Twin Column Parity Mixers and Gaston
A New Mixing Layer and Permutation

Solane El Hirch[1], Joan Daemen[1], Raghvendra Rohit[2], and Rusydi H. Makarim

[1] Radboud University, Nijmegen, The Netherlands
[2] Cryptography Research Centre, Technology Innovation Institute, Abu Dhabi, UAE
solane.elhirch@ru.nl, joan@cs.ru.nl, raghvendra.rohit@tii.ae, rusydi@makarim.id

Abstract. We introduce a new type of mixing layer for the round function of cryptographic permutations, called circulant twin column parity mixer (CPM), that is a generalization of the mixing layers in KECCAK-f and XOODOO. While these mixing layers have a bitwise differential branch number of 4 and a computational cost of 2 (bitwise) additions per bit, the circulant twin CPMs we build have a bitwise differential branch number of 12 at the expense of an increase in computational cost: depending on the dimension this ranges between 3 and 3.34 XORs per bit. Our circulant twin CPMs operate on a state in the form of a rectangular array and can serve as mixing layer in a round function that has as non-linear step a layer of S-boxes operating in parallel on the columns. When sandwiched between two ShiftRow-like mappings, we can obtain a columnwise branch number of 12 and hence it guarantees 12 active S-boxes per two rounds in differential trails. Remarkably, the linear branch numbers (bitwise and columnwise alike) of these mappings is only 4. However, we define the *transpose* of a circulant twin CPM that has linear branch number of 12 and a differential branch number of 4. We give a concrete instantiation of a permutation using such a mixing layer, named Gaston. It operates on a state of 5×64 bits and uses χ operating on columns for its non-linear layer. Most notably, the Gaston round function is lightweight in that it takes as few bitwise operations as the one of NIST lightweight standard ASCON. We show that the best 3-round differential and linear trails of Gaston have much higher weights than those of ASCON. Permutations like Gaston can be very competitive in applications that rely for their security exclusively on good differential properties, such as keyed hashing as in the compression phase of Farfalle.

Keywords: Mixing layer · Permutations · Branch number · Column parity mixer (CPM) · ASCON

1 Introduction

Over the last decades there has been extensive research on lightweight cryptographic primitives, where lightweight means with the objective of being implemented on resource-constrained platforms, e.g., for the Internet of Things

(IoT), Radio Frequency ID (RFID) and sensors. Those platforms require a careful trade-off between efficiency and security. A major challenge is to achieve excellent performance results on a wide spectrum of target devices while providing a good security margin, and this at reasonable implementation cost. In hardware implementations and in bit-sliced software implementations a good estimate of the cost is the total number of binary Boolean operations (XOR, (N)AND, (N)OR, NOT) required to execute the primitive. We will refer to this as the *gate cost*.

In recent years, iterated cryptographic permutations have become increasingly popular, but their design is very similar to that of the data path of block ciphers. In so-called substitution-permutation networks (SPN), the round function has a linear layer and a non-linear S-box layer. In an SPN block cipher the rounds are typically alternated with the addition of a round key, derived from the cipher key by means of a key schedule. The linear layer often consists of the composition of a mixing layer and a bit shuffle moving the bits around.

Many SPNs are designed according to the *wide trail strategy* and the best know example of this is the Advanced Encryption Standard (AES) [13]. With this strategy, one can easily prove strong upper bounds for the expected differential probability of differential trails and for the correlation contribution of linear trails. The simplicity of the strategy and this ability to prove trail bounds have made it one of the most widely used design approaches for block ciphers and permutations.

AES [13] operates on a state that can be represented as a 4×4 byte array. Its mixing layer is called MixColumns and it considers each column of the state as a 4-byte vector and it multiplies them by a 4×4-byte matrix. These matrices are maximum-distance-separable (MDS): they have (differential and linear) branch number 5. The AES linear layer also has a byte shuffle, called ShiftRows. Their combination allows to prove that the number of active S-boxes in any 4-round trail is 25, or in general, the square of the branch number [13]. In the context of lightweight cryptography, there has been ample research about constructing MDS matrices with low gate cost, see [3,18,24–26,28,35,37]. While most research is about 4×4 matrices, the first MixColumn-like MDS matrices for use in cryptographic round functions were 8×8, introduced in the block cipher SHARK [34]. Later 8×8 MDS matrices were used in the compression function of hash functions such as Whirlpool [2] and there has also been research on reducing the gate cost of such mappings [24]. Constructions that combine 8×8 MDS matrices with an appropriate ShiftRows mapping have at least $9^2 = 81$ active S-boxes in any 4-round trail. Still, it would be a stretch to call such mappings and ciphers that make use of them lightweight.

A different flavor of the wide trail strategy we find in Ascon-p, the permutation underlying Ascon [17], Keccak-f, the permutation underlying Keccak [6], and the permutation Xoodoo [12]. The former two have a mixing layer of the type *column parity mixer* (CPM) that cannot be split into a number of parallel mappings but operates on the state as a whole. The mixing layer of Ascon-p consists of separate mix mappings operating on the 5 rows of the state independently, combined with two mixing steps that act as enforcements of the low-diffusion linear step in the S-box layer. The latter is the χ mapping borrowed from KECCAK-f, but then operating on 5-bit columns rather than rows.

In all three permutations the mixing layer has a branch number of 4 and hence they have relatively low worst-case diffusion. Moreover, they do not lend themselves to easy proofs for strong lower bounds on the number of active S-boxes over 4 or more rounds. Still, with computer-aided techniques, strong bounds have been achieved for KECCAK-f [32], XOODOO [15] and ASCON-p [19,30].

A study in [9] provides evidence that this approach outperforms MDS-based designs in that they achieve lower upper bounds on the probability of differential trails and the correlation contribution of linear trails for the same amount of computation, often due to lighter rounds. Moreover, the paper shows that they suffer less from clustering of trails.

The mixing layers in XOODOO and KECCAK-f both require 2 binary XOR operations per bit of the state. In the permutation underlying ASCON, winner of the NIST lightweight cryptography cipher competition [33], the mixing layer p_L costs similarly 2 binary XOR operations per bit. However, the part of the mixing layer in the S-boxes p_S has an additional cost of 1.2 binary XOR operations per bit, totaling to 3.2 binary XOR operations per bit. In this split-up, the gate cost of the non-linear layer is equal in all three permutations as all use the χ mapping: 1 XOR, 1 AND and 1 NOT gate per state bit.

Our Contribution. In this work, we explore the possibility of using the mixing budget of 3.2 binary XOR operations per bit, allocated for ASCON-p, differently. We define a new type of mixing layer, the *circulant twin CPM*, a generalization of the column parity mixers in KECCAK-f and XOODOO. Whereas the latter have differential branch number 4, our twin CPM achieves differential branch number 12. Remarkably, its linear branch number is equal to that of the mixing layer of KECCAK-f, XOODOO and ASCON-p, namely 4. We show that we can *transpose* a twin CPM resulting in linear branch number of 12 and differential branch number of 4 with no effect on the gate cost. We provide a proof-of-concept permutation that uses the twin CPM as its mixing layer that we name Gaston. This permutation is lightweight in the sense that it takes the same number of bitwise Boolean operations per round as ASCON-p. We show that Gaston achieves very good differential trail behaviour for both differential, and even linear, trails over 2 and 3 rounds.

Related Work. There is a lot of research that focuses on finding MDS matrices that have low implementation cost. The efficiency of MDS matrices can be defined in terms of several criteria such as the overall gate cost, latency or circuit depth [3,22,23,27,29,35]. Implementing an MDS matrix with branch number 9 (dimension 8) requires more than 6 XORs per bit. We know of no work that investigates the gate cost of MDS matrices or other mixing layers with branch number above 9. At least for MDS matrices existing research suggests that the number of XORs per bit increases with the branch number. In Table 1, we compare various types of mixing layers.

Table 1: A comparison of different types of mixing layers. For MDS, the dimensions in the second column represent the dimension of the matrix (defined over the finite field) while for the rest it denotes the state size in bits. The symbols d, m and n denote the degree of the field defining polynomial, the number of rows and number of columns, respectively. For the first row of CPM, w represents the number of 5×5 slices and $w \in \{8, 16, 32, 64\}$ based on the variant of Keccak-f.

Mixing layer type	Dimensions	XORs per bit	Branch number Diff.	Branch number Linear	Source
MDS	3×3, $GF(2^d)$	$\frac{5}{3} + \frac{1}{3d}$	4	4	[18]
	4×4, $GF(2^d)$	$2 + \frac{3}{4d}$ †	5	5	[18]
	8×8, $GF(2^4)$	6.06	9	9	[24]
	8×8, $GF(2^8)$	6.125	9	9	[24]
CPM	$5 \times 5 \times w$, $GF(2)$	2	4	4	[6]
	$3 \times 4 \times 32$, $GF(2)$	2	4	4	[12]
	$m \times n$, $GF(2)$	$2 + \frac{h-2}{m}$ ‡	4	4	[36]
Ascon p_L	5×64, $GF(2)$	2	4	4	[17]
Twin CPM	$m \times n$, $GF(2)$	$3 + \frac{1}{m}$	12	4	This work
Transpose of Twin CPM	$m \times n$, $GF(2)$	$3 + \frac{1}{m}$	4	12	This work

† : For $d = 4, 8$ this bound is tight as shown in [37]. A 4×4 matrix over $GF(2^4)$ with cost 35 XORs is used in Saturnin [10].
‡ : h is the Hamming weight of the parity-folding polynomial as defined in [36].

Outline. In Sec. 2, we recall different diffusion metrics for linear layers. The circulant twin CPM and the bit shuffles which make up the linear layer are defined in Sec. 3. The study of the differential diffusion properties of twin CPMs is given in Sec. 4 while in Sec. 5 we discuss the equivalence relations that partition the sets of the shift offsets that define the linear layer. We investigate the linear mask properties of twin CPMs in Sec. 6. In Sec. 7, we describe the search strategy for the shift offsets in the twin CPM mixing layer and row shifts of lightweight permutation Gaston and further provide upper bounds on the weight of trails over 2 and 3 rounds of Gaston using general-purpose solvers. In Sec. 8 we finalize the specifications of Gaston and discuss how Gaston and variants can be used in the Farfalle construction. In Sec. 9 we provide conclusions and open problems.

2 Diffusion Metrics for Linear Layers

We will study iterated permutations with a round function R consisting of a linear layer that we will denote by λ and a non-linear layer that we will denote by γ. We will assume $\mathrm{R} = \gamma \circ \lambda$. The other option would be $\mathrm{R} = \lambda \circ \gamma$. When investigating difference (and mask) propagation through the round function the variant can be addressed by a simple re-phasing.

We assume the round function operates on a state A that consists of bits arranged in an array of rows of equal length. We denote the set of all possible states by $\mathcal{A}$. Note that $\mathcal{A}$ forms a vector space with bitwise addition.

Orthogonal to the rows we have *columns*. We assume the non-linear layer operates in parallel on the columns, as in Ascon-p [16] and Xoodoo [12].

Iterating such a round function provides resistance against differential (DC) and linear cryptanalysis (LC) by the alternation of λ and γ. In particular, their combination targets the avoidance of differential trails with high probability and/or linear trails with high correlation contribution.

Within the linear layer there typically is a mixlayer that ensures that a bit at its output depends on multiple bits at its input and that a difference in few input bits propagates to multiple bits at its output. The mixlayer mixes bits *close to each other in the state* and the role of the shuffles is to move bits close to each other to positions that are far from each other. We call the mixlayer θ.

In this section we recall the basics of difference and linear propagation and list some metrics for λ and θ such as the branch number and the branch histogram to provide an indicator of their performance in the round function of a cryptographic permutation or block cipher.

2.1 Difference Propagation

Differential cryptanalysis exploits high-probability differentials [8]. Let f be a transformation over $\mathbb{F}_2^n$ and let a be an input difference to f and b its output difference. The combination of input difference and output difference (a, b) is called a differential over f and its differential probability (DP) is defined as the fraction of all possible input pairs with difference a that exhibit the difference b after application of f to its members:

$$\mathrm{DP}(a, b) = \frac{\#\{x \in \mathbb{F}_2^n \mid f(x) \oplus f(x \oplus a) = b\}}{2^n} .$$

The *restriction weight* $\mathrm{w_r}$ of a differential relates to its DP as $\mathrm{DP} = 2^{-\mathrm{w_r}}$.

We call a differential over the round function R a *round differential*. Round differentials can be chained to form a *differential trail*. An r-round differential trail Q is determined by the sequence of difference patterns before and after each round $(q^0, q^1, \ldots, q^r)$. The DP of a trail is the fraction of all possible input pairs with difference q^0 that exhibit difference q^i after i rounds for all $i \leq r$.

The DP of a trail is in general hard to compute and often approximated by its expected DP (EDP). The EDP of a differential trail is the product of the DP values of its round differentials and is what you would get if the round differentials would act independently: $\mathrm{EDP}(Q) = \prod_{0<i\leq r} \mathrm{DP}(q^{i-1}, q^i)$. The restriction weight of a trail is defined as the sum of the restriction weights of its round differentials and hence $2^{-\mathrm{w_r}(Q)} = \mathrm{EDP}(Q)$.

2.2 Linear Propagation

Linear cryptanalysis exploits linear approximations with high correlation [31]. Let a be a mask that defines the linear Boolean function of input bits $a^{\mathrm{T}}x$ of a transformation f and b one that defines the linear Boolean function of output bits $b^{\mathrm{T}} f(x)$. The combination of input mask and output mask (a, b) is called a *linear approximation* over f and its correlation is determined as the probability p over all inputs x that the linear function defined by a and the linear function defined by b are equal, namely $2p - 1$:

$$\mathrm{C}(a, b) = \frac{\#\{x \in \mathbb{F}_2^n \mid a^{\mathrm{T}} x + b^{\mathrm{T}} f(x) = 0\}}{2^{n-1}} - 1\,.$$

The *correlation weight* $\mathrm{w_c}$ of a linear approximation is defined by $\mathrm{C}^2 = 2^{-\mathrm{w_c}}$.

We call a linear approximation over the round function R a *round linear approximation*. Round linear approximations can be chained to form a *linear trail*. An r-round linear trail Q is determined by the sequence of masks before and after each round $(q^0, q^1, \ldots, q^r)$. The correlation contribution C of a trail is defined as the product of the correlations of its round linear approximations: $\mathrm{C}(Q) = \prod_{0<i\leq r} \mathrm{C}(q^{i-1}, q^i)$. The correlation weight of a linear trail relates to its correlation C as $\mathrm{C}^2 = 2^{-\mathrm{w_c}}$ and is therefore the sum of the correlation weights of its constituent round linear approximations.

2.3 Diffusion Metrics Related to Differences

In this section we discuss diffusion metrics for the propagation of differences. We will indicate differences of state dimensions by the term *state* and its non-zero bits or columns as *active*. Let $w_b(A)$ be the number of active bits in state A and $w_c(A)$ its number of active columns. We call $w_b(A)$ the *bit weight*, $w_c(A)$ the *column weight* of A and denote by L the linear layer.

The concept of branch number is an important metric for the diffusion power of mixing layers. It was introduced in [14] and popularized through [13]. We will generalize it to individual states. For simplicity, we will implicitly assume we are dealing with differential branch numbers and omit the qualification "differential". We discuss the propagation of linear masks in Sec. 6.

Definition 1 (Bit branch number of a state). *The bit branch number of state A with respect to L is the sum of the bit weight of A and that of $L(A)$:*

$$\mathcal{B}_{b,L}(A) = w_b(A) + w_b(L(A))\,.$$

Definition 2 (Column branch number of a state). *The column branch number of state A with respect to L is the sum of the column weight of A and that of $L(A)$:*

$$\mathcal{B}_{c,L}(A) = w_c(A) + w_c(L(A))\,.$$

The well-known branch numbers of a linear mapping are the minimum of the corresponding branch number over all non-zero states. The bit and column branch numbers are given by:

$$\mathcal{B}_b(L) = \min_{A \in \mathcal{A} \setminus \{0\}} \mathcal{B}_{b,L}(A) \quad \text{and} \quad \mathcal{B}_c(L) = \min_{A \in \mathcal{A} \setminus \{0\}} \mathcal{B}_{c,L}(A)\,.$$

The branch numbers of a mapping give only limited information about its diffusion power. We wish to compare several linear mappings according to their diffusion power. High diffusion power means few states with low branch number. In that respect, a more informative quantitative measure of the "immediate" diffusion power is given by the *branch histograms*.

Definition 3 (Bit branch histogram). *The bit branch histogram of L is the histogram indicating the number of states per bit branch number:*

$$H_{b,L}(w) = \#\{A \in \mathcal{A} \text{ with } \mathcal{B}_{b,L}(A) = w\} \, .$$

The bit branch histogram is the appropriate measure for a mixlayer like θ: a mixlayer strives to minimize the number of states with low bit branch number.

Definition 4 (Column branch histogram). *The column branch histogram of L is the histogram indicating the number of states per column branch number:*

$$H_{c,L}(w) = \#\{A \in \mathcal{A} \text{ with } \mathcal{B}_{c,L}(A) = w\} \, .$$

The column branch histogram is the appropriate measure for a linear layer λ: it says something about the number of 2-round trails with a given number of active columns and that lower bounds their weight. Good diffusion corresponds to branch histograms with a *low left tail*.

2.4 Diffusion Metrics Related to Masks

As was shown in [14], the linear branch of a linear mapping specified by a matrix M is equal to the differential branch number of M^{T}, the transpose of M. When talking about linear branch numbers of a linear mapping we will speak about the (differential) branch number of its transpose.

Using the bit and column Hamming weight, we define the linear branch number of a state A with respect to a linear layer L as the sum of the weights of A and that of $L^{\mathrm{T}}(A)$. Depending on the type of weight, we have two types of linear branch number:

$$\mathcal{B}_{b,L^{\mathrm{T}}}(A) = w_b(A) + w_b(L^{\mathrm{T}}(A)) \quad \text{and} \quad \mathcal{B}_{c,L^{\mathrm{T}}}(A) \quad = w_c(A) + w_c(L^{\mathrm{T}}(A)) \ .$$

The linear bit and column branch numbers of a mapping L are defined by

$$\mathcal{B}_b(L^{\mathrm{T}}) = \min_{A \in \mathcal{A} \setminus \{0\}} \mathcal{B}_{b,L^{\mathrm{T}}}(A) \quad \text{and} \quad \mathcal{B}_c(L^{\mathrm{T}}) = \min_{A \in \mathcal{A} \setminus \{0\}} \mathcal{B}_{c,L^{\mathrm{T}}}(A) \ .$$

The histograms are defined analogously.

3 Circulant Twin Column Parity Mixers and Row Shifts

The round function operates on a state A with m rows and n columns, that we denote as A_i with $0 \leq i < m$. The bit in row i and column j is denoted as $a_{i,j}$.

As we wish our round function to be suited for software implementations, we will assume the row length n is a power of two: $n = 2^\ell$. Still, some of the lemmas we prove are also valid if that is not the case.

We limit our choice of the steps of the round function, except for the round constant addition, to mappings that commute with a cyclic shift of the state in the horizontal direction. Such a property has been called shift-invariant, translation-invariant or rotation-invariant but we will use the term *circulant*. More formally, if τ is a mapping that shifts the state one position to the left, then a mapping α is circulant if $\tau \circ \alpha = \alpha \circ \tau$. Circulance partitions the state space into classes where a class contains all shifted versions of a given state. For the vast majority of states these classes contain n members. In our investigations on propagation, we can restrict ourselves to one representative of each class. We adopt a convention how to choose that representative and call that *canonical*.

For γ, circulance simply means that it applies the same S-box to all columns. For λ we assume it is a mixlayer sandwiched between two cyclic row shift steps as in Xoodoo [12] and all three are circulant.

3.1 Cyclic Row Shifts

Similar to ShiftRows in Rijndael [13], the round function of Gaston has two bit shuffles that cyclically shift the bits within the rows. We call them ρ_{west} and ρ_{east} and we specify them as follow:

$$\begin{aligned}\rho_{\text{west}} &: A_i \leftarrow (A_i \lll w_i), \text{ for } 0 \le i < m \\ \rho_{\text{east}} &: A_i \leftarrow (A_i \lll e_i), \text{ for } 0 \le i < m \ .\end{aligned}$$

Here $(C \lll r)$ denotes row C shifted over offset r, i.e., moving the bit in position j to position $j + r \bmod n$. Sometimes we write $\tau^r(C)$ for $(C \lll r)$.

Each row shift step is parameterized by m offsets, hence there are $2m$ shift offset parameters: w_0 to w_{m-1} and e_0 to e_{m-1}. In the diffusion properties only the differences between the offsets of ρ_{west} (or ρ_{east}) matter and therefore we can fix one offset for each of them to 0. We set $w_0 = 0$ and $e_0 = 0$, and hence to fully specify the row shift steps we must specify $2(m-1)$ offsets. We will denote the array of shift offsets for ρ_{west} by $\mathbf{R}_w$, the array of shift offsets for ρ_{east} by $\mathbf{R}_e$ and their combination by $\mathbf{R}_\rho$.

3.2 Circulant Twin Column Parity Mixers

The mixlayer θ is what we call a *circulant twin column parity mixer* or twin CPM for short. This is a variant of the column parity mixers as in Xoodoo and Keccak-f. A circulant column parity mixer applied to our two-dimensional state would look like this:

$$A_i \leftarrow A_i + (E \lll u) \text{ for } 0 \le i < m, \text{ with } E \leftarrow (P + (P \lll r)) \text{ and } P \leftarrow \sum_{i=0}^{m-1} A_i \ . \tag{1}$$

This mapping computes the bitwise sum of all rows, called the *column parity* P, *folds* it to the θ-effect E and adds that to each row after shifting over some offset u. Here folding consists of adding P and a shifted copy of P. The two shift offsets r and u would be parameters. The CPM of (1) has a computational cost of two XORs per bit. CPMs and their properties were studied in [36].

A circulant twin column parity mixer looks like this:

$$\begin{aligned} A_i \leftarrow A_i + (E \lll u) \text{ for } 0 \leq i < m \\ \text{with } E \leftarrow (P + (P \lll r)) + (Q + (Q \lll s)), \\ \text{and } P \leftarrow \sum_{i=0}^{m-1} A_i \text{ and } Q \leftarrow \sum_{i=0}^{m-1} (A_i \lll t_i)\,. \end{aligned} \tag{2}$$

In (2), next to the column parity P, there is an additional column parity Q, obtained by adding the rows where each one is first shifted over a row-specific offset t_i. Both parities P and Q are folded and the θ-effect E consists of the sum of the two folded parities. The twin CPM of (2) has a computational cost of $3+1/m$ XORs per bit, hence 3.2 for 5 rows and 3.333 for 3 rows. Like the row shift steps, the twin CPM (2) is also a parameterized mapping. Its parameters are: the two folding offsets r and s, the θ-effect addition offset u, and the m offsets t_i for computing parity Q. This totals to $m+3$ shift offset parameters. We denote the array of shift offsets for θ by $\mathbf{R}_\theta$ and the combination of $\mathbf{R}_\theta$ with those of the row shift steps $\mathbf{R}_\rho$ by $\mathbf{R}_\lambda$. We divide $\mathbf{R}_\theta$ into two parts: the ones that determine the distribution of the bit weight of the θ-effect that we group in $\mathbf{R}_E = (s, t_0, t_1, t_2, t_3, t_4)$ and u.

3.3 Polynomial Representation

The mixlayer θ and the row shifts ρ_{west} and ρ_{east} are specified in terms of two operations: bitwise addition of rows and cyclic shifts. These operations lend themselves to a representation of the rows of the state as polynomials with coefficients in $\mathbb{F}_2$. We denote row i of a state by

$$A_i(X) = \sum_j a_{i,j} X^j \text{ for } 0 \leq i < m\,.$$

These are elements of $\mathbb{F}_2[X]$, the ring of binary polynomials. Bitwise addition of rows is just the addition of polynomials in $\mathbb{F}_2[X]$. A cyclic shift of a row A_i over an offset r corresponds with the multiplication of A_i by the polynomial X^r modulo $1+X^n$. It follows that we are working in the ring of binary polynomials modulo $1+X^n$: $\mathbb{F}_2[X]/(1+X^n)$ and a state is a vector of m polynomials, hence the state space $\mathcal{A}$ is $(\mathbb{F}_2[X]/(1+X^n))^m$.

In polynomial representation, the column parity mixer of (1) becomes:

$$A_i \leftarrow A_i + \sum_{j=0}^{m-1} \left(X^u + X^{u+r}\right) A_j \text{ for } 0 \leq i < m\,.$$

Here $(X^u + X^{u+r})\,A_j$ denotes the multiplication of the constant polynomial $(X^u + X^{u+r})$ by a variable polynomial A_j modulo $1 + X^n$.

The circulant twin CPM of (2) becomes:

$$A_i \leftarrow A_i + X^u \sum_{j=0}^{m-1} \left(1 + X^r + X^{t_j} + X^{s+t_j}\right) A_j \text{ for } 0 \leq i < m\,.$$

We will denote the constant polynomials in this expression as ρ_j, hence

$$A_i \leftarrow A_i + X^u E \text{ with } E = \sum_{j=0}^{m-1} \rho_j A_j \text{ and } \rho_j = 1 + X^r + X^{t_j} + X^{s+t_j}\,. \quad (3)$$

The polynomial representation of state and linear mappings can be quite convenient in demonstrating properties. For example, $A_i(x)$ is divisible by 1+X iff $A_i(1) = 0$, that is, it has an even number of active bits.

3.4 The Inverse of a Circulant Twin CPM

Equation (3) represents m equations, one for each row A_i of the state. We can rewrite it as a matrix equation where we represent the state A as an m-dimensional vector with coordinates the rows polynomials. We denote the vector containing the polynomials $R_i = X^u\rho_i$ by R and write $\mathbf{1}$ for an m-dimensional vector with all coordinates equal to 1. Denoting the image by B, this gives:

$$B = A + \mathbf{1}R^{\mathrm{T}}A = (\mathbf{I} + \mathbf{1}R^{\mathrm{T}})A\,,$$

with $\mathbf{I}$ an $m \times m$ unit matrix. For symmetry reasons, the inverse has a similar shape: $A = (\mathbf{I} + \mathbf{1}R'^{\mathrm{T}})B$, where R' is the vector of polynomials for the inverse. Substitution results in: $B = (\mathbf{I}+\mathbf{1}R^{\mathrm{T}})(\mathbf{I}+\mathbf{1}R'^{\mathrm{T}})B$ or simplified $\mathbf{I} = (\mathbf{I}+\mathbf{1}R^{\mathrm{T}})(\mathbf{I}+\mathbf{1}R'^{\mathrm{T}})$. Working this out yields:

$$\begin{aligned} 0 = \mathbf{1}R^{\mathrm{T}} + \mathbf{1}R'^{\mathrm{T}} + \mathbf{1}R^{\mathrm{T}}\mathbf{1}R'^{\mathrm{T}} &= \mathbf{1}R^{\mathrm{T}} + \mathbf{1}R'^{\mathrm{T}} + \mathbf{1}\left(\sum_j R_j\right) R'^{\mathrm{T}} \\ &= \mathbf{1}R^{\mathrm{T}} + \mathbf{1}\left(1 + \sum_j R_j\right) R'^{\mathrm{T}}\,. \end{aligned}$$

Taking the i-th component yields $R_i = \left(1 + \sum_j R_j\right) R'_i$, hence R'_i is given by

$$R'_i = \left(1 + X^u \sum_j \rho_j\right)^{-1} X^u \rho_i \quad .$$

As the polynomials ρ_j are divisible by $1+X$, the expression between the brackets is coprime to $1 + X$ and hence to $1 + X^n$, hence its inverse exists for any choice of the shift offsets if $n = 2^\ell$. Moreover, this inverse is in general dense, similar to the inverse of θ in Keccak-p.

4 Differential Diffusion Properties of Twin CPMs

In this section we study structural properties of twin CPMs that are a function of their shift offsets. This leads to necessary conditions for achieving branch number of 12 and suggests choices leading to a low left tail in the branch histogram. We will group these conditions in a number of condition sets governing the shift offsets. They have the form of inequalities and are always modulo n.

For a state A that has $E = 0$ in (2), a twin CPM acts like the identity. Its bit branch number is simply its number of active bits times two. The set of states with $E = 0$ forms a subspace of $\mathcal{A}$ and plays an important role in this section.

Definition 5 (θ-effect kernel). *The subspace of $\mathcal{A}$ of states that have $E = 0$ in (2) for a twin CPM is its θ-effect kernel.*

We will refer to the θ-effect kernel simply as the kernel. States in the kernel with few active bits have a low branch number and are therefore undesirable.

In this section we start by characterizing the kernel, then we discuss unavoidable states in the kernel, followed by states that are avoidable by a good choice of shift offsets and finally states with low branch number outside the kernel.

4.1 The Kernel and Its Dimension

We define the P-kernel as the subspace of $\mathcal{A}$ for which $P = 0$ in (2) and the Q-kernel analogously. The dimension of the P-kernel is $(m-1)n$. It is namely the subspace of $\mathcal{A}$ that satisfies n independent linear equations: the parity of each column shall be 0. The dimension of the Q-kernel is likewise $(m-1)n$ as it is also a subspace of $\mathcal{A}$ that satisfies n independent linear equations.

We call the intersection of the P-kernel and the Q-kernel the PQ-kernel. Clearly the PQ-kernel is a subspace of the θ-kernel. Its dimension depends on the offsets t_i but is at least $(m-2)n+1$. This is because the sum of all conditions of the P-kernel is the sum of all conditions of the Q-kernel.

We call states in P-kernel with the lowest number of active bits, namely 2, P-orbitals and similar states in the Q-kernel we call Q-orbitals.

Definition 6 (P-orbital). *A P-orbital is a state with 2 active bits that are in the same column: it has $A_i = X^q$ and $A_j = X^q$ for some q.*

Definition 7 (Q-orbital). *A Q-orbital is a state with 2 active bits contributing to the same bit in Q: it has $A_i = X^{q+t_j}$ and $A_j = X^{q+t_i}$ for some q.*

If for two rows i and j, we have $t_i = t_j$, then the state with two active rows $A_i = 1$ and $A_j = 1$ is at the same time a P-orbital and a Q-orbital, and therefore in the kernel. A 2-bit state in the kernel implies a branch number of at most 4 and clearly we want to avoid it. Therefore, we will take all offsets t_i different.

Condition set 1. *Conditions to avoid single-orbital states in the kernel:*

$$t_i \neq t_j, \forall\{i,j\} \subset \{0,1,\ldots,m-1\} \ .$$

The following lemma says something about the dimension of the kernel if the number of columns is a power of two: $n = 2^\ell$.

Lemma 1. *Let $(1+X)^d = \gcd(\rho_0, \rho_1, \ldots, \rho_{m-1}, 1+X^n)$ with $n = 2^\ell$, then the dimension of the kernel is $(m-1)n + d$.*

Proof. As $n = 2^\ell$ we have $1 + X^n = 1 + X^{2^\ell} = (1+X)^{2^\ell} = (1+X)^n$. Then there exists a j such that $(1+X)^d = \gcd(\rho_j, 1+X^n)$. We can do a renumbering such that $\gcd(\rho_0, 1+X^n) = (1+X)^d$, therefore this is without loss of generality. We now rewrite

$$\sum_{j>0} \rho_j A_j = \rho_0 A_0 \,. \tag{4}$$

The number of different solutions of (4) determines the dimension of the kernel. We can choose A_j in the lefthand side freely, resulting in some value B and try to solve $B = \rho_0 A_0$ for A_0. As all ρ_j are divisible by $1 + X^d$, so is B. Hence, we can divide both sides by $1 + X^d$ resulting in an equation $B' = \pi A_0$ with $B' = B/(1+X)^d$ and $\pi = \rho_0/(1+X)^d$. As π is coprime to $1 + X^n$ it has an inverse and the solution is given by $A_0 = \pi^{-1}B'$. Moreover, if C is a solution for A_0 in (4), then $C + \beta(X)(1+X)^{n-d}$ is also a solution as $\rho_0(1+X)^{n-d} = 0$. The term $\beta(X)(1+X)^{n-d}$ can take on 2^d possible values, namely those obtained by taking for β the polynomials of degree less than d. Hence, the choice of the A_j in the lefthand side accounts for dimension $(m-1)n$ and the 2^d values of A_0 per element of that vector space adds d to the dimension. $\square$

4.2 Minimizing the Kernel

By choosing our offsets well, we can ensure that all states in the kernel have an even number of active bits. First in Lemma 2, we link the shape of the polynomials ρ_j with the number of factors $1+X$ they contain. Then in Lemma 3 we prove that the kernel contains only states with an even number of bits if the polynomials ρ_i satisfy a certain condition.

Lemma 2. *Let $n = 2^\ell$ and $0 \le a < b < c < d < n$. Then*

$$\gcd(X^a + X^b + X^c + X^d, 1 + X^n) = 1 + X \iff a + b + c + d \bmod 2 = 1\,,$$

i.e., if $\{a, b, c, d\}$ has an odd number of odd integers.

Proof. Dividing $X^a + X^b + X^c + X^d$ by $1 + X$ gives $\sum_{a \le i < b} X^i + \sum_{c \le i < d} X^i$. This polynomial is only divisible by $1 + X$ if it has an even number of terms. This expression has an even number of terms iff $b - a$ and $d - c$ are both odd or both even, or equivalently iff $a + b + c + d$ is even. $\square$

Lemma 3. *If $n = 2^\ell$ and for all polynomials ρ_j the GCD with $1 + X^n$ is the same: $\gcd(\rho_j, 1+X^n) = 1+X^d$ with $d < n$, all states in the kernel have an even number of active bits.*

Proof. If $\gcd(\rho_j, 1+X^n) = (1+X)^d$ then $\rho_j \bmod (1+X)^{d+1} = (1+X)^d$. We now have

$$\begin{aligned}
E \bmod (1+X)^{d+1} &= \left(\sum_i \rho_i A_i\right) \bmod (1+X)^{d+1} \\
&= \left(\sum_i (1+X)^d A_i\right) \bmod (1+X)^{d+1} \\
&= \left((1+X)^d \sum_i A_i\right) \bmod (1+X)^{d+1} \\
&= (1+X)^d \left(\left(\sum_i A_i\right) \bmod (1+X)\right) .
\end{aligned}$$

If A has odd bit weight, $(\sum_j A_j) \bmod (1+X) = 1$ and therefore $E \neq 0$. □

We now give a corollary with a sufficient condition for minimizing the kernel.

Corollary 1. *For $n = 2^\ell$, if in θ we have $r + s \bmod 2 = 1$, then the dimension of kernel is $(m-1)n+1$ and only contains states with an even number of active bits.*

Proof. If $r + s \bmod 2 = 1$, then for all polynomials ρ_j we have $(0 + r + t_j + (s + t_j)) \bmod 2 = 1$. Due to Lemma 2 this implies that for all ρ_j we have $\gcd(\rho_j, 1+X^n) = 1+X$. It follows from Lemma 3 that the kernel only contains states with an even number of active bits and from Lemma 1 that the dimension of the kernel is $(m-1)n+1$. □

4.3 Row Twins

Irrespective of the choice of shift offsets, the kernel contains states with 8 active bits. These states have a specific structure and we call them *row twins*.

Definition 8. *A state A is a row twin if it only has two active rows with indices in $\{i, j\} \subset \{0, 1, \ldots, m-1\}$ and*

$$A_i = \rho_j = 1 + X^r + X^{t_i} + X^{s+t_i},\ A_j = \rho_i = 1 + X^r + X^{t_j} + X^{s+t_j} ,$$

or if it is a shifted version of such a state.

Lemma 4. *Row twins are in the kernel.*

Proof. From (3) we have $E = \rho_i A_i + \rho_j A_j = \rho_i \rho_j + \rho_j \rho_i = 0$. □

There are $\binom{m}{2} = m(m-1)/2$ canonical row twins. To avoid row twins with less than 8 active bits, each row in a row twins shall have 4 active bits implying that the 4 powers in ρ_j shall be different. This results in a number of conditions for the shift offsets of θ.

Condition set 2. *Conditions to avoid row twins with less than 8 active bits:*

$$\begin{aligned}
&r \neq 0,\ s \neq 0 \\
&t_i \neq 0,\ t_i \neq r,\ t_i \neq -s,\ t_i \neq r - s \qquad \forall i \in \{0, 1, \ldots, m-1\}
\end{aligned}$$

4.4 Vortices

Irrespective of the choice of shift offsets, the kernel contains states with 6 active bits. We call these *vortices*, after states with similar structure in Keccak-f.

Definition 9. *A state A is a vortex if it only has three active rows with indices in $\{i, j, k\} \subset \{0, 1, \ldots, m-1\}$ and*

$$A_i = X^{t_j} + X^{t_k},\ A_j = X^{t_k} + X^{t_i},\ A_k = X^{t_i} + X^{t_j}\ ,$$

or if it is a shifted version of such a state.

Lemma 5. *Vortices are in the kernel.*

Proof. As $P = A_i + A_j + A_k = 0$, the vortex is in the P-kernel, and as $Q = X^{t_i} A_i + X^{t_j} A_j + X^{t_k} A_k = 0$, it is in the Q-kernel. Hence it is in the θ- kernel. □

As the indices of the active rows completely determine the shape of a vortex, there are $\binom{5}{3} = 10$ canonical vortices. Due to the existence of vortices, the binary branch number of any circulant twin CPM is at most 12.

4.5 Avoiding States in the Kernel with Less than 6 Active Bits

The Condition sets 1 are covered by a more general set of conditions to avoid states in the kernel with 4 active bits. There are three types of such states and we will cover them in the following three lemmas.

Lemma 6. *For even n, let $\{i, j\} \subset \{0, 1, \ldots, m-1\}$ and $t_i = t_j + n/2$, then the state A with two active rows $A_i = A_j = 1 + X^{n/2}$ is in the kernel.*

Proof. The state is in the P kernel as $P = (1 + X^{n/2}) + (1 + X^{n/2}) = 0$. It is also in the Q-kernel as $Q = X^{t_j}(1 + X^{n/2}) + (X^{t_j + n/2})(1 + X^{n/2}) = 0$. □

Lemma 7. *Let $\{i, j, k\} \subset \{0, 1, \ldots, m-1\}$ and $t_i + t_j = 2t_k$, then the state A with three active rows $A_i = X^{t_k}$, $A_j = X^{t_i}$ and $A_k = X^{t_k} + X^{t_i}$ is in the kernel.*

Proof. The state is in the P-kernel as $P = X^{t_k} + X^{t_i} + X^{t_k} + X^{t_i} = 0$. It is also in the Q-kernel as $Q = X^{t_i + t_k} + X^{t_j + t_i} + X^{t_k + t_k} + X^{t_k + t_i} = 0$. □

Lemma 8. *Let $\{i, j, k, l\} \subset \{0, 1, \ldots, m-1\}$ and $t_i + t_j = t_k + t_l$, then the state with 4 active rows $A_i = 1$, $A_j = X^{t_k - t_j}$, $A_k = 1$ and $A_l = X^{t_i - t_l}$ is in the kernel.*

Proof. The state is in the P-kernel as $1 + X^{t_k - t_j} + 1 + X^{t_i - t_l} = 0$. It is also in the q-kernel as $A_i = X^{t_i} + X^{t_k} + X^{t_k} + X^{t_i} = 0$. □

These three lemmas combined with Condition sets 1 result in three types of conditions that we group in the following set.

Condition set 3. *To avoid in-kernel states with 4 active bits (or 2 active bits). For any set* $\{i,j\} \subset \{0,1,\ldots,m-1\}$ *(for even* n*):*

$$2t_i \neq 2t_j \, .$$

For any sets $\{i,j\} \subset \{0,1,\ldots,m-1\}$ *and* $k \in \{0,1,\ldots,m-1\}$ *with* $k \notin \{i,j\}$*:*

$$t_i + t_j \neq 2t_k \, .$$

For any two sets $\{i,j\} \subset \{0,1,\ldots,m-1\}$ *and* $\{k,l\} \subset \{0,1,\ldots,m-1\}$ *and* $\{i,j\} \cap \{k,l\} = \varnothing$*:*

$$t_i + t_j \neq t_k + t_l \, .$$

4.6 States with Low Branch Number outside the Kernel

We now discuss structures that result in states outside the kernel with low branch number. If the θ-effect E of a state has d active bits, the state has (bit) branch number of at least md.

This is because each active bit in E is added to m rows and if in a row it is added to a passive bit, that bit is active after θ and vice versa, hence it will always contribute 1 to the bit branch number. As the polynomials ρ_j all have even parity, so does E and therefore the smallest non-zero value of d is 2. In that case the branch number is at least $2m$. If E has more than 2 active bits, it has at least 4 and the branch number is at least $4m$. In the following, we discuss states with a low number of active bits before θ with 2 active bits in E.

Each of the polynomials ρ_j has 4 active bits and therefore for any single-bit state A, E has 4 active bits. A state A outside the kernel with two active bits may have $2, 4, 6$ or 8 active bits in E. P- and Q-orbitals lead to at most 4 active bits in E.

A P-orbital with active bits in $(0,i)$ and $(0,j)$ has $A_i = 1$ and $A_j = 1$ and 0 in other rows, and yields:

$$E = \rho_i + \rho_j = X^{t_i} + X^{s+t_i} + X^{t_j} + X^{s+t_j} \, . \tag{5}$$

A Q-orbital with active bits in (t_i, j) and (t_j, i) has $A_i = X^{t_j}$ and $A_j = X^{t_i}$ and 0 in the other rows and yields:

$$E = X^{t_j}\rho_i + X^{t_i}\rho_j = X^{t_j} + X^{r+t_i} + X^{t_i} + X^{r+t_j} \, . \tag{6}$$

If two exponents in (5) collide, we obtain an E with two active bits. The same holds for (6) and this leads to $2\binom{n}{2}$ more conditions.

Condition set 4. *Conditions for* P*-orbitals or* Q*-orbitals having* E *with two active bits. For all* $\{i,j\} \in \{0,1,\ldots,m-1\}$*:*

$$t_i \neq t_j + r$$
$$t_i \neq t_j + s \, .$$

Notice that we can build a state with 3 active bits that has at most 4 active bits in E by taking for the first two bits a Q-orbital and the last two bits a P-orbital, or vice versa. For example, let the 3 active bits be at (t_j, i), (t_i, j) and (t_i, k), then we have:

$$E = X^{t_j}\rho_i + X^{t_i}\rho_j + X^{t_i}\rho_k = X^{t_j} + X^{r+t_j} + X^{t_k+t_i} + X^{s+t_k+t_i}\,. \qquad (7)$$

This can be seen as a chain: a Q-orbital followed by a P-orbital. Note that k can be equal to i.

A P-orbital followed by a Q-orbital: (t_k, i), (t_k, j) and (t_j, k), gives:

$$E = X^{t_k}\rho_i + X^{t_k}\rho_j + X^{t_j}\rho_k = X^{t_j} + X^{r+t_j} + X^{t_i+t_k} + X^{s+t_i+t_k}\,. \qquad (8)$$

These chains can be generalized by building longer sequences of alternating P-orbitals and Q-orbitals. The number of active bits grows with the chain length and hence the longer the chains are, the less threatening.

5 Equivalence Classes of Shift Offset Vectors

In this section we discuss two equivalence relations that partition the sets of shift offset vectors $\mathbf{R}_\lambda$. The equivalence stems from alternative representations of the state that lead to linear mappings λ with the same bit branch and column branch histograms. These representations change the indexes of the bits, but preserve grouping of bits in columns.

If we denote the alternative representation of a state A by A', then we have $A' = \sigma(A)$ with σ a bit shuffle. Clearly, $B = \lambda(A)$ implies $B' = \sigma(B) = \sigma(\lambda(A)) = \sigma(\lambda(\sigma^{-1}(A')))$. In other words, $B' = \lambda'(A')$ with $\lambda' = \sigma \circ \lambda \circ \sigma^{-1}$. As σ is a bit shuffle, the bit branch number of A with respect to λ is equal to that of A' with respect to λ'. Moreover, if σ preserves the grouping of bits in columns, this is also true for the column branch number.

We will now describe two groups of bit shuffles whose elements convert a mapping $\lambda = \rho_{\text{west}} \circ \theta \circ \rho_{\text{east}}$ characterized by some offset vector $\mathbf{R}_\lambda$ to a linear mapping of the same shape but with a different offset vector.

5.1 Multiplicative Factor Equivalence

The first group of bit shuffles rearranges the bits within the rows of a state by multiplying their horizontal indexes with a fixed constant q modulo n. Expressions in the horizontal index shall be taken modulo n.

Definition 10 (Multiplicative shuffle). *Let $q \in (\mathbb{Z}/n\mathbb{Z})^*$, then π_q is the permutation over domain $\mathbb{F}_2^n$ defined by:*

$$A' = \pi_q(A) \Leftrightarrow \forall j \in \mathbb{Z}/n\mathbb{Z} : a'_{qj} = a_j\,,$$

with A a row and a_j its component at index j. We call π_q the multiplicative shuffle with shuffling factor q (operating on n-bit vectors).

Clearly, as q is coprime to n, π_q has an inverse. It is $\pi_{q^{-1}}$ with q^{-1} the multiplicative inverse of q in $(\mathbb{Z}/n\mathbb{Z})^*$.

Lemma 9. *Let $A' = \pi_q(A)$ and $B' = \pi_q(B)$. Then, if $B = (A \lll t)$ we have $B' = (A' \lll qt)$, or equivalently, $\pi_q \circ \tau^t \circ \pi_{q^{-1}} = \tau^{qt}$.*

Proof. Using $A' = \pi_q(A)$, $B' = \pi_q(B)$ and $B = (A \lll t)$ yields:

$$\forall i \in \mathbb{Z}/n\mathbb{Z} : b'_{qi} = b_i = a_{i+t} = a'_{q(i+t)} = a'_{qi+qt} .$$

A change of variable $j \leftarrow qi$ results in $\forall j \in \mathbb{Z}/n\mathbb{Z} : b'_j = a'_{j+qr}$, in other words $B' = (A' \lll qt)$. □

We can define a multiplicative shuffle on a rectangular state with m rows by applying it to all its rows. Clearly, it preserves the grouping of bits in columns.

A linear layer λ with some offset vector $\mathbf{R}_\lambda$ only makes use of cyclic shifts and row additions and therefore $\pi_q \circ \lambda \circ \pi_{q^{-1}}$ is of the same type but has an offset vector that is given by $q\mathbf{R}_\lambda$, i.e., $\mathbf{R}_\lambda$ with all entries multiplied by q modulo n.

As offset vectors $\mathbf{R}_\lambda$ and $q\mathbf{R}_\lambda$ with q coprime to n are equivalent, we can limit investigating variants with a given entry fixed to some constant value. In this respect the following lemma is useful.

Lemma 10. *Assume $n = 2^\ell$ and q is coprime to n. Then if r is of the form $u2^d$ with u odd, $qr \bmod n$ is of the form $v2^d$ with v odd.*

Proof. We have:

$$qu2^d \bmod 2^\ell = (qu \bmod 2^\ell)(2^d \bmod 2^\ell) \bmod 2^\ell = (qu \bmod 2^\ell)2^d \bmod 2^\ell$$

The group $(\mathbb{Z}/2^\ell\mathbb{Z})^*$ consists of the odd integers $< 2^\ell$. As the product of two group elements is a group element, $qu \bmod 2^\ell \in (\mathbb{Z}/2^\ell\mathbb{Z})^*$ and hence odd. □

Corollary 2. *Multiplication of shift offset by an invertible factor q modulo $n = 2^\ell$ preserves its parity: it maps odd to odd and even to even.*

5.2 Implications for $\mathbf{R}_\theta$

Let us now focus on θ, in particular $\mathbf{R}_E$. According to Corollary 1, the kernel is minimized by taking r odd and s even or vice versa. Let us assume r is odd. Then we can fix its value to 1 in the offset vector $\mathbf{R}_E$ without loss of generality: we can obtain an offset vector with any possible odd value q for r by just multiplying by q. We can limit s to even values in the interval $[2, n/2]$. It starts at 2 as according to Condition sets 2 we wish $s \neq 0$. The end of the interval is $n/2$ due to the following reason. The two terms in ρ_i that depend on s are X^{t_i} and X^{s+t_i}. Assuming the offsets t_i range over all possible values, X^a and X^b can be obtained in two different ways: $t_i = a$ and $s = b - a \bmod n$ or $t_i = b$ and $s = a - b \bmod n$. The values of $b - a$ and $a - b$ cannot both be above n and hence without loss of generality we can set the limit $s \leq n/2$.

These restrictions on r and s reduce the number of options for $\mathbf{R}_E$ from n^7 to less than $n^6/2$. When we add u to $\mathbf{R}_E$, we obtain $\mathbf{R}_\theta$ and it turns out that there are pairs of offset vectors $\mathbf{R}_\theta$ that result in equivalent mappings θ.

Lemma 11. *The following offset vectors yield θ mappings that are equivalent*

$$\mathbf{R}_\theta = [1, a, b_0, \ldots, b_{m-1}, c]$$
$$\mathbf{R}'_\theta = [1, a, 1 - b_0 - a, \ldots, 1 - b_{m-1} - a, -c - 1]\,.$$

Proof. Consider $R_j = X^u \rho_j = X^u + X^{u+r} + X^{u+t_j} + X^{u+s+t_j}$. Filling in $\mathbf{R}_\theta$:

$$R_j = X^c + X^{c+1} + X^{c+b_j} + X^{c+a+b_j}\ ,$$

and filling in $\mathbf{R}'_\theta$:

$$\begin{aligned} R'_j &= X^{-c-1} + X^{-c} + X^{-c-1+1-b_j-a} + X^{-c-1+1-b_j} \\ &= X^{-c} + X^{-c-1} + X^{-c-b_j} + X^{-c-b_j-a}\,. \end{aligned}$$

Composition with π_{-1} yields $R''_j = X^c + X^{c+1} + X^{c+b_j} + X^{c+b_j+a} = R_j$. □

Note that the offset t_i in $\mathbf{R}'_\theta$ is even if t_i in $\mathbf{R}_\theta$ is odd and vice versa. For the case of odd m, we can avoid investigating equivalent offset vectors by requiring the number of odd t_i values in a canonical offset vector to be odd.

5.3 Row Order Equivalence

A row shuffle is a bit shuffle that keeps the order of the bits in the rows intact but changes the order of the rows. Naturally it preserves both bit and column branch numbers. As there are $m!$ ways to shuffle m rows, this creates classes of shift offsets of size up to $m!$ that are equivalent.

As a canonical representation we can adopt the one where the offsets t_i are ordered by increasing value. Moreover, according to Condition sets 1 we should take all t_i different, thus we can adopt $\forall 0 < i < m : t_i > t_{i-1}$. This reduces the number of offset vectors $\mathbf{R}_E$ to investigate to less than $n/4\binom{n}{m}$.

Corollary 3. *All interesting classes of offset vectors $\mathbf{R}_E$ are represented by the lists with the following features*

- $r = 1$ *and* $s \in [2, n]$ *and even,*
- *For all* $i > 0$, $t_i > t_{i-1}$,
- $t_{n-1} \bmod 2 = \sum_{i<n-1} t_i \bmod 2$.

6 Linear Mask Propagation Properties

The propagation of linear masks through a linear mapping, important in the context of linear cryptanalysis, can be easily described with the matrix representation of the linear map. If we denote the matrix of L by M, we have $U = \mathsf{M}^{\mathrm{T}} V$ with V the mask at the output and U the mask at its input.

For a twin CPM we wish to express its transpose in terms of row additions and cyclic shifts. This allows expressing masks at the input as a function of masks at the output. This is more convenient with the polynomial representation.

6.1 Representing Masks as Polynomials

In linear cryptanalysis, a mask defines a linear function of state bits, i.e., a sum of the state bits in positions where the mask is 1. If a state A has the shape of a binary vector, the shape of the mask u is a binary vector of the same dimension A and the linear function corresponding to mask u is given by $u^{\mathrm{T}}A$. In our case the state is an array of m rows A_i and we can represent masks similarly, where the linear function corresponding to a mask U is $\sum_i {U_i}^{\mathrm{T}} A_i$.

We will express the m rows of a mask by polynomials $(U_0, U_1, \dots, U_{m-1})$ and its propagation through θ in polynomial form.

Definition 11 (transpose of a polynomial). *The transpose of a polynomial $P(X) \in \mathbb{F}_2[X]/(1+X^n)$ is given by the polynomial $P(X^{-1})$. Therefore, if $P = \sum_j p_j X^j$ then $P^{\mathrm{T}}(X) = \sum_i p_j X^{n-j \bmod n}$.*

With this we can express linear functions of a state using polynomials.

Lemma 12 (Expression for a linear function). *The linear function of a state $A \in (\mathbb{F}_2[X]/(1+X^n))^m$ defined by a mask $U \in (\mathbb{F}_2[X]/(1+X^n))^m$ is given by $U^{\mathrm{T}}A \bmod X$. In other words, it is the coefficient of X^0 in $U^{\mathrm{T}}A$.*

Proof. The coefficient of X^0 in $U_i^{\mathrm{T}}(X)A_i(X) \bmod 1+X^n = U_i(-X)A_i(X) \bmod 1+X^n$ is $\sum_j u_{i,j}a_{i,j}$. Taking the sum over all rows gives $\sum_{i,j} u_{i,j}a_{i,j}$. □

6.2 Mask Propagation through a Twin CPM

We will now show how input and output masks are related over a generalization of a circulant twin CPM in polynomial representation.

Lemma 13 (Mask propagation throught a circulant mapping). *Let L be determined by a square matrix of polynomials $\rho_{i,j}$ with $B_j \leftarrow A_j + \sum_i \rho_{i,j}A_i$. An input mask $V \in (\mathbb{F}_2[X]/(1+X^n))^m$ depends on an output mask $U \in (\mathbb{F}_2[X]/(1+X^n))^m$ as:*

$$V_i = U_i + \sum_j \rho_{j,i}^{\mathrm{T}} U_j \, .$$

Proof. We can compute an input mask V from the output mask U as follows:

$$\begin{aligned}
\sum_i U_i^{\mathrm{T}} B_i = \quad & \sum_i U_i^{\mathrm{T}} \left(A_i + \sum_j \rho_{i,j} A_j \right) = \sum_i U_i^{\mathrm{T}} A_i + \sum_j \sum_i U_i^{\mathrm{T}} \rho_{i,j} A_j \\
= \quad & \sum_i U_i^{\mathrm{T}} A_i + \sum_i \sum_j U_j^{\mathrm{T}} \rho_{j,i} A_i = \sum_i \left(U_i^{\mathrm{T}} + \sum_j U_j^{\mathrm{T}} \rho_{j,i} \right) A_i \\
= \quad & \sum_i \left(U_i + \sum_j \rho_{j,i}^{\mathrm{T}} U_j \right)^{\mathrm{T}} A_i \, .
\end{aligned}$$

hence, we have $\sum_i U_i^{\mathrm{T}} B_i = \sum_i V_i^{\mathrm{T}} A_i$ with $V_i = U_i + \sum_j \rho_{j,i}^{\mathrm{T}} U_j$. □

Therefore, the polynomial representation of the transpose of θ is given by:

$$A_i \leftarrow A_i + \sum_{j=0}^{4} X^{-u} \left(1 + X^{-r} + X^{-t_i} + X^{-s-t_i}\right) A_j \, .$$

This translates readily to:

$$\begin{aligned} A_i \leftarrow & A_i + ((E + (F \ggg t_i)) \ggg u) \text{ for } 0 \leq i < m \text{ with} \\ & E \leftarrow P + (P \ggg r) \, , \;\; F \leftarrow P + (P \ggg s) \text{ and } P \leftarrow \sum_{i=0}^{4} A_i \, . \end{aligned} \tag{9}$$

We see two θ-effects, E and F with different folding offsets: r and s. They are both added to the rows, where the second one is shifted over different offsets t_i.

6.3 Low-Weight 3-Round Linear Trails

The P-kernel of θ^{T} consists of all states with an even number of active bits in each column. It has elements with only two active bits, namely P-orbitals, in total $n\binom{m}{2}$ of them. It follows that the bit branch number of θ^{T} is at most 4. It is also at least 4 because for any state A with a single active bit $\theta^{\mathrm{T}}(A)$ and $\left(\theta^{\mathrm{T}}\right)^{-1}(A)$ have more than 3 bits.

If all shift offsets in $\mathbf{R}_w$ are different and similarly all shift offsets in $\mathbf{R}_e$, the bit branch number of θ^{T} translates to a column branch number of 4 for λ^{T}. As an active column has at least weight (and minimum reverse weight) 2, it follows that there are 2-round trails with weight 8.

For three rounds we consider a particular type of linear trails. We know a single-bit mask at the output of χ, restricted to a single column, is correlated to the same single-bit mask at its input with correlation 1/2. Therefore, with respect to masks with at most a single active bit per column, χ can act as the identity. The type of 3-round linear trails we consider are those where this is the case for the middle χ.

An orbital at θ of the second round is transformed by ${\rho_{\mathrm{west}}}^{-1}$, goes through χ unchanged and is then transformed by ${\rho_{\mathrm{east}}}^{-1}$ to end up at the output of θ of the first round. If these bits are in the same column, they form again an orbital and are in the kernel, implying that we see it unchanged at the input of θ. This would mean a 3-round trail that has in each χ-step only two active bits, thus with weight only 12.

We can avoid these by having conditions on the shift offsets formed by the combination of ρ_{west} and ρ_{east}. We will denote them by $c_i = e_i + w_i$.

From the example above, we should ensure $\forall i, j : c_i \neq c_j$. These are the rules of Condition sets 1, applied to c_i. Similarly, violation of the rules in Condition sets 3 applied to c_i would lead to trails with 4-bit states in the kernel of θ^{T} in two consecutive rounds, and thus to 3-round linear trails with weight only 24.

We cannot avoid the existence of linear trails with 6-bit states in the kernel of θ^{T} in two consecutive rounds, and these result in 3-round linear trails with weight at most 36. In particular, they are like the vortices of Definition 9, but with c_i taking the place of t_i. In total there are $\binom{m}{3}$ canonical vortices and there may be additional trails with 6-bit states in the kernel of θ^{T} in two consecutive rounds. If the 6 active bits of the masks at input/output of the χ steps in these trails are not in 6 different columns, the weight goes under 36.

6.4 Transpose of Twin CPMs

Twin CPMs have stronger diffusion for the propagation of differences than for the propagation of masks. In some use cases we may wish to have the inverse. For those cases we can use the transpose of a twin CPM as specified in (9).

7 Application: the Design of Gaston

In this section we will illustrate the power of circulant twin CPMs by building a concrete iterated permutation using it as the mixing layer in its round function. Our permutation operates on a two-dimensional state with the same dimensions as Ascon-p and we decided to call it Gaston.

7.1 The Structure of the Round Function and Shape of the State

Gaston operates on 320-bit states with $m = 5$ rows and $n = 64$ columns. Its round function consists of a linear layer followed by a non-linear layer. As in the iterated permutation Xoodoo [12], the linear layer consists of the mixlayer θ, preceded and followed by row shift steps ρ_{west} and ρ_{east} respectively and a round constant addition ι. The non-linear layer consists of the application of the χ-mapping that operates in parallel on the 5-bit columns. The main differences with the Xoodoo round function are that θ is a circulant twin CPM rather than a circulant CPM and that the state has 5-bit columns rather than 3-bit columns.

After fixing the round function structure, the design effort mainly consists of determining the shift offsets of the mixlayer and the row shift steps. We discuss our search strategy to find shift offsets for the mixlayer as well as for the two row shuffles in Sec. 7.2, the procedure we followed for $\mathbf{R}_\theta$ in Sec. 7.3 and for $\mathbf{R}_\rho$ in Sec. 7.4 and Sec. 7.5.

7.2 General Search Strategy

As discussed in Sec. 3, the shift offsets of the linear layer λ are gathered in a shift offset vector $\mathbf{R}_\lambda = (\mathbf{R}_\theta, \mathbf{R}_\rho)$. We aim to find a shift offset vector $\mathbf{R}_\lambda$ that minimizes the maximum DP value of differentials over a fixed round version of Gaston, say with 4 rounds. Determining the maximum DP value of 4-round differentials is actually a huge task, even for a single value of $\mathbf{R}_\lambda$. Moreover, the number of $\mathbf{R}_\lambda$ values is astronomical: It has $3m + 1$ entries each in the range $[0, 63]$, thus for Gaston there are $64^{16} = 2^{96}$ possible values. Taking into account the equivalence classes discussed in Sec. 5 this reduces to $16\binom{64}{5}64^9 \approx 2^{81}$, but it remains astronomical.

With minimization of the maximum DP of differentials over 4-round Gaston being infeasible, we turn to local optimization. In particular, we will choose a value for $\mathbf{R}_\lambda$ that gives rise to a linear layer λ with a column branch number of 12. There are many such values of $\mathbf{R}_\lambda$ and as secondary criterion we look at the left tail in the column branch histogram of the corresponding mappings λ.

For λ to have a column branch number of 12, θ must have a bit branch number of 12. Moreover, the height of the tail of the column branch histogram of λ depends strongly on that of the bit branch histogram of θ. Both the bit branch number and the height of the tail of the bit branch histogram of θ are fully determined by $\mathbf{R}_\theta$. Therefore, we opt for a two-phase approach: we first determine $\mathbf{R}_\theta$ and then only $\mathbf{R}_\rho$, where we choose $\mathbf{R}_\theta$ based on quantitative bit-level diffusion properties of θ and $\mathbf{R}_\rho$ is chosen to translate this bit-level diffusion to the column level for λ.

Our two phases are each divided into a number of steps. In particular, for $\mathbf{R}_\rho$ we first find a suitable candidate for $\mathbf{R}_w$ before making a choice on $\mathbf{R}_e$.

7.3 Selecting the Offsets of $\mathbf{R}_\theta$

To find the offsets for $\mathbf{R}_\theta$, the steps are the following:

1. $\mathbf{R}_E$-condition-filtering: we eliminate all $\mathbf{R}_E$ candidates leading to avoidable in-kernel states with less than 6 active bits. This phase reduces the $16\binom{64}{5} \approx 2^{27}$ $\mathbf{R}_E$ candidates that move on to the next phase to $14\,282\,988 \approx 2^{23}$ candidates.
2. $\mathbf{R}_E$-kernel-filtering: we keep the $\mathbf{R}_E$ candidates that have no in-kernel states with less than 6 active bits and count the number of in-kernel states with 6 active bits. There are 828 candidates for $\mathbf{R}_E$ with 11 6-bit states.
3. $\mathbf{R}_E$-candidate-pool: for each $\mathbf{R}_E$ candidate we compute the θ-effect E of all states with 3 active bits and keep those that minimize the number of 3-bit states that result in E having 2 active bits. This minimum turns out to be 1 and there are 3 such candidates for $\mathbf{R}_E$.
4. u-selection: we exhaustively test all u candidates and select the value that moves all active bits to different columns for states outside the kernel with 5 active bits or less. After this phase $\mathbf{R}_\theta$ is fully determined.

$\mathbf{R}_E$*-Condition-Filtering.* Several cases of in-kernel and out-of-kernel states with bit branch number less than 12 are avoidable by discarding shift offsets defined in Sec. 4. More specifically, we discard candidates leading to single-orbital states (Condition sets 1) as well as 2-bit and 4-bit states in the kernel (Condition sets 3). Offsets that result in in-kernel states with an odd number of active bits, in-kernel row twins states with less that 8 active bits (Condition sets 2) and 2-bit states that have a θ-effect E with 2 active bits (Condition sets 4) are also eliminated.

$\mathbf{R}_E$-*Kernel-Filtering.* For each $\mathbf{R}_E$ candidate we generate all states up to 6 active bits and discard any candidate that leads to in-kernel states with 4 active bits as well as 2-bit states with 2 bits in E that were not caught during the $\mathbf{R}_E$-condition-filtering phase. We keep those that result in the minimum number of in-kernel states with bit branch number of 12.

$\mathbf{R}_E$-*Candidate-Pool.* For each $\mathbf{R}_E$ candidate we look at the θ-effect E of all out-of-kernel states with less than 6 active input bits. For a θ-effect E with 2 active bits the bitwise branch number is at least 10. If E has 4 active bits then states have a bitwise branch number of at least 20. During this phase we discard $\mathbf{R}_E$ candidates that do not minimize the number of 3-bit states with 2 bits in E. We report in Table 2 the histograms for the chosen candidate (right table) and we also give an example of a discarded $\mathbf{R}_E$ candidate (left table).

Table 2: 2-dimensional histograms giving the number of states according to the number of active bits in the θ-effect (vertical axis) and the number of active bits at the input (horizontal axis). We have $\mathbf{R}_E = (1, 20, 4, 6, 21, 33, 61)$ at the left and $\mathbf{R}_E = (1, 18, 25, 32, 52, 60, 63)$ at the right.

	3	4	5	6
0				11
2	6	28	168	868

	3	4	5	6
0				11
2	1	24	158	686

u-Selection. For the 3 $\mathbf{R}_E$ candidates, we generate the histogram listing the number of states according to the bit weight at input and output for all u values. We generate all states with at most 6 active input bits and at most 2 active bits in the θ-effect E. In the generated histograms, we see that u impacts an effect that we call *affected-loss* that we now explain. An active bit in an unaffected column contributes 2 to the branch number and an active bit in an affected column does not contribute to it. We choose the u candidate that minimizes that affected-loss. As the offset u shifts the θ-effect E before it is added to the rows, we only need to look at out-of-kernel states. We chose to look only at those with few active bits that have two active bits in E as that is where the risk of getting a branch number below 12 is. We select the u value where active bits do not overlap with affected columns for all states up to 5 active bits. We report in Table 4 the histogram associated to the chosen candidate and we also give an example of a u candidate that can be discarded in Table 3. In particular, there are 24 4-bit states with 2 bits in E and a bit overlapping with a affected column costs 2 bits. If u is fixed to 29 then out of these 24 states there are 3 states that have an image with 10 active bits. It means that there are 2 bits at the input that are overlapping with an affected column. Moreover, 2 states have an image with 12 active bits which means that 1 bit is overlapping with an affected column. For $u = 23$ all 24 4-bit states will have an image with 14 active bits. We report in Table 4 the bit branch histogram associated to the chosen candidate.

Table 3: 2-dimensional bit weight histograms for $\mathbf{R}_E = (1, 18, 25, 32, 52, 60, 63)$ with offset $u = 29$.

	6	7	8	9	10	11	12	13	14	15
3								1		
4					3		2		19	
5						6		2		150
6	11		2		3		40		73	

Table 4: 2-dimensional bit weight histogram for $\mathbf{R}_E = (1, 18, 25, 32, 52, 60, 63)$ with offset $u = 23$.

	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1																5		
2																	29	
3								1								5		149
4									24				1		14		67	
5										158		3		17		289		506
6	11						18		15		653		42		132		3012	
7				1				200		143		4447		578		1930		29567
8			346		18		25		1421		1432		25849		7119		24174	
9						94		303		12900		16428		207062		81499		284023
10					10952		1247		3139		94506		141168		1313837		894509	
11				17		58		10545		39889		842033		1492497		10832247		9631128

7.4 Selecting the Offsets of $\mathbf{R}_w$

The mixlayer θ being set, we can generate low-branch number in-kernel and out-of-kernel states. The role of the row shuffle $\mathbf{R}_w$ is to move bits that are in the same column in a state at the output of θ to different columns at the output of λ. To determine $\mathbf{R}_w$ we proceed with the following steps:

1. $\mathbf{R}_w$-kernel-filtering: for each $\mathbf{R}_w$ candidate we apply ρ_{west} to the 11 in-kernel states with 6 active bits, see Table 4. We keep those that shift all 6 bits to different columns at the output of λ. This phase reduces the $63 * 62 * 61 * 60 \approx 2^{24}$ candidates for $\mathbf{R}_w$ to $3\,943\,564 \approx 2^{22}$.
2. $\mathbf{R}_w$-condition-filtering: we reduce the set of $\mathbf{R}_w$ candidates by keeping those that for the 10 bits in any 2 affected columns are moved to at least 9 different columns. This reduces the set of 2^{22} candidates for $\mathbf{R}_w$ candidates to $1\,362\,650 \approx 2^{20}$ candidates.
3. $\mathbf{R}_w$-kernel-filtering: we apply each $\mathbf{R}_w$ candidate to all in-kernel states with 8 active bits, see Table 4. We keep those that shift at most 2 active bits to the same column. We are left with 9212 candidates.
4. $\mathbf{R}_w$-candidate-pool: we apply each $\mathbf{R}_w$ candidate to out-of-kernel states with at most bit branch number 18. We keep those where at most one column will have two active bits at the output of λ. There are 24 candidates.

$\mathbf{R}_w$*-Kernel-Filtering.* The desired effect of ρ_{west} is the following. For low-branch-number states, the row shuffle ρ_{west} should move active bits in the state after θ

to different columns at the output of λ. When bits cluster in columns it results in a state having a column branch number under λ smaller than the bit branch number under θ. The effect of active bits at the output of θ ending up in the same column after λ is called *bit huddling* [9]. In this phase, for all $\mathbf{R}_w$ values we check whether for the 11 states with bit branch number 12 there is bit huddling at the output of λ and keep only those where there is none.

$\mathbf{R}_w$*-Condition-Filtering.* For states outside the kernel, the state at the output of θ will have a number of affected columns. For low-branch-number states, typically an affected column will have most of its bits active, most often all of them. Now, if we have a state consisting of two affected columns at the output of θ, the positions of these affected columns can be chosen such that there are at least 2 bits in the same column after ρ_{west}: the 10 bits huddle to 9 columns. By a judicious choice of the shift offsets in $\mathbf{R}_w$ this number can be limited to not be lower than 9 for any two affected column position. The conditions on the shift offsets to limit this bit huddling coincide with Condition sets 3.

$\mathbf{R}_w$*-Kernel-Filtering.* We keep any $\mathbf{R}_w$ that moves the 8 active bits before ρ_{west} to at least 7 different columns after ρ_{west}.

$\mathbf{R}_w$*-Candidate-Pool.* We require that a $\mathbf{R}_w$ candidate moves active bits in the state before ρ_{west} to different columns after ρ_{west}. We minimize the bit huddling effect to one column having at most 2 active bits.

7.5 Selecting the Offsets of $\mathbf{R}_e$

Following the procedure to determine a suitable $\mathbf{R}_w$, for each of the 24 $\mathbf{R}_w$ candidates we go through the following steps to make a choice on $\mathbf{R}_e$:

1. $\mathbf{R}_e$-kernel-filtering: the $\mathbf{R}_e$ candidates are selected from the set of $\mathbf{R}_w$ candidates passing the $\mathbf{R}_w$-kernel-filtering phase.
2. $\mathbf{R}_e$-condition-filtering: for each $\mathbf{R}_e$ candidate we take into account $\mathbf{R}_w$. We eliminate any $\mathbf{R}_e$ that results in vortices with less than 4 bits remaining in the kernel of θ^{T} for two successive rounds.
3. $\mathbf{R}_e$-vortices-filtering: we apply each $(\mathbf{R}_w, \mathbf{R}_e)$ candidate to the 10 canonical vortices obtained by the combination of ρ_{west} and ρ_{east}. Candidates that result in vortices with less than 6 bits in the kernel of θ^{T} for two successive rounds when χ acts as the identity are discarded.
4. $\mathbf{R}_\rho$-tie-break: we check the minimum squared correlation C^2 of 3-round linear trails in the kernel of θ^{T} for the $(\mathbf{R}_w, \mathbf{R}_e)$ candidates.

$\mathbf{R}_e$*-Kernel-Filtering.* The conditions for a $\mathbf{R}_e$ candidate in this step are the same as the conditions for a $\mathbf{R}_w$ candidate in the $\mathbf{R}_w$-kernel-filtering step, but applied to the additive inverses of the ρ_{east} shift offsets that we denote as $-\mathbf{R}_e$. A $\mathbf{R}_e$ candidate passes the step if it moves all bits of the 11 6-bit states at the input of θ to different columns after ${\rho_{\text{east}}}^{-1}$. As all these 6-bit states are in-kernel, the bits before and after θ are the same. Thus, the list of candidates for $-\mathbf{R}_e$ is the same as the list of candidates generated during the $\mathbf{R}_w$-kernel-filtering step.

$\mathbf{R}_e$-*Condition-Filtering.* As described in Sec. 6.3, we eliminate all candidates leading up to low-weight 3-round linear trails with 4-bit states in the kernel of θ^{T} in two consecutive rounds when χ acts as the identity. The conditions to discard those offsets coincide with Condition sets 3 applied to $\rho_{\text{west}} + \rho_{\text{east}}$ and ensure that we don't obtain a 3-round in-kernel linear trail with weight 24 when χ acts as the identity.

$\mathbf{R}_e$-*Vortices-Filtering.* As stated in Sec. 6.3, we can generate all 10 canonical vortices obtained from the combination of $\mathbf{R}_w$ and $\mathbf{R}_e$. For each candidate, we apply $\rho_{\text{east}}{}^{-1}$, ρ_{west} as well as $\rho_{\text{west}} \circ \rho_{\text{east}} \circ \rho_{\text{west}}$ and require that all 6 bits in the 10 vortices are moved to different columns after each operation. Doing so ensures that we avoid vortices with 6 active bits in the kernel of θ^{T} in two consecutive rounds which lead to 3-round linear trails with weight less than 36.

$\mathbf{R}_\rho$-*Tie-Break.* For a $(\mathbf{R}_w, \mathbf{R}_e)$ candidate, we model the linear propagation behavior of the permutation with SMT and we check the minimum weight of 3-round linear trails in the kernel of θ^{T}. We select a candidate that maximizes the minimum squared correlation C^2 of trails over 3 rounds in the kernel of θ^{T}. The parameters for the linear layer are given in Table 6 in Sec. 8.

7.6 Trail Search and Bounds

Several of the steps in the selection process for the linear layer require the generation of states up to some branch number. To generate these states, we use the tree traversal technique and we make use of the two-level tree search as well as canonicity and a *score* function, both defined in [19]. We further use Satisfiability Modulo Theories (SMT) and Mixed-Integer Linear Programming (MILP) in a hybrid manner [30] to obtain bounds on the differential and linear trails (see Appendix A). We then discuss our results in Table 5.

Two-Level Tree Search. The two-level tree traversal strategy introduced in [19] allows efficiently generating all states with branch number below a given target. The strategy arranges the states in a tree with nodes identified by a row list, that specifies the active rows of the state. The parent of a state is the row list with the last row removed and the root of the tree is the empty list. The active rows are lists of active bits, indicated by their x-coordinates. When traversing the tree, going to the first child of a node corresponds to adding the smallest possible active row after its last active row in the list. Going to the sibling of a node corresponds to iterating the last active row to the next value. Lastly, going to the parent of a node is done by removing the last active row from the row list.

Canonicity. The linear layer λ and the non-linear layer χ are both shift invariant with respect to translation along the horizontal axis. This symmetry property partitions the state space into equivalence classes and we only need to visit one node per class. The representative of such a class is called a *canonical* state and corresponds to the *smallest* state of its class, for some order relation. We adopt an order relation where a non-canonical state has no canonical descendants. Hence, we can safely prune complete subtrees of non-canonical state in the tree.

Score. We denote by *score* the function that gives a lower bound on the branch number of a node and all its descendants. It allows us to safely prune complete subtrees from nodes with score higher than the target branch number. Moreover, we are only interested in low-branch number states and thus the states generation can be limited by the number of active bits in the θ-effect E. Since the addition of a bit in a state at the input of θ always adds 4 active bits at the output of θ with the number of affected columns, it is enough to generate all states with a score below 4 times the limit weight at the input.

SMT and MILP Models for Analysis of Trails in Gaston. Since the automated methods for searching differential and linear trails are well known, we omit their description in this section due to space limitation. However, we discuss the modeling for both automated tools in Appendix A.

Results. We use the parameters from Table 6 to compute the differential and linear properties of Gaston and compare them with Ascon in Table 5.

Table 5: Differential and linear bounds of Gaston where kernel of E and P are taken at the first round. The exact numbers in the table represents the minimum values.

Differential			
	Gaston		Ascon
	$\mathrm{Ker}(E) = 0$	$\mathrm{Ker}(E) \neq 0$	
Column-wise branch number	12	$13 - 16$	4
Weight of 2-round trail (B.1)	24	$26 - 34$	8
Weight of 3-round trail (B.1)	≤ 106	-	40
Linear			
	$\mathrm{Ker}(P) = 0$	$\mathrm{Ker}(P) \neq 0$	
Column-wise branch number	4	21	4
Weight of 2-round trail	8	42	8
Weight of 3-round trail (B.2)	34	≥ 44	28

From Table 5, we observe that the column-wise branch number (differential) of Gaston's mixing layer is three times larger than that of Ascon, while the linear column-wise branch number of both mixing layers are the same. The best 3-round differential trail we find for Gaston has weight 106, much higher than Ascon's 3-round differential trail with weight 40. Moreover, the optimal weight of Gaston's 3-round linear trail is 34 while it is 28 for Ascon.

Algorithm 1 Definition of Gaston

```
Parameters: Number $q$ of rounds
for Round index $i$ from $1-q$ up to $0$ do $A = \mathrm{R}_i(A)$

The round function R:
for index $j$ from 0 to 4 do  $A_j \leftarrow (A_j \lll e_j)$                                   ▷ $\rho_{\text{east}}$
$E \leftarrow (\sum A_j + (\sum A_j \lll r)) + (\sum (A_j \lll t_j) + (\sum (A_j \lll t_j) \lll s))$
for index $j$ from 0 to 4 do $A_j \leftarrow A_j + (E \lll u)$                                  ▷ $\theta$
for index $j$ from 0 to 4 do $A_j \leftarrow (A_j \lll w_j)$                                    ▷ $\rho_{\text{west}}$
$A_0 \leftarrow A_0 + \mathrm{C}_i$                                                              ▷ $\iota$
for index $j$ from 0 to 4 do $A_j \leftarrow A_j + \overline{A_{j+1}} \cdot A_{j+2}$            ▷ $\chi$
```

8 Specifications of Gaston and Use Case

In this section we provide a non-ambiguous specification of the Gaston family of permutations and present a concrete use case.

As in Ascon-p and Xoodoo, the number of rounds in Gaston is not fixed but depends on the use case. The round function is the same for all rounds, except for the round constants. As in Xoodoo and Ascon-p, the round constants are numbered from the last round backwards, therefore, the last round always has the same round constant, irrespective of the number of rounds and likewise the penultimate round, etc. The round constants used in Gaston correspond to those used in Ascon-p . Algorithm 1 describes Gaston and uses the shift offset parameters alongside the round constants that are specified in Table 6. We denote by $A_j \cdot A_{j'}$ the bitwise product (AND) of rows A_j and $A_{j'}$ while $\overline{A_j}$ denotes the bitwise complement of row A_j. A reference implementation of Gaston is available at `https://gitlab.science.ru.nl/selhirch/circulanttwincpm`.

Table 6: List of parameters for the linear diffusion layer of Gaston at the left and the round constants C_i with $-11 \leq i \leq 0$, in hexadecimal notation at the right.

Index	$\mathbf{R}_\theta = (r\ \ s\ \ t_j\ \ u)$	$\mathbf{R}_w$	$\mathbf{R}_e$
0	1 18 25 23	0	0
1	32	56	60
2	52	31	22
3	60	46	27
4	63	43	4

i	C_i	i	C_i	i	C_i	i	C_i
-11	`0xf0`	-8	`0xc3`	-5	`0x96`	-2	`0x69`
-10	`0xe1`	-7	`0xb4`	-4	`0x87`	-1	`0x5a`
-9	`0xd2`	-6	`0xa5`	-3	`0x78`	0	`0x4b`

A permutation by itself does not constitute as a cryptographic primitive, it must be used in a construction to be cryptographically useful. Constructions include Even-Mansour [20] to build a block cipher, Farfalle [4] to build a deck function (a variable-input-length variable-output-length pseudorandom function), sponge to build a hash function or an extendable output function (XOF) [5] and (monkey)duplex [7] as a basis for authenticated encryption schemes like Ascon. There is a actually quite a list of permutation-based constructions and modes and this list is still growing.

We believe Gaston, and in some cases a similar permutation using a transpose twin CPM, are competitive in most of these constructions but choosing the required number of rounds would require some more comprehensive cryptanalysis. In Farfalle however the security depends mostly on differential and correlation properties and the number of rounds is easier to choose based on existing analysis. Therefore we will concentrate on that use case in this subsection.

Farfalle makes use of 4 permutations that in the case of Xoofff all are 6-round Xoodoo. When instantiated with Gaston, we propose the following:

p_c in the compression layer. In [21] it has been shown that the strength against inner collisions in Farfalle is given by $\max_a \sum_b \mathrm{DP}^2(a,b) \leq \max_{a,b} \mathrm{DP}(a,b)$ under reasonable assumptions. To achieve 128 bits of security with respect to this, it is sufficient that $\max_{a,b} \mathrm{DP}(a,b) \leq 2^{-128}$. With the current bounds we are confident that 4 rounds are sufficient.

p_e in the expansion layer. This permutation needs to protect against state/key-recovery attacks using output only and to avoid biases in the output. For the latter the input-output correlations are important and therefore a Gaston variant with a transpose twin CPM would be the logical choice. Assuming the squared correlation of the best linear trails for this variant would be similar to the DP of differential trails for Gaston, 4 rounds would be sufficient to avoid detectable biases in the output. Moreover, based on the cryptanalysis of [11], the high diffusion of the twin CPM in Gaston in comparison to that of the CPM of Xoodoo would make 4 rounds sufficient to offer a comfortable security margin with respect to state/key-recovery attacks.

p_d between the compression and expansion layer. Its role is to protect against input/output attacks. Here the algebraic degree of the concatenation of p_d and p_e is relevant rather than difference propagation and correlation. Here a Gaston variant with a classical CPM would be the logical choice, thanks to its low gate cost of 2 XORs per state bit. We believe again 4 rounds would be sufficient, resulting in an algebraic degree close to 256.

p_b for deriving the initial mask from the key K. The purpose of this permutation is to diffuse the key bits to the mask in case of a biased key. Here we propose to take Gaston with 4 rounds where the reduction in rounds with respect to Xoodoo is motivated by the higher diffusion of the twin CPM as compared to the CPM.

Thus, all 4 permutations have 4 rounds instead of 6 rounds. The rounds of Gaston are slightly more expensive per bit than those of Xoodoo but this is still compensated by the decrease in number of rounds. Additionally, the Gaston-based Farfalle instance operates on smaller blocks than Xoofff, 320 bits instead of 384. This is an advantage on constrained platforms. We believe this provides a promising perspective for the future adoption of Gaston and its variants.

9 Conclusions

Circulant twin CPMs are a generalization of CPMs used as mixing layer in Keccak-f and Xoodoo. Together with their transpose, they form a new category of mixing layer that has very high local diffusion in the form of a differential branch number (linear branch number for the transpose) not seen before for such a low cost. The optimization for differential or linear branch number is a new feature that allows to get a much higher specific branch number for the same amount of computation than MDS matrices.

We use a twin CPM in a proof-of-concept lightweight permutation Gaston that has a round function that has the same width, shape and gate cost as that of Ascon-p. Due to its high branch number, the best 2-round differential trails in Gaston have weight 24 (vs. 8 in Ascon-p) and the best linear trails of Gaston have weight 34 (vs. 28 in Ascon-p). No 3-round differential trails were found with weight less than 106 while in Ascon-p there are 3-round differential trails with weight 40.

Open Problems. For Gaston it would be interesting to have tighter bounds for the weight of differential trails over more than 2 rounds and linear trails over more than 3 rounds. This will require the introduction of new techniques. Moreover, interesting aspects to investigate are clustering of differential trails in differentials, clustering of linear trails in linear approximations and the validation of the approximation of the DP of differential trails by $\mathrm{DP}(Q) \approx 2^{-\mathrm{w_r}(Q)}$. Another direction for future research is to explore permutations operating on states with other dimensions such as those of Xoodoo.

Finally, it would be interesting to more closely investigate the use of Gaston-like permutations within constructions like Farfalle, (monkey)-duplex such as Ascon or Xoodyak and others and do a comprehensive analysis of the permutation in that context.

Acknowledgements

Solane El Hirch is supported by the Cryptography Research Center of the Technology Innovation Institute (TII), Abu Dhabi (UAE), under the TII-Radboud project with title *Evaluation and Implementation of Lightweight Cryptographic Primitives and Protocols.* Joan Daemen is supported by the European Research Council under the ERC advanced grant agreement under grant ERC-2017-ADG Nr. 788980 ESCADA. Rusydi H. Makarim did this work while he was at the Cryptography Research Center of the Technology Innovation Institute (TII), Abu Dhabi (UAE). We would also like to thank the reviewers of CRYPTO 2023 and our shepherd Bart Preneel for their insightful suggestions which helped us in significantly improving the quality of paper.

References

1. Abdelkhalek, A., Sasaki, Y., Todo, Y., Tolba, M., Youssef, A.M.: MILP modeling for (large) s-boxes to optimize probability of differential characteristics. IACR Trans. Symmetric Cryptol. **2017**(4), 99–129 (2017), https://doi.org/10.13154/tosc.v2017.i4.99-129
2. Barreto, P., Rijmen, V.: The WHIRLPOOL hashing function. Submitted to NESSIE, Sept 2000, revised May 2003. Available online at https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=664b5286124b28abf2d30a07ba6f9e020f4138fe
3. Beierle, C., Kranz, T., Leander, G.: Lightweight multiplication in GF(2^n) with applications to MDS matrices. In: Robshaw, M., Katz, J. (eds.) Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9814, pp. 625–653. Springer (2016), https://doi.org/10.1007/978-3-662-53018-4_23
4. Bertoni, G., Daemen, J., Hoffert, S., Peeters, M., Van Assche, G., Van Keer, R.: Farfalle: parallel permutation-based cryptography. IACR Trans. Symmetric Cryptol. **2017**(4), 1–38 (2017), https://tosc.iacr.org/index.php/ToSC/article/view/801
5. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Sponge functions. Ecrypt Hash Workshop 2007 (May 2007)
6. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: The Keccak reference (January 2011), https://keccak.team/papers.html
7. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Permutation-based encryption, authentication and authenticated encryption. In: Directions in Authenticated Ciphers (2012)
8. Biham, E., Shamir, A.: Differential cryptanalysis of DES-like cryptosystems. In: Menezes, A., Vanstone, S.A. (eds.) Advances in Cryptology - CRYPTO '90, 10th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1990, Proceedings. Lecture Notes in Computer Science, vol. 537, pp. 2–21. Springer (1990), https://doi.org/10.1007/3-540-38424-3_1
9. Bordes, N., Daemen, J., Kuijsters, D., Van Assche, G.: Thinking outside the superbox. In: Malkin, T., Peikert, C. (eds.) Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part III. Lecture Notes in Computer Science, vol. 12827, pp. 337–367. Springer (2021), https://doi.org/10.1007/978-3-030-84252-9_12
10. Canteaut, A., Duval, S., Leurent, G., Naya-Plasencia, M., Perrin, L., Pornin, T., Schrottenloher, A.: Saturnin: a suite of lightweight symmetric algorithms for post-quantum security. IACR Trans. Symmetric Cryptol. **2020**(S1), 160–207 (2020), https://doi.org/10.13154/tosc.v2020.iS1.160-207
11. Cui, T., Grassi, L.: Algebraic key-recovery attacks on reduced-round Xoofff. In: Dunkelman, O., Jr., M.J.J., O'Flynn, C. (eds.) Selected Areas in Cryptography - SAC 2020 - 27th International Conference, Halifax, NS, Canada (Virtual Event), October 21-23, 2020, Revised Selected Papers. Lecture Notes in Computer Science, vol. 12804, pp. 171–197. Springer (2020), https://doi.org/10.1007/978-3-030-81652-0_7
12. Daemen, J., Hoffert, S., Van Assche, G., Van Keer, R.: The design of Xoodoo and Xoofff. IACR Trans. Symmetric Cryptol. **2018**(4), 1–38 (2018), https://tosc.iacr.org/index.php/ToSC/article/view/7359

13. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Information Security and Cryptography, Springer (2002), https://doi.org/10.1007/978-3-662-04722-4
14. Daemen, J.: Cipher and hash function design, strategies based on linear and differential cryptanalysis, PhD Thesis. K.U.Leuven (1995), http://jda.noekeon.org/
15. Daemen, J., Mella, S., Van Assche, G.: Tighter trail bounds for Xoodoo. IACR Cryptol. ePrint Arch. p. 1088 (2022), https://eprint.iacr.org/2022/1088
16. Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M.: Ascon v1.2. Submission to NIST Lightweight Cryptography Standardization Process (round 2) (March 2019), https://ascon.iaik.tugraz.at/
17. Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M.: Ascon v1.2: Lightweight authenticated encryption and hashing. J. Cryptol. **34**(3), 33 (2021), https://doi.org/10.1007/s00145-021-09398-9
18. Duval, S., Leurent, G.: MDS matrices with lightweight circuits. IACR Trans. Symmetric Cryptol. **2018**(2), 48–78 (2018), https://doi.org/10.13154/tosc.v2018.i2.48-78
19. El Hirch, S., Mella, S., Mehrdad, A., Daemen, J.: Improved differential and linear trail bounds for ASCON. IACR Trans. Symmetric Cryptol. **2022**(4), 145–178 (2022), https://doi.org/10.46586/tosc.v2022.i4.145-178
20. Even, S., Mansour, Y.: A construction of a cipher from a single pseudorandom permutation. In: Imai, H., Rivest, R.L., Matsumoto, T. (eds.) Advances in Cryptology - ASIACRYPT '91, International Conference on the Theory and Applications of Cryptology, Fujiyoshida, Japan, November 11-14, 1991, Proceedings. Lecture Notes in Computer Science, vol. 739, pp. 210–224. Springer (1991), https://doi.org/10.1007/3-540-57332-1_17
21. Fuchs, J., Rotella, Y., Daemen, J.: On the security of keyed hashing based on an unkeyed block function. IACR Cryptol. ePrint Arch. p. 1172 (2022), https://eprint.iacr.org/2022/1172
22. Guo, J., Peyrin, T., Poschmann, A.: The PHOTON family of lightweight hash functions. In: Rogaway, P. (ed.) Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings. Lecture Notes in Computer Science, vol. 6841, pp. 222–239. Springer (2011), https://doi.org/10.1007/978-3-642-22792-9_13
23. Guo, J., Peyrin, T., Poschmann, A., Robshaw, M.J.B.: The LED block cipher. In: Preneel, B., Takagi, T. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings. Lecture Notes in Computer Science, vol. 6917, pp. 326–341. Springer (2011), https://doi.org/10.1007/978-3-642-23951-9_22
24. Kranz, T., Leander, G., Stoffelen, K., Wiemer, F.: Shorter linear straight-line programs for MDS matrices. IACR Trans. Symmetric Cryptol. **2017**(4), 188–211 (2017), https://doi.org/10.13154/tosc.v2017.i4.188-211
25. Li, C., Wang, Q.: Design of lightweight linear diffusion layers from near-mds matrices. IACR Trans. Symmetric Cryptol. **2017**(1), 129–155 (2017), https://doi.org/10.13154/tosc.v2017.i1.129-155
26. Li, S., Sun, S., Shi, D., Li, C., Hu, L.: Lightweight iterative MDS matrices: How small can we go? IACR Trans. Symmetric Cryptol. **2019**(4), 147–170 (2019), https://doi.org/10.13154/tosc.v2019.i4.147-170
27. Li, Y., Wang, M.: On the construction of lightweight circulant involutory MDS matrices. In: Peyrin, T. (ed.) Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Pa-

pers. Lecture Notes in Computer Science, vol. 9783, pp. 121–139. Springer (2016), https://doi.org/10.1007/978-3-662-52993-5_7
28. Liu, M., Sim, S.M.: Lightweight MDS generalized circulant matrices. In: Peyrin, T. (ed.) Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers. Lecture Notes in Computer Science, vol. 9783, pp. 101–120. Springer (2016), https://doi.org/10.1007/978-3-662-52993-5_6
29. Liu, M., Sim, S.M.: Lightweight MDS generalized circulant matrices. In: Peyrin, T. (ed.) Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers. Lecture Notes in Computer Science, vol. 9783, pp. 101–120. Springer (2016), https://doi.org/10.1007/978-3-662-52993-5_6
30. Makarim, R.H., Rohit, R.: Towards tight differential bounds of Ascon: a hybrid usage of SMT and MILP. IACR Trans. Symmetric Cryptol. **2022**(3), 303–340 (2022), https://doi.org/10.46586/tosc.v2022.i3.303-340
31. Matsui, M., Yamagishi, A.: A new method for known plaintext attack of FEAL cipher. In: Rueppel, R.A. (ed.) Advances in Cryptology - EUROCRYPT '92, Workshop on the Theory and Application of of Cryptographic Techniques, Balatonfüred, Hungary, May 24-28, 1992, Proceedings. Lecture Notes in Computer Science, vol. 658, pp. 81–91. Springer (1992), https://doi.org/10.1007/3-540-47555-9_7
32. Mella, S., Daemen, J., Van Assche, G.: New techniques for trail bounds and application to differential trails in Keccak. IACR Trans. Symmetric Cryptol. **2017**(1), 329–357 (2017)
33. National Institute of Standards and Technology: Lightweight Cryptography (LWC) Standardization project (2019), https://csrc.nist.gov/projects/lightweight-cryptography
34. Rijmen, V., Daemen, J., Preneel, B., Bosselaers, A., De Win, E.: The cipher SHARK. In: Gollmann, D. (ed.) Fast Software Encryption, Third International Workshop, Cambridge, UK, February 21-23, 1996, Proceedings. Lecture Notes in Computer Science, vol. 1039, pp. 99–111. Springer (1996), https://doi.org/10.1007/3-540-60865-6_47
35. Sim, S.M., Khoo, K., Oggier, F.E., Peyrin, T.: Lightweight MDS involution matrices. In: Leander, G. (ed.) Fast Software Encryption - 22nd International Workshop, FSE 2015, Istanbul, Turkey, March 8-11, 2015, Revised Selected Papers. Lecture Notes in Computer Science, vol. 9054, pp. 471–493. Springer (2015), https://doi.org/10.1007/978-3-662-48116-5_23
36. Stoffelen, K., Daemen, J.: Column parity mixers. IACR Trans. Symmetric Cryptol. **2018**(1), 126–159 (2018), https://doi.org/10.13154/tosc.v2018.i1.126-159
37. Venkateswarlu, A., Kesarwani, A., Sarkar, S.: On the lower bound of cost of MDS matrices. IACR Trans. Symmetric Cryptol. **2022**(4), 266–290 (2022), https://doi.org/10.46586/tosc.v2022.i4.266-290

A Automated Models for Analysis of Trails in Gaston

In order to obtain bounds on the differential and linear branch number of Gaston, we use Satisfiability Modulo Theories (SMT) and Mixed-Integer Linear Programming (MILP) in a hybrid manner [30]. We discuss the construction of the models for both automated tools followed by the discussion on the obtained bounds. We restrict our discussion on the model for differential trails since the one for linear trails is analogous. The source codes for both differential and linear trails are available at https://gitlab.science.ru.nl/selhirch/circulanttwincpm.

A.1 MILP Model for Computing Differential Weight

Our MILP modeling consists of three steps, namely 1) model constraints for a single round, i.e., 5-bit χ S-box and the linear layer $\lambda = \rho_{west} \circ \theta \circ \rho_{east}$, 2) add constraints for multiple rounds by extending the previous step, and 3) add the objective function to minimize the weight.

Constraints for χ. Let $\{x_i\}_{i=0}^{4}$ and $\{y_i\}_{i=0}^{4}$ be binary variables denoting the input and output difference of χ, respectively. We use the approach from [1, Appendix B] to model weights of differential transitions. More precisely, we first divide the DDT of χ into multiple DDTs based on weights. Then for each DDT, we compute the minimized product-of-sum (POS) representation of Boolean functions using Logic Friday. Each POS form is then converted into a set of linear inequalities where these inequalities remove the impossible propagations in DDT [1, Sec. 3]. Finally, the differential transitions via χ can be modeled by the following inequalities.

$$\begin{aligned}
x_0+x_1+x_2+x_3+x_4 \geq d, \quad x_0+x_1+x_2+x_3+x_4 \leq 5\cdot d \\
y_0+y_1+y_2+y_3+y_4 \geq d, \quad y_0+y_1+y_2+y_3+y_4 \leq 5\cdot d \\
w^{(2)}+w^{(3)}+w^{(4)} = d \\
\sum_{i=0}^{4} a_i^j x_i + \sum_{i=0}^{4} b_i^j y_i + c^j - 10\cdot w^{(2)} \geq 0,\ \text{for } j=0,\ldots,\ell(w^{(2)})-1 \\
\sum_{i=0}^{4} d_i^j x_i + \sum_{i=0}^{4} e_i^j y_i + f^j - 10\cdot w^{(3)} \geq 0,\ \text{for } j=0,\ldots,\ell(w^{(3)})-1 \\
\sum_{i=0}^{4} g_i^j x_i + \sum_{i=0}^{4} h_i^j y_i + k^j - 10\cdot w^{(4)} \geq 0,\ \text{for } j=0,\ldots,\ell(w^{(4)})-1\,.
\end{aligned} \tag{10}$$

In (10), $d, w^{(2)}, w^{(3)}, w^{(4)}$ are binary variables. We enforce $d = 1$ if and only if the S-box is active. The variables $w^{(2)}, w^{(4)}, w^{(4)}$ mean the difference propagates through χ with weight either 2, 3 and 4, respectively. The number of linear inequalities (number of terms in the POS representation) corresponding to weight 2, 3 and 4 is given by $\ell(w^{(2)})$, $\ell(w^{(4)})$ and $\ell(w^{(4)})$, respectively. We provide the values of $\ell(w^{(2)})$, $\ell(w^{(4)})$, $\ell(w^{(4)})$ and coefficients (a_i^j, b_i^j, c^j), (d_i^j, e_i^j, f^j) and (g_i^j, h_i^j, k^j) in our codes.

Constraints for Linear layer. Let $\{y_i\}_{i=0}^{319}$ and $\{x_i\}_{i=0}^{319}$ be the input and output difference of the linear layer, respectively. Since the linear layer is a composition of different operations (see Sec. 3.1 and 3.2), we explain the modeling procedure for each individual operation. For simplicity, in this section we use the notation $[i + j]$ which means $i + j \bmod 64$.

State after ρ_{east}. This operation performs left cyclic rotation on each row of the input state with shift offset vector $\mathbf{R}_e = (e_0, e_1, e_2, e_3, e_4)$. Accordingly, there are no new constraints and the updated state is given by

$$z_{64\cdot j+i} \leftarrow y_{64\cdot j+\,[e_j+i]}, \text{ for } i = 0, \ldots, 63 \text{ and } j = 0, \ldots, 4\,. \tag{11}$$

The P Operation. For each column of the state in (11), this operation computes the column-parity. To model it, we simply need to consider the bitwise XOR modeling of 5 binary variables. For each column, this can be done as follows.

$$z_i + z_{64+i} + z_{128+i} + z_{192+i} + z_{256+i} + p_i = 2 \cdot u_i, \text{ for } i = 0, \ldots, 63\,. \tag{12}$$

Here $p_i \in \{0, 1\}$ and $u_i \in \{0, 1, 2, 3\}$. Notice that $p_i = 1$ if and only if the Hamming weight of the i-th column, i.e., $(z_i, z_{64+i}, z_{128+i}, z_{192+i}, z_{256+i})$ is odd.

The Q Operation. This operation has two steps. First, the rows 0, 1, 2, 3 and 4 of state in (11) are cyclically shifted to the left by t_0, t_1, t_2, t_3 and t_4, respectively. Second, we compute the column-parity of this state. Thus, similar to the P operation, we need to model the bitwise XOR modeling of 5 binary variables. For columns $i = 0, \ldots, 63$, the constraints are as follows.

$$z_{[i+t_0]} + z_{64+[i+t_1]} + z_{128+[i+t_2]} + z_{192+[i+t_3]} + z_{256+[i+t_4]} + q_i = 2 \cdot v_i\,.$$

Here $q_i \in \{0, 1\}$ and $v_i \in \{0, 1, 2, 3\}$.

The E Operation. This operation computes the θ-effect using variables $\{p_i\}_{i=0}^{63}$, $\{q_i\}_{i=0}^{63}$ and their shifted versions. To model it, we have the following constraints.

$$p_i + p_{[i+r]} + q_i + q_{[i+s]} + g_i = 2 \cdot b_i, \text{ for } i = 0, \ldots, 63\,. \tag{13}$$

In (13), we have $g_i \in \{0, 1\}$, $b_i \in \{0, 1, 2\}$ and (r, s) are the shift offsets from $\mathbf{R}_\theta$.

State after XORing Shifted Version of Variables $\{g_i\}_{i=0}^{63}$. Given the shift offset u from $\mathbf{R}_\theta$, consider $\{g_{[i+u]}\}_{i=0}^{63}$. Let $\{f_i\}_{i=0}^{319}$ be binary variables denoting state after XORing $\{g_{[i+u]}\}_{i=0}^{63}$ to each row in (11). We have the following constraints for this operation.

$$\left.\left.\begin{aligned} g_{[i+u]} + z_{64\cdot j+i} - f_{64\cdot j+i} &\geq 0 \\ g_{[i+u]} - z_{64\cdot j+i} + f_{64\cdot j+i} &\geq 0 \\ -g_{[i+u]} + z_{64\cdot j+i} + f_{64\cdot j+i} &\geq 0 \\ -g_{[i+u]} - z_{64\cdot j+i} - f_{64\cdot j+i} &\geq -2 \end{aligned}\right\} \text{ for } i = 0, \ldots, 63 \right\} \text{ for } j = 0, \ldots, 4 \tag{14}$$

Note that the four inequalities in (14) remove the impossible propagations $(0, 0, 1)$, $(0, 1, 0)$, $(1, 0, 0)$ and $(1, 1, 1)$.

State after ρ_{west}. This operation performs left cyclic rotation on each row of the input state $\{f_i\}_{i=0}^{319}$ with shift offset vector $\mathbf{R}_w = (w_0, w_1, w_2, w_3, w_4)$. The output state is given as follows.

$$x_{64\cdot j+i} \leftarrow f_{64\cdot j+[w_j+i]}, \text{ for } i = 0, \ldots, 63 \text{ and } j = 0, \ldots, 4\,. \tag{15}$$

Constraints for Multiple Rounds. We add the aforementioned constraints for 64 S-boxes and for the linear layer in each round. Furthermore, we add a constraint that the sum of input active bits is at least 1 in order to ensure a nonzero input difference.

Objective Function. Let $w_{i,0}^{(2)}, \dots, w_{i,63}^{(2)}$, $w_{i,0}^{(3)}, \dots, w_{i,63}^{(3)}$, $w_{i,0}^{(4)}, \dots, w_{i,63}^{(4)}$ be the variables denoting the activity (along with weights) of 64 S-boxes at round i. Then for $\mathbf{r}$ rounds, the objective function as given in (16) is to minimize the weight of a $\mathbf{r}$-round differential trail.

$$\min \Big(\sum_{i=0}^{\mathbf{r}-1} \sum_{j=0}^{63} 2 \cdot w_{i,j}^{(2)} + 3 \cdot w_{i,j}^{(3)} + 4 \cdot w_{i,j}^{(4)} \Big) \tag{16}$$

A.2 SMT Modeling

Constraints for the Linear Layer. The SMT constraints for the differential (resp. linear) propagation of θ is immediate to derive from its definition in (2) (resp. its transpose in (9)). This is also the case for the differential propagation of ρ_{east} and ρ_{west}, in which their transpose for the linear propagation performs bitwise rotation in the opposite direction.

Constraints for χ. For the differential propagation of the nonlinear layer χ, we construct the set of constraints for χ based on a Boolean function, say f, that defines the (in)-validity of a differential. Concretely, for any nonzero weight differential (x, y), we define $f(x\|y) = 1$ and 0 otherwise, where $x\|y$ denotes the vector concatenation of x and y. The constraints that represent f can be constructed in a row-by-row basis from the DDT of χ. The inclusion of the weight in the model is immediate to derive using an auxiliary variable that stores the weight from a given differential. Note that this approach is generic and works for any small-size permutation.

Reducing Search Time. Due to the complexity of the function θ in Gaston, the SMT solver takes a substantial amount of time to find a trail of weight less than a chosen bound, even for a satisfiable instance of a small number of rounds. One strategy to reduce the search time of a SMT solver is to add new restrictions that simplify the constraints. We are particularly interested in a set of trails where the effect of θ on a state at a particular round is minimal such as the states that lie in the θ-kernel.

B Differential and Linear Trails of Gaston

B.1 Differential Trails

Figure 1 depicts the optimal 2-round differential trail of Gaston while Fig. 2 gives a differential trail for 3-round with the best found weight of 106. Note that this may not be the optimal weight of a 3-round trail.

```
................................................................
..........1...................................1.................
1...............................................................
.....1.......1...........................1......................
................................................................
---------------------------------------------------------------- χ
................................................................
..........1...................................1.................
1...............................................................
.....1.......1...........................1......................
................................................................
---------------------------------------------------------------- ρ_east
................................................................
..............1...................................1.............
..........................................1.....................
..............1...........................1.......1.............
................................................................
---------------------------------------------------------------- θ
................................................................
..............1...................................1.............
..........................................1.....................
..............1...........................1.......1.............
................................................................
---------------------------------------------------------------- ρ_west
................................................................
......................1...................................1.....
...........1....................................................
....1...........................1...........................1...
................................................................
---------------------------------------------------------------- χ
...........1....................................................
...........1..........1...................................1.1...
...........1................................................1...
....1...........................1...........................1...
......................1.........................................
```

Fig. 1: 2-round differential trail with 12 [6, 6] active S-boxes and weight 24 [12, 12]

B.2 Linear Trails

Figure 3 depicts the optimal 3-round linear trail of Gaston with weight 34.

C Inverse of θ

In Fig. 4 we illustrate the density of the inverse of θ: it corresponds to the preimage of a state that has a single active bit at position $(0, 0)$. Other states with a single active bit have preimages with similar shape and Hamming weight.

D Test Vectors

```
.............................................1.................1
...........................................................1....
......................................1.................1.......
...........................................1..........1......1..
...............................1.................1..............
---------------------------------------------------------------- χ
.............................................1.................1
...........................................................1....
......................................1.................1.......
...........................................1..........1......1..
...............................1.................1..............
---------------------------------------------------------------- ρ_east
.............................................1.................1
...............................................................1
................1.................1.............................
................1..........1......1.............................
...........................1.................1..................
---------------------------------------------------------------- θ
.............................................1.................1
...............................................................1
................1.................1.............................
................1..........1......1.............................
...........................1.................1..................
---------------------------------------------------------------- ρ_west
.............................................1.................1
.......1........................................................
...1.............................................1..............
..................................1..........1......1...........
..1.............................................1...............
---------------------------------------------------------------- χ
...1.........................................1.................1
.......1........................................................
...1..............................1.............11..1...........
..................................1.............1...1...........
..1.............................................1...............
---------------------------------------------------------------- ρ_east
...1.........................................1.................1
...........1....................................................
............1.............11..1..............1..................
.......1.............1...1......................................
............................................1.................1.
---------------------------------------------------------------- θ
...1...1.............1................1.....11.....1.1.........1
.......1...1.........1................1.....1......1.1..........
.......1....1........1....11..1.......1.....11.....1.1..........
.........................1............1.....1......1.1..........
.......1.............1................1............1.1........1.
---------------------------------------------------------------- ρ_west
...1...1.............1................1.....11.....1.1.........1
...............1...1.........1................1.....1......1.1..
.......1.....11.....1.1.................1....1........1....11..1
.....1.1...................................1............1.....1.
........1.1........1........1.............1................1....
---------------------------------------------------------------- χ
...1.........11......1................1.....1......1.1.....1.1..
..............11...1.........1..........1..1..1.....1........11.
.............11.....1.1.....1...........1..1.1........1....11.11
.....1.1...................................1.1..........1.....11
........1.1.................1.............1..................1..
```

Fig. 2: 3-round differential trail with 50 [10, 9, 31] active S-boxes and weight 106 [20, 19, 67]

```
............................1...............................1...
...1.........................1..................................
...1.........................1..................................
............................................................1...
...........1....................1...........................1...
----------------------------------------------------------------  χ
............................1...................................
...1.........................1..................................
................................................................
............................................................1...
...........1....................1...............................
----------------------------------------------------------------  ρ_east
............................1...................................
.......1.........................1..............................
................................................................
.................................1..............................
.......1....................1...................................
----------------------------------------------------------------  θ
............................1...................................
.......1.........................1..............................
................................................................
.................................1..............................
.......1....................1...................................
----------------------------------------------------------------  ρ_west
............................1...................................
...............1.........................1......................
................................................................
...................................................1............
............................1....................1..............
----------------------------------------------------------------  χ
................................................................
...............1.........................1......................
.........................................1......................
...................................................1............
............................1....................1..............
----------------------------------------------------------------  ρ_east
................................................................
...................1.........................1..................
...................1............................................
........................1.......................................
........................1....................1..................
----------------------------------------------------------------  θ
................................................................
...................1.........................1..................
...................1............................................
........................1.......................................
........................1....................1..................
----------------------------------------------------------------  ρ_west
................................................................
...........................1.........................1..........
....................................................1...........
..........................................1.....................
..1..........................................1..................
----------------------------------------------------------------  χ
.............................................1..................
...........................1.........................1..........
....................................................11..........
..........................................1.........1...........
..1.......................................1..1..................
```

Fig. 3: 3-round linear trail with [6, 5, 6] active S-boxes and weight 34 [12, 10, 12]

```
1001011011110010010010100100100001011100100101111100001000011110
0100000010101100000011011000110000001100100001011110101001000011
0111011100101010101101111100110111001100011011001110010010000100
1100101000010111100000110001010100001100111011101101000010011110
1010110110011010100101001001000011001010111100000101110001100000
```

Fig. 4: θ-preimage of a state with a single active bit at $(0, 0)$, with θ the mixing layer in Gaston

0x0000000000000000	0x88B326096BEBC635
0x0000000000000000	0x6CA8FB64BC5CE6CA
0x0000000000000000	0xF1CE3840D8190713
0x0000000000000000	0x54D70067438689B5
0x0000000000000000	0xF17FE863F958F32B
0x1F4AD9906DA6A254	0x1BA89B5B5C4583B6
0x4B84D7F83F2BDDFA	0x22135709AE53417D
0x468A0853578A00E3	0x9847B975E9EC9F3D
0x6C05A0506DF7F66E	0xCE042DF2A402591D
0x4EFB22112453C964	0x563EC68FC30307EA
0xFFFFFFFFFFFFFFFF	0x3117D51B14937067
0x0123456789ABCDEF	0x338F17F773C13F79
0xFEDCBA9876543210	0xDFB86E0868D252AB
0xAAAAAAAAAAAAAAAA	0x0D461D35EB863DE7
0x0101010101010101	0x08BCE3E354C7231A

Table 7: Test vectors for 12-rounds of Gaston. The input and its coresponding output are on the left-side and the right-side respectively.