $L$ for $i \in [n]$.

**Definition 3.** We call a position $i \in [n]$ of a column vector $v = L[*, j]$ *semi-critical*, when

$$\begin{aligned} & v_{i+1} = v_{i+2} = 0 \\ \text{or} \quad & v_i = v_{i+2} = 1 \ \wedge \ v_{i+1} = 0 \\ \text{or} \quad & v_{i+1} = v_{i+3} = 1 \ \wedge \ v_{i+2} = 0. \end{aligned}$$

We call a position $i \in [n]$ of a column vector $v = L[*, j]$ *critical*, when

$$\begin{aligned} & v_{i+1} = v_{i+2} = 0 \\ \text{or} \quad & v_{i+1} = v_{i+3} = 1 \ \wedge \ v_{i+2} = 0. \end{aligned}$$

Considering a vector $v \in \mathbb{F}_2^n$ which is not semi-critical at position $i$ means that the unit vector $e_i$ is in the image of $\Delta_v(\chi)$. When position $i$ is not critical this means that $e_i$ or $e_i + e_{i-1}$ is in $\mathrm{Im}(\Delta_v(\chi))$. We show this later. In order to prove that any non-trivial subspace trail results after a few rounds in the full vector space, we show that every unit vector is contained. For this we care about the critical and semi-critical positions. We introduce two functions $\Theta$ and $\Omega$, which can efficiently be evaluated on $L$.

**Definition 4.** The function $\Theta : \mathbb{F}_2^{n\times n} \times [n] \to \mathbb{N}$ computes for every column in $L$ if there is at least one position, which is not semi-critical:

$$\Theta(L,i) = \begin{cases} 1, & \text{if } \exists j \in [n] \text{ so that position } j \text{ of } L[*,i] \text{ is not semi-critical} \\ 0, & \text{else} \end{cases}$$

The number of column vectors of $L$, which are critical at position $i$ are computed by $\Omega : \mathbb{F}_2^{n\times n} \times [n] \to \mathbb{N}$:

$$\Omega(L,i) = |\{v_i \text{ is critical } \mid v = L[*,j],\ j \in [n]\}|.$$

Now finally Theorem 6 gives an efficiently computable condition to calculate the full dimension of a subspace trail. If the premise of the following theorem holds for $L$, then so it does for $L' = L \circ P$ for all permutation matrices $P$. Thus we only need to state the theorem for $L$.

**Theorem 6.** *Let $U \xrightarrow{\chi\circ L} V$ be a subspace trail, where $U$ contains $k > 0$ unit vectors and*

$$\begin{aligned} &\forall i \in [n] :\ \Theta(L,i) = 1 \\ \textit{and}\quad &\forall i \in [n] :\ \Omega(L,i) < k. \end{aligned}$$

*Then we have $V = \mathbb{F}_2^n$.*

*Proof.* We can write the subspace trail as $U \xrightarrow{L} V'$ and $V' \xrightarrow{\chi} V$. Then we know that $V'$ contains at least $k$ column vectors of $L$. First we show that

$$\forall i \in [n] \colon \Omega(L,i) < k \Rightarrow e_i \in V \ \vee\ e_i + e_{i-1} \in V.$$

Since $\Omega(L,i) < k$, there exists at least one $b \in V'$ so that position $i$ is not critical. Now we have two cases:

- **Case 1:** Position $i$ of $b$ is not semi-critical

  Let $y$ be an arbitrary element in $\mathrm{Im}(\Delta_b(\chi))$. Then $y'$, where $y'_j = y_j$ for $j \in [n],\ j \neq i$ and $y'_i = y_i + 1$, is also in $\mathrm{Im}(\Delta_b(\chi))$ according to Lemma 4. This leads to

  $$y + y' = e_i \in \mathrm{span}(\mathrm{Im}(\Delta_b(\chi)))$$

  so that $e_i$ is in $V$.

- **Case 2:** Position $i$ of $b$ is semi-critical

  This leads to $b_i = b_{i+2} = 1$ and $b_{i+1} = 0$ so that

  $$\forall y \in \mathrm{Im}(\Delta_b(\chi)) \colon y_i + y_{i-1} + b_{i-1} = 0.$$

  Let $y$ be an arbitrary element in $\mathrm{Im}(\Delta_b(\chi))$. Then $y'$, where $y'_j = y_j$ for $j \in [n],\ j \neq i \ \wedge\ j \neq i-1$ and $y'_i = y_i + 1 \ \wedge\ y'_{i-1} = y_{i-1} + 1$, is also in $\mathrm{Im}(\Delta_b(\chi))$ according to Lemma 4. The second constraint is fulfilled since $y_i + y_{i-1} = y'_i + y'_{i-1}$. This leads to $y + y' = e_i + e_{i-1} \in V$.

It holds that $\forall i \in [n] \colon \Theta(L,i) = 1$ so that $V'$ contains at least one column vector of $L$ which has a position $j$ that is not semi-critical. As you can see above, this leads to $e_j \in V$.

Now we need to show that every unit vector $e_i$ is contained in $V$. We already showed that $V$ contains $e_i$ or $e_i + e_{i-1}$ for all $i \in [n]$. So we just have to look at the case where we can deduce that $e_i + e_{i-1} \in V$. It exists an $m \in [n]$ so that

$$\begin{aligned} &e_m \in V \\ \text{and}\quad &e_{j-1} + e_j \in V,\ \forall j \in \{m+1, \ldots, i\}. \end{aligned}$$

By adding $e_m$ and $\sum_{j=m+1}^{i} e_{j-1} + e_j$, which are all in $V$, it follows that $e_i \in V$. This leads to $V = \mathbb{F}_2^n$.

□

# 7 Linearization Attacks

The trivial linearization is one possible variant of algebraic attacks. Here each key stream bit of Rasta is seen as an equation dependent on the key bits, i.e. the algebraic normal form (ANF) is considered. Every non-linear term, i. e. every monomial of degree 2 or higher, is substituted with a new variable, resulting in a linear system in a (significantly) increased number of variables. After collecting enough linear independent equations, the key can be recovered by solving the linear equation system. In the (conservative) analysis of Rasta the cost of setting up the equations as well as solving the linear system has been ignored. Instead, the idea was to bound the data limit so that collecting a sufficient amount of equations is not possible.

For Rasta and Dasta, as $\chi$ is a quadratic function, the algebraic degree of the cipher is upper-bounded by $2^r$ for $r$ rounds. Therefore, the number of different monomials which can occur in the different equations is

$$U = \sum_{i=0}^{2^r} \binom{n}{i}, \tag{12}$$

Furthermore, by guessing $g$ key bits the time complexity of an attack is multiplied by $2^g$ and the number of monomials is reduced to

$$U_g = \sum_{i=0}^{2^r} \binom{n-g}{i}.$$

For Rasta it was argued in [Dob+18] that, due to the random choice of the linear layers, it is valid to assume that all possible monomials actually occur (with probability close to 1/2) in the system of equations. The parameters were then chosen such that, under this assumption, linearization will not be possible due to constrains in the data limit.

For Dasta we make the same assumption, i.e. the resulting system is well approximated by the worst case and contains all possible monomials up to the attainable maximal degree. As theoretical arguments seem very difficult for Dasta – actually for almost any symmetric cipher – we performed extensive simulations to substantiate our assumption. Those experiments, that can be found in Appendix B, did not show any significant difference between Rasta, Dasta and a random function with respect to the distribution of monomials. Thus, with respect to linearization attacks and more generally algebraic attacks, we state that the same parameters for Dasta as for Rasta will make those attacks infeasible.

For an actual attack, assuming the we can solve a system of linear equation with $n$ unknowns and equations in time $n^{2.37}$ using ideas of Coppersmith and Winograd [CW87] and recent improvements [Gal14], we can conclude that such an attack is only possible as long as

$$\left(\sum_{i=0}^{2^r} \binom{n}{i}\right)^{2.37} \leq 2^s. \tag{13}$$

We derive concrete bounds for the instances of Dasta in Table 2 in Section 3.

# 8 Conclusion

In this paper, for the first time, we present a symmetric cipher, that has at the same time a very low AND-depth and a very low number of ANDs per bit and does not rely on the

use of a PRNG. This is achieved by our variant of Rasta, called Dasta, by restricting the possible linear layers to a subset that excludes any weak instances. Technically, this required an analysis of the similarity of bit permutations. For linear approximations, a detailed study of the $\chi$ function, in particular a characterization of the possible input masks for a given output mask (see Lemma 3) and an upper bound of the correlation in this case (see Theorem 4) have been given.

We see a couple of further research directions. First, as the block sizes in Rasta are chosen rather conservatively, it might be interesting to see if they could be reduced, in particular taking into account (i) our improved understanding of the non-linear layer and (ii) the costs of actually solving the system of equations in the case of linearization. Second, for Dasta it might be of interest to have even more structured linear layers, e.g. following the ideas in [Per+17] after examining linear approximations for this approach.

Finally, even so we could not find any weaknesses in our design, as the design is rather unconventional, we encourage further analysis to (hopefully) strengthen the trust in Dasta.

## Acknowledgments

The authors thank Andre Esser for his careful proofreading. We would also like to thank the anonymous reviewers for their helpful comments that significantly improved our work. This work was partially funded by the DFG (German Research Foundation) under Germany´s Excellence Strategy - EXC 2092 CASA – 390781972 and within the DFG project LE 3372/4-1.

## References

[Alb+15] M. R. Albrecht, C. Rechberger, T. Schneider, T. Tiessen, and M. Zohner. "Ciphers for MPC and FHE". In: *EUROCRYPT 2015, Part I*. Ed. by E. Oswald and M. Fischlin. Vol. 9056. LNCS. Springer, Heidelberg, Apr. 2015, pp. 430–454. DOI: 10.1007/978-3-662-46800-5_17 (cit. on pp. 46, 48).

[BBK14] A. Biryukov, C. Bouillaguet, and D. Khovratovich. "Cryptographic Schemes Based on the ASASA Structure: Black-Box, White-Box, and Public-Key (Extended Abstract)". In: *ASIACRYPT 2014, Part I*. Ed. by P. Sarkar and T. Iwata. Vol. 8873. LNCS. Springer, Heidelberg, Dec. 2014, pp. 63–84. DOI: 10.1007/978-3-662-45611-8_4 (cit. on p. 46).

[BGV11] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. *Fully Homomorphic Encryption without Bootstrapping*. Cryptology ePrint Archive, Report 2011/277. http://eprint.iacr.org/2011/277. 2011 (cit. on p. 46).

[Bea+13] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers. *The SIMON and SPECK Families of Lightweight Block Ciphers*. Cryptology ePrint Archive, Report 2013/404. http://eprint.iacr.org/2013/404. 2013 (cit. on p. 66).

[Ber+13] G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche. "Keccak". In: *EUROCRYPT 2013*. Ed. by T. Johansson and P. Q. Nguyen. Vol. 7881. LNCS. Springer, Heidelberg, May 2013, pp. 313–314. DOI: 10.1007/978-3-642-38348-9_19 (cit. on pp. 47, 50).

[CW87] D. Coppersmith and S. Winograd. "Matrix multiplication via arithmetic progressions". In: *Proceedings of the nineteenth annual ACM symposium on Theory of computing*. 1987, pp. 1–6 (cit. on p. 75).

[Can+15] A. Canteaut, S. Carpov, C. Fontaine, T. Lepoint, M. Naya-Plasencia, P. Paillier, and R. Sirdey. *Stream ciphers: A Practical Solution for Efficient Homomorphic-Ciphertext Compression*. Cryptology ePrint Archive, Report 2015/113. http://eprint.iacr.org/2015/113. 2015 (cit. on p. 46).

[DR01] J. Daemen and V. Rijmen. "The Wide Trail Design Strategy". In: *Cryptography and Coding, 8th IMA International Conference, Cirencester, UK, December 17-19, 2001, Proceedings*. Ed. by B. Honary. Vol. 2260. Lecture Notes in Computer Science. Springer, 2001, pp. 222–238. ISBN: 3-540-43026-1. DOI: 10.1007/3-540-45325-3_20. URL: https://doi.org/10.1007/3-540-45325-3_20 (cit. on p. 49).

[Dae95] J. Daemen. "Cipher and hash function design strategies based on linear and differential cryptanalysis". PhD thesis. Katholieke Universiteit Leuven, 1995 (cit. on pp. 47, 49, 64, 66).

[Din+19] I. Dinur, D. Kales, A. Promitzer, S. Ramacher, and C. Rechberger. "Linear Equivalence of Block Ciphers with Partial Non-Linear Layers: Application to LowMC". In: *EUROCRYPT 2019, Part I*. Ed. by Y. Ishai and V. Rijmen. Vol. 11476. LNCS. Springer, Heidelberg, May 2019, pp. 343–372. DOI: 10.1007/978-3-030-17653-2_12 (cit. on p. 47).

[Dob+18] C. Dobraunig, M. Eichlseder, L. Grassi, V. Lallemand, G. Leander, E. List, F. Mendel, and C. Rechberger. "Rasta: A Cipher with Low ANDdepth and Few ANDs per Bit". In: *CRYPTO 2018, Part I*. Ed. by H. Shacham and A. Boldyreva. Vol. 10991. LNCS. Springer, Heidelberg, Aug. 2018, pp. 662–692. DOI: 10.1007/978-3-319-96884-1_22 (cit. on pp. 46, 50, 56, 64, 75).

[GRR16] L. Grassi, C. Rechberger, and S. Rønjom. "Subspace Trail Cryptanalysis and its Applications to AES". In: *IACR Trans. Symm. Cryptol.* 2016.2 (2016). http://tosc.iacr.org/index.php/ToSC/article/view/571, pp. 192–225. ISSN: 2519-173X. DOI: 10.13154/tosc.v2016.i2.192-225 (cit. on pp. 56, 70).

[Gal14] F. L. Gall. "Powers of Tensors and Fast Matrix Multiplication". In: *CoRR* abs/1401.7714 (2014). arXiv: 1401.7714. URL: http://arxiv.org/abs/1401.7714 (cit. on p. 75).

[HS13] S. Halevi and V. Shoup. "Design and implementation of a homomorphic-encryption library". In: (2013) (cit. on pp. 48, 57, 58).

[KLT15] S. Kölbl, G. Leander, and T. Tiessen. "Observations on the SIMON Block Cipher Family". In: *CRYPTO 2015, Part I*. Ed. by R. Gennaro and M. J. B. Robshaw. Vol. 9215. LNCS. Springer, Heidelberg, Aug. 2015, pp. 161–185. DOI: 10.1007/978-3-662-47989-6_8 (cit. on pp. 64, 66, 67, 71, 78, 80).

[LTW18] G. Leander, C. Tezcan, and F. Wiemer. "Searching for Subspace Trails and Truncated Differentials". In: *IACR Trans. Symm. Cryptol.* 2018.1 (2018), pp. 74–100. ISSN: 2519-173X. DOI: 10.13154/tosc.v2018.i1.74-100 (cit. on p. 71).

[MS78] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes*. Elsevier, 1978 (cit. on p. 49).

[McE78] R. J. McEliece. "A public-key cryptosystem based on algebraic". In: *Coding Thv* 4244 (1978), pp. 114–116 (cit. on p. 55).

[Méa+16] P. Méaux, A. Journault, F.-X. Standaert, and C. Carlet. "Towards Stream Ciphers for Efficient FHE with Low-Noise Ciphertexts". In: *EUROCRYPT 2016, Part I*. Ed. by M. Fischlin and J.-S. Coron. Vol. 9665. LNCS. Springer, Heidelberg, May 2016, pp. 311–343. DOI: 10.1007/978-3-662-49890-3_13 (cit. on p. 46).

[Nat15] National Institute of Standards and Technology. *FIPS PUB 202: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*. U.S. Department of Commerce. 2015 (cit. on p. 50).

[Per+17] L. Perrin, A. Promitzer, S. Ramacher, and C. Rechberger. *Improvements to the Linear Layer of LowMC: A Faster Picnic*. Cryptology ePrint Archive, Report 2017/1148. https://eprint.iacr.org/2017/1148. 2017 (cit. on pp. 47, 76).

[Ran93] D. Randall. "Efficient Generation of Random Nonsingular Matrices". In: *Random Struct. Algorithms* 4.1 (1993), pp. 111–118. DOI: 10.1002/rsa.3240040108. URL: https://doi.org/10.1002/rsa.3240040108 (cit. on p. 57).

# A Proofs

## A.1 Proof of Theorem 3

Before we state the proof of Theorem 3 we have to generalize another statement of [KLT15].

**Theorem 7.** *Let $f(x) = x \odot S^1(x)$, where $x \in \mathbb{F}_2^n$ and $n$ is odd. Then*

$$\hat{f}(\alpha,\beta)^2 = \begin{cases} 2^{n+1} & \text{if } \beta = \mathbf{1} \text{ and } \boldsymbol{hw}(\alpha) \equiv 1 \bmod 2 \\ 2^{n+d} & \text{if } \beta \neq \mathbf{1} \text{ and } \alpha \in U_\beta^\perp \\ 0 & \text{else} \end{cases}$$

*where $U_\beta = \{y \mid \beta \odot S^1(y) + S^{-1}(\beta \odot y) = \mathbf{0}\}$ and $d = \dim U_\beta$.*

*Proof.*

$$\hat{f}(\alpha,\beta)^2 = 2^n \sum_{y \in U_\beta} \mu(\langle \beta, f(y)\rangle + \langle \alpha, y\rangle) \quad \text{(shown in [KLT15, p. 12])}$$

For all $y \in U_\beta$ holds $\beta_i y_{i-1} = \beta_{i+1} y_{i+1}$:

$$y \in U_\beta \Rightarrow \begin{pmatrix} \beta_0 y_{n-1} \\ \beta_1 y_0 \\ \vdots \\ \beta_{n-1} y_{n-2} \end{pmatrix} = \begin{pmatrix} \beta_1 y_1 \\ \beta_2 y_2 \\ \vdots \\ \beta_0 y_0 \end{pmatrix}$$

We can write $f_\beta(y) = \langle \beta, f(y)\rangle = \sum_{i=0}^{n-1} \beta_i y_i y_{i-1}$ as

$$f_\beta(y) = \beta_0 y_0 y_{n-1} + \beta_1 y_1 y_0 + \cdots + \beta_{n-1} y_{n-1} y_0$$

- Case 1: $\beta \neq \mathbf{1}$

  Then there exists a $\beta_i = 0$, so that we can write $f_\beta$ with $n-1$ monomials. Beginning at $k = i+2$ we can write every second monomial $\beta_k y_k y_{k-1}$ as $\beta_{k-1} y_{k-2} y_{k-1}$, because of $y \in U_\beta$. Now $f_\beta$ is constant zero on $U_\beta$ and $\langle \alpha, y\rangle$ is unbalanced iff $\alpha \in U_\beta^\perp$.

- Case 2: $\beta = \mathbf{1}$

  For $\beta = \mathbf{1}$, $U_\beta$ contains only $\mathbf{0}$ and $\mathbf{1}$ since the dimension of $U_\beta$ is 1 as showed in theorem 3. Because of $f_\beta(\mathbf{1}) = 1$ and $f_\beta(\mathbf{0}) = 0$, $\hat{f}$ is unbalanced iff $\langle \alpha, \mathbf{1} \rangle$ is 1. It follows that $\hat{f}(\alpha, \beta)^2 = 2^{n+1}$ if $\mathbf{hw}(\alpha) \equiv 1 \bmod 2$ and otherwise $\hat{f}(\alpha, \beta)^2 = 0$.

□

After this preparation, we can continue with the proof of Theorem 3.

*Proof.* Let $L_\beta(x) = \beta \odot S^1(x) + S^{-1}(\beta \odot x)$. Clearly $L_\beta$ is linear, so that we can write the function in matrix form. Also $U_\beta = \ker L_\beta(x)$ holds. Define the matrices $M_\beta$, $M_{S^1}$ and $M_{S^{-1}}$ as

$$M_\beta = \begin{pmatrix} \beta_0 & 0 & \cdots & 0 \\ 0 & \beta_1 & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & \beta_{n-1} \end{pmatrix} \quad M_{S^1} = \begin{pmatrix} 0_{1,n-1} & I_1 \\ I_{n-1} & 0_{n-1,1} \end{pmatrix} \quad M_{S^{-1}} = \begin{pmatrix} 0_{n-1,1} & I_{n-1} \\ I_1 & 0_{1,n-1} \end{pmatrix}$$

where $I_m$ is the identity matrix of dimension $m \times m$ and $0_{n,m} \in \mathbb{F}_2^{n \times m}$ is the matrix containing all zeros. We can then write $L_\beta$ in matrix form as

$$\begin{pmatrix} 0 & \beta_1 & 0 & \cdots & 0 & \beta_0 \\ \beta_1 & 0 & \beta_2 & 0 & \cdots & 0 \\ 0 & \beta_2 & 0 & \beta_3 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & \ddots & 0 & \beta_{n-1} \\ \beta_0 & 0 & \cdots & 0 & \beta_{n-1} & 0 \end{pmatrix}.$$

When we look at a block of 1s with length $l$ in $\beta$, i. e. $\beta_{i-1} = 0, \beta_i = \beta_{i+1} = \cdots = \beta_{i+l-1} = 1$ and $\beta_{i+l} = 0$, then the $l+1$ rows $i, i+1, \cdots i+l$ are non-zero. If $l+1$ is even, then all rows are linearly independent. If $l+1$ is odd, then the sum of the rows $i, i+2, \cdots, i+l-2$ is equal to row $i+l$, while the first $l$ rows are linearly independent. The rows are enumerated from 1 to $n$. Thus this block increases the rank of $L_\beta$ by $2 \cdot \lceil \frac{l}{2} \rceil$. When $\beta = \mathbf{1}$, the sum of the rows $1, 3, \cdots, n-2$ is equal to the $n$-th row, while the first $n-1$ rows are linearly independent.

Therefore the rank of $L_\beta$ is $2 \cdot \theta(\beta)$ and the dimension of $U_\beta$ is $n - 2 \cdot \theta(\beta)$. Together with Theorem 7 this is

$$\hat{f}(\alpha, \beta)^2 = 2^{n+(n-2\cdot\theta(\beta))} = 2^{2n-2\cdot\theta(\beta)}$$

for $\beta = \mathbf{1}$ and $\mathbf{hw}(\alpha) \equiv 1$ or $\beta \neq \mathbf{1}$ and $\alpha \in U_\beta^\perp$. This leads to

$$C_f^2(\alpha \to \beta) = \frac{\hat{f}(\alpha, \beta)^2}{2^{2n}} = 2^{-2\cdot\theta(\beta)}$$

where $\theta(\mathbf{1}) = \frac{(n+1)}{2}$.

□

## A.2 Proof of Theorem 5

*Proof.* In [KLT15, Theorem 1] it is shown that

$$\Pr\left[\alpha \xrightarrow{f} \beta\right] = \begin{cases} 2^{-(n-d)} & \text{if } \beta + \alpha \odot S^1(\alpha) \in \mathrm{Im}(L_\alpha) \\ 0 & \text{else} \end{cases}$$

where

$$L_\alpha(x) = x \odot S^1(\alpha) + \alpha \odot S^1(x)$$

and

$$d = \dim(\ker(L_\alpha)).$$

Now we need to proof two things. The first part is showing that the rank of $L_\alpha$ is $n-1$ when $\alpha = \mathbf{1}$, and $\mathbf{hw}(\text{varibits} + \text{doublebits})$ otherwise. This is done in [KLT15, Theorem 2]. For the second part of the proof, we need to check whether $y = \beta + \alpha \odot S^1(\alpha) \in \mathrm{Im}(L_\alpha)$. We distinguish two cases:

- Case $\alpha = \mathbf{1}$:

  Then is

  $$L_\alpha(x) = \begin{pmatrix} x_{n-1} + x_0 \\ x_0 + x_1 \\ x_1 + x_2 \\ \vdots \\ x_{n-2} + x_{n-1} \end{pmatrix}$$

  and

  $$\begin{aligned} y \in \mathrm{Im}(L_\alpha) &\Leftrightarrow \sum_{i=0}^{n-2} y_i = y_{n-1} \\ &\Leftrightarrow \sum_{i=0}^{n-2} \beta_i = \beta_{n-1} + 1. \end{aligned}$$

  This leads to $y \in \mathrm{Im}(L_\alpha) \Leftrightarrow \mathbf{hw}(\beta) \equiv 1 \bmod 2$.

- Case $\alpha \neq \mathbf{1}$:

  It exists at least one $i \in [n]$ so that $\alpha_i = 0$. Then is

  $$L_\alpha(x) = \begin{pmatrix} \alpha_0 x_{n-1} + \alpha_{n-1} x_0 \\ \alpha_1 x_0 + \alpha_0 x_1 \\ \vdots \\ \alpha_{i-1} x_{i-2} + \alpha_{i-2} x_{i-1} \\ \alpha_{i-1} x_i \\ \alpha_{i+1} x_i \\ \alpha_{i+2} x_{i+1} + \alpha_{i+1} x_{i+2} \\ \vdots \\ \alpha_{n-1} x_{n-2} + \alpha_{n-2} x_{n-1} \end{pmatrix} \quad \text{and} \quad \alpha \odot S^1(\alpha) = \begin{pmatrix} \alpha_0 \alpha_{n-1} \\ \alpha_1 \alpha_0 \\ \vdots \\ \alpha_{i-1} \alpha_{i-2} \\ \alpha_{i-1} \\ \alpha_{i+1} \\ \alpha_{i+2} \alpha_{i+1} \\ \vdots \\ \alpha_{n-1} \alpha_{n-2} \end{pmatrix}.$$

  Let $x \in \mathbb{F}_2^n$, where $x_i = 1$, $x_{i+1+2k} = \alpha_{i+1+2k}$ and $x_{i+2+2k} = 0$ for $k \in [\frac{n-1}{2}]$. Then is $L_\alpha(x) = \alpha \odot S^1(\alpha)$ and since $L_\alpha$ is linear we just have to check the conditions for

$\beta$ to be in the image of $L_\alpha$. If $\beta_j = 1$ then $\alpha_j = 1 \vee \alpha_{j-1} = 1$. When $\alpha_j = \alpha_{j-2} = 1$ and $\alpha_{j-1} = 0$, then must hold $\beta_j = \beta_{j-1}$. This leads to

$$y \in \mathrm{Im}(L_\alpha) \Leftrightarrow \beta \odot \overline{\text{varibits}} = \mathbf{0} \wedge (\beta + S^1(\beta)) \odot \text{doublebits} = \mathbf{0}.$$

□

# B Experiments

In order to check the assumptions on the independence of the ANFs we do several experiments to verify resistance against linearization attacks. Furthermore we analyze the role of the round constants. We look at four different variants of Rasta and Dasta:

- Standard Rasta as defined in Section 2.1 (std)
- Standard Rasta without round constants (std-noc)
- Dasta without round constants as defined in Equation 3 with complete random bit permutations (perm-noc)
- Dasta with round constants, which are added after the fixed linear layer $L$ (perm)

For the following experiments were computed the ANFs of the key stream dependent on the key bits for multiple random instances of all four variants. This means that all parts of the ciphers, which are not fixed, are sampled uniformly for every instance. To get an acceptable computation time, the block size is set to $n = 11$ and the number of rounds to 2 and 3. Some of the data in the following plots is grouped, i. e. for multiple x values is calculated the intersection of them and their associated y values.

## B.1 Linearization attack

For the linearization attack the ANFs of the key stream blocks are generated for different instances. After the computation of the ANFs for a new key stream block, all monomials of a degree of two or higher are renamed and the constant monomial is ignored. Then all ANFs which are computed so far are put in a matrix. After $m$ blocks, the matrix has $n \cdot m$ rows and $2^n - 1$ columns with entries of $\mathbb{F}_2$. Now we try to solve the linear equations with Gaussian elimination. For every row with just a single 1 entry the value of a monomial is revealed. Of course the cipher is broken if all monomials are known, but e. g., knowing only the monomials of degree 1 is also sufficient since they are exactly the key.

The attack works in a known plaintext scenario, where the adversary gets equations with the ANF on the one side and the result i. e. the key stream bit, on the other side. The constant monomial is added to the key stream bit and all row additions in matrix, which are done to eliminate all but one monomial, are done simultaneously to the other sides of the equations.

The desired behavior of the ciphers would be that every ANF occurs with the same probability. This means that we expect to need $\frac{2^l}{2^l-1}$ rows for rank 1 in our matrix, where $l$ is the number of possible non-constant monomials (see Eq. 12), since the first row is linear independent when it is not zero. This occurs with probability $\frac{2^l-1}{2^l}$. For a matrix $\mathbb{F}_2^{t\times l}$ with rank $m$ and $m < l$, we expect to need $\frac{2^l}{2^l-2^m}$ more rows for rank $m+1$. This means that the expected number of uniformly sampled rows of $\mathbb{F}_2^l$ to get rank $l$ is

$$\sum_{i=0}^{l-1} \frac{2^l}{2^l - 2^m} = \sum_{i=0}^{l-1} \frac{1}{1 - 2^{i-l}}.$$

Since every key stream block has $n$ ANFs, the number of expected blocks for 2 and 3 rounds are 52 and 181.

The experiments for 2 rounds showed for all four Rasta variants that the hole system can be solved with 52 blocks, while the first monomials can be solved after 51 blocks. For 3 rounds the result was the same with 180 and 181 blocks. This shows that the linear dependencies between the ANFs are close to the random case and that there are no small subsystems with few monomials.

## B.2 Linear dependencies

The next experiment looks for linear dependencies between multiple ANF blocks. Let an ANF represented as a vector in $\mathbb{F}_2^m$ for $m$ possible monomials (except the constant monomial). A linear combination for a block $b \in (\mathbb{F}_2^m)^n$ is computed by

$$\langle \alpha, b \rangle = \sum_{i=0}^{n-1} \alpha_i \cdot b_i$$

where $\alpha \in \mathbb{F}_2^n$. If the ANFs are sampled uniformly for any linear combination and any block, each monomial occurs with probability 0.5. This means that we expect for a fixed monomial with a degree less or equal than $\min\{2^r, n\}$ over all linear combinations and all blocks a binomial distribution. To define that precisely we need more notation. Let

$$\begin{aligned} &f_u : \mathbb{F}_2^n \to \mathbb{N} \\ &f_u(\alpha) = \mid \{b \in B \mid \langle \alpha, b \rangle_u = 1\} \mid \end{aligned}$$

be the number of blocks which contain the monomial $u \in [m]$ for the linear combination $\alpha$ and the set of ANF blocks of the key stream $B = \{b_0, b_1, \cdots\}$. Let

$$\begin{aligned} &F_u(x) : \ \{0, 1, \cdots, |B|\} \to \{0, 1, \cdots, 2^n - 1\} \\ &F_u(x) = \mid \{\alpha \in \mathbb{F}_2^n \setminus \{\mathbf{0}\} \mid f_u(\alpha) = x\} \mid \end{aligned}$$

be the distribution of how often a fixed monomial occurs in a set of blocks over all non-zero linear combinations. Since we assume that each monomial occurs with probability 0.5 in a non-zero linear combination, we expect the monomial in $\frac{|B|}{2}$ ANFs for one linear combination. For all non-zero linear combinations we expect $F_u$ to behave like the binomial distribution, i. e.

$$\mathbb{E}[F_u(x)] = \binom{|B|}{x} \cdot 2^{-|B|}$$

Figure 7 shows the distribution of $F_u(x)$ for the monomial $k_1 k_0$ and $|B| = 1000$ for all four variants of Rasta and depicts no notable differences between all of them.

In the next step we do not focus just on one monomial, but all possible monomials. Therefore we look at the expected minimum resp. maximum of $f_u$ for an arbitrary monomial $u$ , i. e. we look for

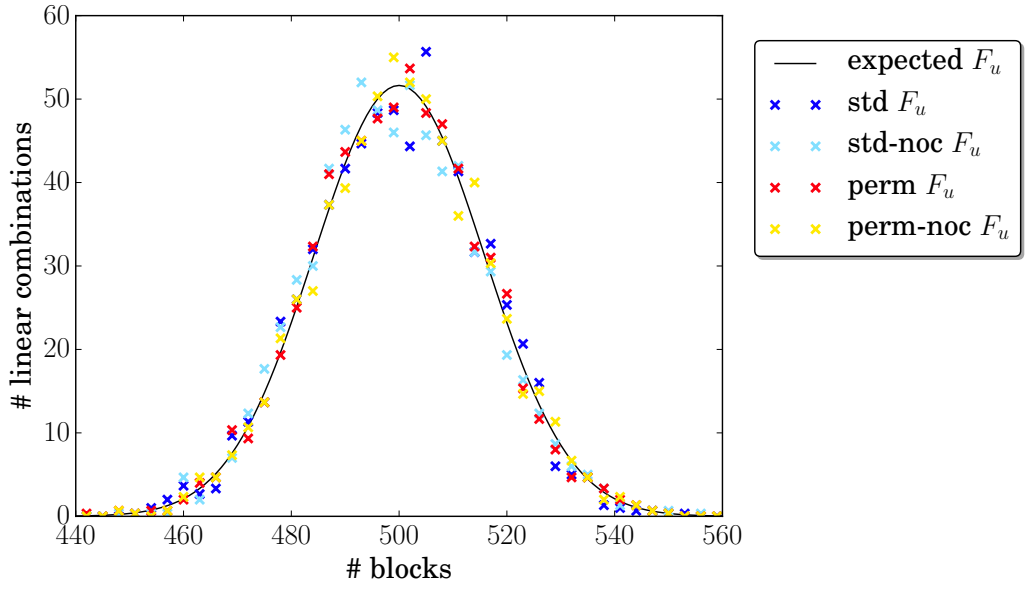


**Figure 7:** Distribution $F_u(x)$ for $|B| = 1000$ and monomial $k_1k_0$ for 2 rounds, grouped by 3.

$$f_{\max}, f_{\min} : \mathbb{F}_2^n \setminus \{\mathbf{0}\} \to \{0, 1, \cdots, |B|\}$$
$$f_{\max}(\alpha) = \max \left\{ x_i \;\middle|\; i \in [m] \wedge x = \sum_{b \in B} \langle \alpha, b \rangle \right\}$$
$$f_{\min}(\alpha) = \min \left\{ x_i \;\middle|\; i \in [m] \wedge x = \sum_{b \in B} \langle \alpha, b \rangle \right\}.$$

where $\langle \alpha, b \rangle$ is considered as a vector of $\mathbb{Z}^m$, so that $x_i \in [|B| + 1]$ for $i \in [m]$.

Now we assume again that all ANFs of the key stream blocks are sampled uniformly. The probability $\Pr[f_{\max}(\alpha) \leq |B|]$ is clearly 1, for $f_{\max}(\alpha) \leq |B| - 1$ holds

$$\Pr[f_{\max}(\alpha) \leq |B| - 1] = \left(1 - 2^{-|B|}\right)^m$$

since $\Pr[x_i \leq |B| - 1] = 1 - 2^{-|B|} \quad \forall i \in [m]$.

We expect $\langle \alpha, b \rangle_i$ to be 1 with probability 0.5 for every $b \in B$. Let $v_i = (\langle \alpha, b_0 \rangle_i, \langle \alpha, b_1 \rangle_i, \cdots, \langle \alpha, b_{|B|-1} \rangle_i)$ be the coefficients for a fixed monomial in all linear combinations, i. e. if the coefficient is one the monomial occurs in the linear combination, if the coefficient is zero the monomial does not occur. There are $2^{|B|}$ possible values for $v_i$. Let $x_i$ be equal to the number of ones in $v_i$. There are

$$\sum_{i=c+1}^{|B|} \binom{|B|}{i}$$

possible values for $v_i$ so that $x_i > c$. This leads to

$$\Pr\left[x_i > c\right] = 2^{-|B|} \cdot \sum_{i=c+1}^{|B|} \binom{|B|}{i}.$$

Saying $f_{\max}(\alpha) \leq c$ is the same as $\nexists i \in [m] : \ x_i > c$. Together with the equation above this can be combined to

$$\Pr\left[f_{\max}(\alpha) \leq c\right] = \left(1 - 2^{-|B|} \cdot \sum_{i=c+1}^{|B|} \binom{|B|}{i}\right)^m.$$

The probability for one function value of $f_{\max}$ is

$$\Pr\left[f_{\max}(\alpha) = c\right] = \Pr\left[f_{\max}(\alpha) \leq c\right] - \Pr\left[f_{\max}(\alpha) \leq c-1\right].$$

With the same argumentation is

$$\Pr\left[f_{\min}(\alpha) \geq a\right] = \left(1 - 2^{-|B|} \cdot \sum_{i=|B|-a+1}^{|B|} \binom{|B|}{i}\right)^m$$
$$\Pr\left[f_{\min}(\alpha) = a\right] = \Pr\left[f_{\min}(\alpha) \geq a\right] - \Pr\left[f_{\min}(\alpha) \geq a+1\right].$$

We define $F_{\max}, F_{\min} : \ \{0, 1, \cdots, |B|\} \to \{0, 1, \cdots, 2^n - 1\}$ as

$$F_{\max}(x) = \mid \{\alpha \in \mathbb{F}_2^n \setminus \{\mathbf{0}\} \mid f_{\max}(\alpha) = x\} \mid$$
$$F_{\min}(x) = \mid \{\alpha \in \mathbb{F}_2^n \setminus \{\mathbf{0}\} \mid f_{\min}(\alpha) = x\} \mid .$$

For uniformly sampled ANFs we expect

$$\mathbb{E}[F_{\max}(x)] = (2^n - 1) \cdot \Pr\left[f_{\max}(\alpha) = a\right]$$
$$\mathbb{E}[F_{\min}(x)] = (2^n - 1) \cdot \Pr\left[f_{\min}(\alpha) = a\right].$$

Figure 8 shows the distribution of $F_{\max}$ and $F_{\min}$ for all of the four variants of Rasta for a random experiment with $|B| = 1000$ compared to the expected distribution. The result shows that all variants are near the expected values, so that the maximal deviation of the occurrence of a monomial in a linear combination behaves close to the random case.

We define the maximal deviation of any monomial over all linear combination for a set of ANF blocks $B = \{b_0, b_1, \cdots\}$ as

$$\delta = \max_{\forall \alpha \in \mathbb{F}_2^n \setminus \{\mathbf{0}\}} \left\{ \left| \frac{|B|}{2} - f_{\max}(\alpha) \right|, \left| \frac{|B|}{2} - f_{\min}(\alpha) \right| \right\}.$$

In Table 7 $\delta$ is computed for a random instance of the Rasta variants with $|B| = 1000$. The results show that there is in the experiment no linear combination for which a monomial occurs very often or very rarely for every Rasta variant. If that happened, an adversary could use that to get small subsystems of monomials and lower the complexity of a linearization attack.

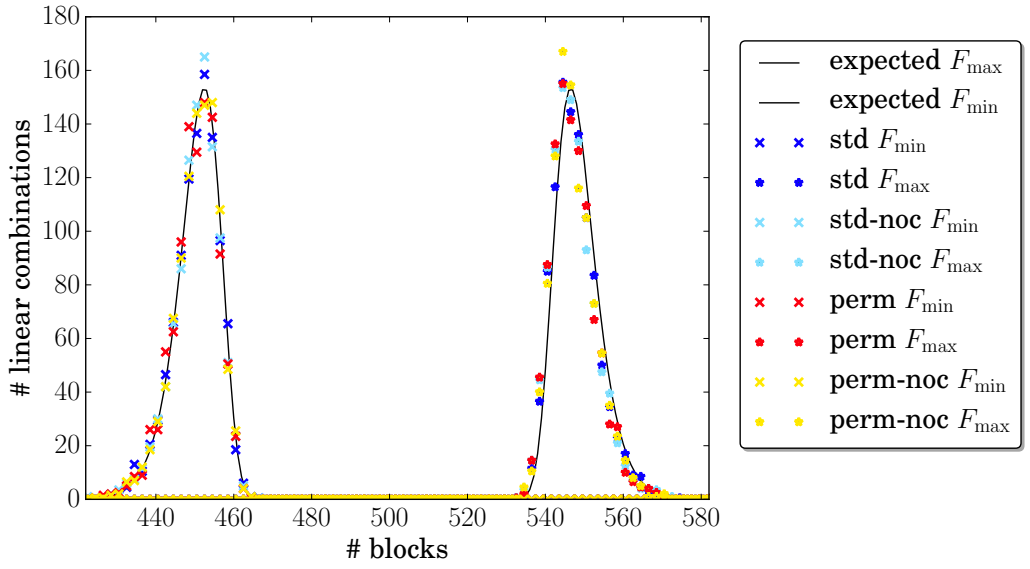


**Figure 8:** Distribution of $f_{\max}$ and $f_{\min}$ for $|B| = 1000$ and 2 rounds, grouped by 2.

**Table 7:** Results of experiment for $|B| = 1000$.

| Rasta variant | rounds | $\delta$ | rounds | $\delta$ |
|---|---|---|---|---|
| std | 2 | 0.078 | 3 | 0.078 |
| std-noc | 2 | 0.078 | 3 | 0.082 |
| perm | 2 | 0.082 | 3 | 0.083 |
| perm-noc | 2 | 0.075 | 3 | 0.080 |

## B.3 Monomial count

Now we look at the number of monomials of the ANFs of the key stream bits. We define by

$$F_c(x) : \{0, 1, \cdots, m\} \to \{0, 1, \cdots, |B|\}$$
$$F_c(x) = |\{b \in B \mid \mathbf{hw}(b) = x\}|$$

the number of ANFs with $x$ monomials in a set of ANFs $B = \{b_0, b_1, \cdots\}$ and $b_i \in \mathbb{F}_2^m$. If the ANFs are sampled uniformly we expect $F_u$ to be binomial distributed, i. e.

$$\mathbb{E}[F_c(x)] = \binom{|B|}{x} \cdot 2^{-|B|}.$$

Figure 9 shows $F_u$ for all four Rasta variants in an experiment with random instances and $|B| = 11000$ compared to the expected distribution. All variants behave close to the random case.

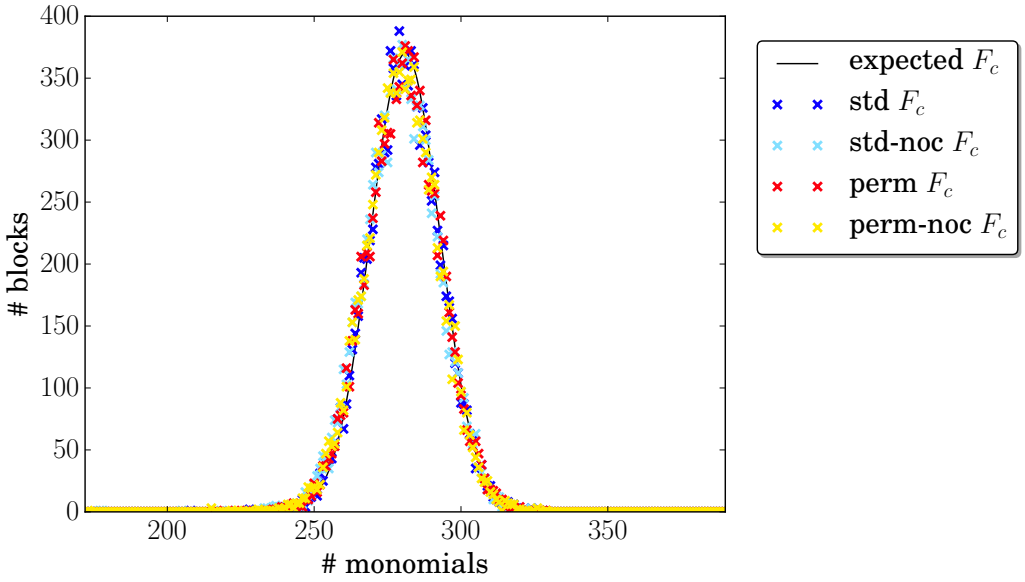


**Figure 9:** Distribution of monomial count for $|B| = 11000$ and 2 rounds.