New Techniques for Searching Differential Trails in Keccak

Guozhen Liu\textsuperscript{2,1}, Weidong Qiu\textsuperscript{2} and Yi Tu\textsuperscript{1}

\textsuperscript{1} Division of Mathematical Sciences, School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore, Singapore, tuyi0002@e.ntu.edu.sg
\textsuperscript{2} School of Cyber Science and Engineering, Shanghai Jiao Tong University, Shanghai, China, liuguozhen@sjtu.edu.cn, qiuwd@sjtu.edu.cn

\textbf{Abstract.} Keccak-f is the permutation used in the NIST SHA-3 hash function standard. Inspired by the previous exhaustive differential trail search methods by Mella \textit{et al.} at ToSC 2017, we introduce in this paper new algorithms to cover 3-round trail cores with propagation weight at least 53, up from the previous best weight 45.

To achieve the goal, the concept of ideal improvement assumption is proposed to construct theoretical representative of subspaces so as to efficiently cover the search space of 3-round trail cores with at least one out-Kernel $\alpha$ state. Of particular note is that the exhaustiveness in 3-round trail core search of at least one out-Kernel $\alpha$ is only experimentally verified. With the knowledge of all 3-round trail cores of weight up to 53, lower bounds on 4/5/6-round trails are tightened to 56/58/108, from the previous 48/50/92, respectively.

\textbf{Keywords:} SHA3 · Keccak-f · Differential Trail Search · Exhaustive Search · Lower Bound of Differential Trail

\section{Introduction}

\subsection{The birth of Keccak}

Following the breakthrough of cryptanalysis against a series of standards such as MD5 [WY05], SHA-0 [WYY05b], and SHA-1 [WYY05a], the U.S. National Institute of Standards and Technology (NIST) organized the SHA-3 competition (2008–2012), which received 64 submissions. Keccak [BDPV11] designed by Bertoni \textit{et al.} won the competition in 2012, and was later formally adopted as the Secure Hash Algorithm 3 (SHA-3) standard in 2015 [The15]. The idea is to have a new standard with different design principle from SHA-1 and SHA-2, hence in the very unlikely event of similar issues to SHA-1 found on SHA-2, there will be a backup solution SHA-3.

The SHA-3 family has four main instances with fixed digest lengths, namely, SHA3-224, SHA3-256, SHA3-384, and SHA3-512, which correspond to Keccak[c] $\triangleq$ Keccak[r = 1600 $-$ c, c] where $c \in \{448, 512, 768, 1024\}$ are the capacity sizes in bits. There are also two extendable-output functions named SHAKE128 and SHAKE256, which support digest of variable sizes and collision resistance of level up to 128 and 256 bits, respectively. All hash functions of the SHA-3 family share the same underlying permutation function Keccak-f of 1600-bit internal state. To promote cryptanalysis, the designers of Keccak also proposed variants with capacity of 160 bits, as well as the underlying Keccak-f permutation of sizes in $\{200, 400, 800, 1600\}$, under the Keccak Crunchy Crypto Collision and Pre-image Contest [BDH\textsuperscript{+}]. The same Keccak-f permutation (or its variant) could be used to construct authenticated encryption such as Keyak [BDP\textsuperscript{+}16b] and Ketje [BDP\textsuperscript{+}16a],
message authentication code such as KMAC [The16], as well as key derivation function such as Kravatte [DHVV18].

1.2 The security status of Keccak

Since the birth, Keccak has received a significant amount of cryptanalysis from the public research community, and generally the results can be divided by the following security notions. So far, there is no attack threatening the security of Keccak, and actually all attacks work on Keccak variants reduced to at most 9 out of the total 24 rounds, leaving a large security margin of more than 62% of the rounds.

Collision Resistance is regarding the sponge hash functions based on Keccak-f and its variants. In 2012, Dinur et al. [DDS12] found then the best practical collision attacks against Keccak[448]/Keccak[512] reduced to 4 rounds. The results was later furnished in the journal version [DDS14b]. These 4-round collisions were generated by combining a 1-round connector and a 3-round differential trail. The same authors presented practical collision attacks on 3-round Keccak[768]/Keccak[1024], and theoretical collision attacks (with complexities beyond the reach of practical resources) on 5/4-round Keccak[512]/Keccak[768] in [DDS14a] using internal differentials. Qiao et al. [QSLG17] extended the idea of connector from 1 round to 2 rounds, extending the collision attacks to 5 rounds. Song, Liao, and Guo [SLG17] further extended the connectors to 3 rounds and found the first practical collision against one 6-round variant in the Keccak collision contest. These results were then summarized in [GLL+19], where practical collisions of 5-round SHAKE128, SHA3-224, and SHA3-256 were presented. Up to date, there exists no attack against any SHA-3 variants reduced to 6 rounds or more.

Preimage Resistance is also regarding the sponge hash functions. In 2016, Guo, Liu, and Song [GLS16] proposed the linear structure and found preimage attacks up to 4 rounds, including practical solutions to the 4-round Keccak[1440, 160] and the 3-round Keccak[1440, 160] and Keccak[640, 160] of the Keccak preimage contest. This was later extended by Li et al. [LSLW17] to the so-called cross-linear structure, and found its application in improving the complexities of some attacks, including the first practical preimage on 3-round Keccak[240, 160]. Further complexity improvements were found by Li and Sun [LS19]. In summary, the best preimage attacks, in terms of number of attacked rounds, are for no more than 4 rounds.

Key Recovery is regarding the keyed constructions such as authenticated encryption schemes, message authentication codes, and key derivation functions. The most effective attacks so far against the Keccak-f-based keyed constructions are cube attacks [DLWQ17, DMP15, HWX17, LBDW17, SGSL18, BDL19], taking advantage of the low algebraic degree 2 (or its inverse 3) of the Keccak round function. Among them, the best attack works for up to 9 rounds.

Distinguishers. There are also distinguishers against the permutation Keccak-f itself, under models like limited-birthday [DGPW12] and zero-sum [AM09, GLS16]. While the limited-birthday distinguisher is based on differentials, which worked for up to 8 rounds with complexities bounded by $2^{512}$, zero-sum distinguisher takes advantage of the low algebraic degrees of the Keccak round function and its inverse and worked for 11 rounds practically and 14 rounds with complexity $2^{129}$.

1.3 Differential trail search

Differential trails are a key part of the security analysis against symmetric-key primitives, e.g., they are the key factors deciding the effectiveness, in terms of rounds and attack complexities, of the attacks for collisions and differential-based distinguishers. However,
unlike AES, there is no known tight bound for differential trails of Keccak-f due to its weak aligned round function. The weight $W$ of the best differential trails refers to the smallest value $\log_2(1/p)$, where $p$ is the differential probability. $W$, for a given number of rounds, can be lower bounded by some theoretical reasoning such as counting the minimum number of active Sboxes in all differential trails, and upper bounded by differentials actually found. There have been continuous efforts in closing the gap between the lower and upper bounds for Keccak-f, mostly by finding effective ways to enumerate all possible differentials falling in the gap.

It is trivial to see $W[r=1] = 2$ and $W[r=2] = 8$ (the minimum weight for 1-round and 2-round differential trails). The Keccak designers [BDPV11] made the first effort in providing a tighter bounds, and $W[r=3]$ is proven to be 32 — a tight bound by Daemen and Van Assche in [DV12] with a matching differential trail found by Duc et al. in [DGPW12]. A more sophisticated technique was proposed by Mella-Daemen-Van Assche in [MDV17] to exhaustively search 3-round differential trails up to weight 45, which was used to give tighter bounds for differential trails of 4, 5, and 6 rounds.

1.4 Our contributions

Following the direction of [MDV17], this paper focuses on exhaustively enumerating 3-round differential trails of Keccak-f to higher weight. The search space dramatically increases with the weight, hence the main difficulty to overcome here is to find an efficient way to quickly eliminate differential trails out of the target range, and hence to reduce the search space into one small enough for exhaustive search in practice. Two main new techniques are proposed in this paper: idealized differentials to enable early abortion and a careful division of the subcases.

Depending on whether the state falls in the Column-Parity Kernel (Kernel for short), the two consecutive rounds of the differential are divided into 4 cases, i.e., $[K][K]$, $[K][N]$, $[N][K]$, and $[N][N]$\footnote{\(K\) (resp. \(N\)) denotes the state is in (resp. out) Kernel.}. In searching 3-round trail cores with $[K][K]$ feature, rather than enumerating in-Kernel $\alpha_1$ by recursively adding orbitals as done in [DV12], we construct candidate $\alpha_1$ based on $\beta_1$ structures that ensure the resulting $\alpha_2$'s are in-Kernel. For in-Kernel $\alpha_1$ of $m$ orbitals, through $\pi \circ \rho$, by examining how the $2m$ bits are distributed to $\beta_1$ slices, the compatible in-Kernel $\alpha_2$ are enumerated. With such candidate $\beta_1$ structures, $\alpha_1$ are constructed with data complexity $2^{12}$, leading to exhaustive coverage of 3-round trail cores of weight no greater than 53.

In searching 3-round trail cores with at least one out-Kernel $\alpha$, i.e., 3-round trail cores with $[N][K]$, $[N][N]$ or $[K][N]$ feature, the search space of out-Kernel states $\alpha$ is divided into subspaces $V_\alpha$ derived from $\alpha$. Each subspace has a theoretical representatives in terms of the number of active rows of all 3-round trail cores. The theoretical representatives are generated under the ideal improvement assumption which are deduced to construct the theoretically optimal state of the subspace. Therefore the subspace can be efficiently pruned with constraints on its theoretical representative. With this new search strategy, 3-round trail cores of weight up to 53 can be covered. Note that the exhaustive search with threshold weight 53 is only experimentally proved of which the technical details will be illustrated in Section 5.

With the help of the exhaustively searched 3-round trail cores of weight up to 53, lower bound of propagation weight of 4/5/6-round trails is tightened to 56/58/108 from the previous best result 48/50/92 in [MDV17], respectively.

Paper Organization. The paper is organized as follows. In Section 2, brief introduction of Keccak-f[1600] as well as propagation properties of linear operations used in deducing search algorithms are given. In Section 3, the basic concepts of differential trail cores and how they are classified in exhaustive search are illustrated. The search strategies of $[K][K]$,
\(|N|K|\) and \(|N|N|\), and \(|K|N|\) 3-round trail cores are explained in Section 4, 5 and 6 respectively. In Section 7, the valid 3-round trail cores searched with our new techniques are provided and discussed. We conclude this work in Section 8. Some additional details of the analysis are postponed to Appendix.

2 Keccak and Propagation Properties

In this section, a brief introduction to Keccak-\(f[1600]\) is given at first. Some frequently-used concepts, notations, and properties used for deducing the search strategies are presented as well.

2.1 Keccak

Keccak [BDPV11] is a family of sponge functions with permutations Keccak-\(f[b]\) where \(b \in \{25, 50, 100, 200, 400, 800, 1600\}\}. We study the differential propagation of Keccak-\(f[1600]\) which is the SHA-3 permutation.

The 1600 bits of its state are organized in a 3-dimensional array of \(5 \times 5 \times 64\) size. Each bit is denoted by a \((x, y, z)\) coordinate where \(0 \leq x \leq 4\), \(0 \leq y \leq 4\) and \(0 \leq z \leq 63\). To better study the round function, structures that contain partial bits of the state are introduced as shown in Fig 1. A group of 5 bits that share the same \(y\) (resp. \(x\)) coordinate is called a row (resp. column), denoted by \((x, z)\) (resp. \((y, z)\)), while a lane consists of 64 bits of the same \((x, y)\) coordinates. The \(5 \times 5\) bits of the same \(z\) coordinate form a slice. Other structures include plane and sheet.

For structures composed of columns, i.e., column, slice, sheet and state, parity is defined. Parity, denoted by \(p\), is the XOR of all 5 bits of a column. Thus parity of a column, i.e., \(p(x, z)\) is a bit. If \(p(x, z) = 0\) (or \(p(x, z) = 1\)), there are even (or odd) number of active bits in column \((x, z)\). The column that has an odd (or even) number of active bits is called an odd (or even) column. For a slice, its parity \(p(z)\) is a \(1 \times 5\) row and for a state \(a\), the parity \(p(a)\) is a \(5 \times 64\) plane.

The round function of Keccak-\(f[1600]\) consists of 5 operations, i.e., the linear components \(\theta\), \(\rho\), \(\pi\) and \(\iota\), and the non-linear component \(\chi\).

- \(\theta\) adds a pattern to each bit position \((x, y, z)\). The pattern which is defined as \(\theta\)-effect in [BDPV11,DV12,MDV17] is the XOR of the parity of column \((x-1, z)\) and column \((x+1, z-1)\).
- \(\rho\) rotates bits on lane \((x, y)\) with an offset defined by \((x, y)\).
- \(\pi\) rearranges the \(5 \times 5 = 25\) bits on one slice.
- \(\iota\) adds a constant to each round. It is always omitted in the analysis as it has no influence on differential propagation.
- \(\chi\) is a 5-bit S-box. It is the only non-linear operation in Keccak-\(f[1600]\).

2.2 Propagation properties of linear layer operations

The linear layer of one round is composed of three components, i.e., \(\theta\), \(\rho\), and \(\pi\), denoted by \(\lambda = \pi \circ \rho \circ \theta\). Propagation properties of those operations are described in the following paragraphs.
Figure 1: Different structures of round state.
2.2.1 Propagation property of $\theta$

Out of the four linear components, only $\theta$ operation may alter the number of active bits. Given a state $\alpha$ if its parity $p(\alpha) = 0$ where $p(0)$ is a $5 \times 64$ zero vector, $\theta$ acts as an identity function, i.e., there is no $\theta$-effect on the state. State with zero parity is called in-Kernel, denoted by $K$. Otherwise, it is out-Kernel, denoted by $N$. The number of active bits of in-Kernel states is preserved through $\theta$.

However, if $p(\alpha)$ is a nonzero vector, the $\theta$-effect really complicates the propagation of active bits. To better understand $\theta$-effect of out-Kernel states, works [BDPV11, DV12, MDV17] introduced run to reorganize the nonzero parity. If the coordinates of a group of odd columns satisfy that $z_{i+1} = (z_i + 1) \mod 64$ and $x_{i+1} = (x_i + 3) \mod 5$, where $(x_i, z_i)$ and $(x_{i+1}, z_{i+1})$ are any continuous odd columns, this collection of odd columns form a run. According to the definition of $\theta$ operation, no matter how many odd columns are in a run, it only affects two columns. The affected columns whose active bits will be flipped through $\theta$ are called affected columns. Parity thus can be partitioned by a series of runs. Fig. 2 shows two parities of 1-run and 2-run as well as the affected columns.

![Figure 2: Examples of a 1-run and a 2-run parity.](image)

On the other hand, we can construct parities by listing all the possible combinations of runs. In later differential trail search sections, parities of 1-run, 2-run and 3-run are enumerated. For any out-Kernel difference, adding two active bits to a column (if it is possible), its parity remains unchanged. Practically, a group of 2 active bits that are in the same column is called an orbital. For in-Kernel differences, its active columns contain several orbitals.

Given a $k$-run parity of states $\alpha$, parity-bare states are introduced to represent all states that share the same parity out-Kernel in regarding to differential probability of $\chi^{-1}(\alpha)$ and $\chi(\beta)$. Here $\alpha$ are out-Kernel input differences of $\theta$ and $\beta = \lambda(\alpha)$ is the input difference of $\chi$. As introduced in [MDV17], of all the out-Kernel differences under the parity, parity-bare states have the least number of active bits in differences $\alpha$ and $\beta$, i.e., their hamming weight $||\alpha|| + ||\beta||$ is minimal. $||\cdot||$ denotes the number of active bits of a structure which is also called the hamming weight. Given a $k$-run parity, by adding orbitals to its parity-bare states all out-Kernel differences under the parity are obtained.

2.2.2 $\pi \circ \rho$-effect

While the $\theta$-effect flips bits on affected rows of the out-Kernel states, $\rho$ and $\pi$ are only permutations of bit positions. The $\pi \circ \rho$-effect is the joint effect of $\rho$ and $\pi$ operations imposed on the active bits of state after $\theta$-effect. For example, under the Eq. 1 model, $\alpha$ is transformed to $\beta^\theta$ through $\theta$-effect, and $\beta^\theta$ is permuted to $\beta$ through $\pi \circ \rho$-effect.

$$\alpha \xrightarrow{\theta} \beta^\theta \xrightarrow{\rho} \beta^\rho \xrightarrow{\pi} \beta \quad (1)$$

$^2$In this example $z = 4$ rather than 64.
\( \pi \circ \rho \)-effect considers the bit positions before and after the associated linear functions. For example, given a bit of \( \beta^\theta \), the corresponding bit coordinates, row index, column index, and slice index at \( \beta \) of the specific bit are carefully examined. Similarly, given a bit of \( \beta \), the origin bit coordinates, row index, column index, and slice index at \( \beta^\theta \) are checked as well.

When \( \alpha \) is in-Kernel \( \theta \) becomes an identity function. In other words, there is only \( \pi \circ \rho \)-effect at such cases which simplifies the propagation of the complex \( \lambda \).

The \( \pi \circ \rho \)-effect is used to bypass \( \pi \circ \rho \). Given \( \beta^\theta \), one does not have to operate \( \pi \circ \rho \) on the whole state. Rather, with only the active bits, the active columns, rows, and slices are easily obtained which is more advantageous when \( \alpha \) is in-Kernel. Generally, \( \pi \circ \rho \)-effect has the following use scenarios in the subsequent trail search strategy.

- When \( \alpha \) is in-Kernel, the number of active rows, i.e., \( \text{Row}_\beta \) is obtained through combining the row index of each bit.
- When \( \alpha \) is out-Kernel with orbitals added to the non-affected columns, the number of added rows \( \text{AddedRow}_\beta \), is also obtained by combining the added row index with the previous row index set.
- Given \( \beta \), if the corresponding \( \alpha \) is supposed to be in-Kernel, check the origin column index of \( \beta \) bits to verify whether it has in-Kernel \( \alpha \).

\( \pi \circ \rho \)-effect is widely used to efficiently count the number of active rows in the trail search strategy.

### 2.2.3 \( \rho \) property

In Section 4, a property of \( \rho \) explained in Lemma 1 is used in searching 3-round \( |K|K \) trail cores.

**Lemma 1.** For any in-Kernel \( \alpha \) with \( m \) orbitals, through \( \pi \circ \rho \), there are at most \( m \) bits in any active \( \beta \) slices.

**Proof.** From the definition of \( \rho \) operation, it is direct that the 25 bits of any \( \alpha \) slices must be distributed to 25 different \( \beta \) slices and vice versa. As the \( m \) orbitals are at most placed in \( m \) slices the number of active bits of any \( \beta \) slices must not exceed \( m \). □

### 3 3-Round Differential Trail Cores

Differential trail cores and the classification of 3-round trail cores are introduced in this section. Most of the concepts are brought from previous works [BDPV11, DV12, MDV17].

#### 3.1 3-round trail cores

A 3-round differential trail, denoted by \((\alpha_0, \alpha_1, \alpha_2)\) (or \((\beta_0, \beta_1, \beta_2)\)) is shown in Eq. 2, where \( \alpha_i \) and \( \beta_i \) are input difference and output difference of \( \lambda \) respectively. It contains three \( \chi \) layers with differential propagation probability \( P_3 = P(\beta_0) + P(\beta_1) + P(\beta_3) \). In our analytic context, propagation weight, i.e., \( w_3 = w(\beta_0) + w(\beta_1) + w(\beta_2) \), is extensively used to represent differential probability of trails.

\[
\begin{align*}
\alpha_0 & \xrightarrow{\lambda} \beta_0 & \xrightarrow{\chi} \alpha_1 & \xrightarrow{\lambda} \beta_1 & \xrightarrow{\chi} \alpha_2 & \xrightarrow{\lambda} \beta_2 & \xrightarrow{\chi} \alpha_3 
\end{align*}
\] (2)

Comparing to 3-round trails, 3-round trail cores \((\alpha_1, \alpha_2)\) (or \((\beta_1, \beta_2)\)) shown in Eq. 3 contain only two differences rather than three. Given any \( \alpha_1 \), the minimal \( \chi \)-compatible propagation weight \( w(\beta_0) \) which is formally denoted by \( w^{\text{rev}}(\alpha_1) \), is obtained simply by
checking the differential distribution table. Therefore a 3-round trail core represents a set of 3-round trails among which the minimal propagation weight is \( w_3 = w^{rev}(\alpha_1) + w(\beta_1) + w(\beta_2) \).

\[
\alpha_1 \xrightarrow{\lambda} \beta_1 \xrightarrow{\chi} \alpha_2 \xrightarrow{\lambda} \beta_2
\]  

Similarly, 2-round trail cores are denoted by \((\alpha_1)\) or \(\alpha\) for general. We also simply call a 2-round trail core a \(\alpha\) state in introducing ideas and methods. In previous works [DV12, MDV17], 2-round trail cores \((\alpha_1)/(\alpha_2)\) (or \((\beta_1)/\beta_2)) are exhaustively searched and extended forward or backward by one round to obtain 3-round trail cores. The search space is so huge that only 3-round trail cores with propagation weight no greater than a threshold weight \(T_3\) are covered.

To exhaustively generate 3-round trail cores with a greater threshold weight \(T_3\), namely the valid 3-round trail cores, we follow the previous rule of first collecting possible 2-round trail cores and then extending them forward or backward by one round. A 2-round trail core \(\alpha_1\) (resp. \(\alpha_2\)) is called valid when there exists at least one \(\chi\)-compatible \(\alpha_2\) (resp. \(\alpha_1\)) that satisfies \(T_3\).

Additionally, the number of active rows is used to evaluate the propagation weight of 3-round trail cores as the best differential probability (or propagation weight) of an active row is \(2^{-2}\) (or 2). For example, the number of active rows of valid 3-round trails must satisfy that \(\text{Row}_3 \leq \lceil \frac{T_3}{2} \rceil\) where \(\text{Row}_3 = \text{Row}_{\alpha_1} + \text{Row}_{\alpha_2} + \text{Row}_{\beta_2}\). The number of active rows of a difference state \(\alpha\) is denoted by \(\text{Row}_{\alpha}\).

### 3.2 Classification of 3-round trail cores

According to whether \(\alpha_1\) and \(\alpha_2\) are in-Kernel or not (refer to Eq. 3), the 3-round trail cores \((\alpha_1, \alpha_2)\) can be categorized into four groups, i.e., \(|K|K|\), \(|N|K|\), \(|N|N|\) and \(|K|N|\) where \(K\) (resp. \(N\)) represents the state pattern \(\alpha\) is in-Kernel or out-Kernel. Dedicated search strategies are proposed to cover the groups of 3-round trail cores with different characteristics, i.e.,

- the trail cores in the \(|K|K|\) group are separately searched (in Section 4) due to the special characteristic that both \(\alpha_1\) and \(\alpha_2\) are in-Kernel;
- all the other trail cores with at least one out-Kernel \(\alpha\) are searched under the same strategy of different methods. More specifically,
  - the trail cores in \(|N|K|\) and \(|N|N|\) groups are searched (in Section 5) with the same method;
  - while the trail cores in the \(|K|N|\) group are covered (in Section 6) with a slightly different algorithm of similar ideas.

In summary, there are three kinds of 3-round trail cores of which each will follow a specific search algorithm.

### 4 Generating \(|K|K|\) Trail Cores

In this chapter, we introduce the methods used to generate valid 3-round trail cores \((\alpha_1, \alpha_2)\) of \(|K|K|\) feature. With the constraint that \(\alpha_2\) is in-Kernel, a strategy of constructing and filtering target \(\alpha_1\) that ensures both \(\alpha_1\) and \(\alpha_2\) in-Kernel is illustrated. Combining with another constraint on \(\text{Row}_3\), satisfactory \(\alpha_1\) states are collected and extended forward by one round.
4.1 Generating target $\alpha_1$

As $\alpha_1$ is in-Kernel, it has $m$ orbitals distributed at different slices where $m \in \{1, 2, \ldots\}$. At $\beta_1$, the $2m$ bits will be distributed to $k$ slices through $\pi \circ \rho$. As $\pi \circ \rho$ are linear operations, $\alpha_1$ and $\beta_1$ are linearly equivalent. Thus $\alpha_1$ can be represented by a $k$-tuple $\{n_1, n_2, \ldots, n_k\}$ where $n_k$ is the number of active bits of the $i$-th $\beta_1$ slice $\beta_1|_{z_i}$. In this way, in-Kernel $\alpha_1$ are divided into disjoint subsets with $k$-tuple $\{n_1, n_2, \ldots, n_k\}$ that depicts bit distribution pattern of the corresponding $\beta_1$. For example, for $\alpha_1$ of 4 orbitals, there are four $k$-tuples, i.e., \{2, 2, 2, 2\}, \{2, 2, 4\}, \{2, 3, 3\} and \{4, 4\}.

**Definition 1. Valid $\beta_1$ slice.** An active slice $\beta_1|_{z_i}$ that contains at least one $\chi$-compatible in-Kernel slice $\alpha|_{z_i+1}$ through $\chi$ operation is called a valid slice. There are at least 2 active bits in a valid $\beta_1$ slice.

**Definition 2. Candidate $\beta_1$ structure.** Any target $\alpha_1$ must consist of a group of valid $\beta_1$ slices that contain $2m$ bits in total, i.e., $\sum_{i=1}^{k} ||\beta_1|_{z_i}| = 2m$. The combination of the number of bits of valid $\beta_1$ slices $\bigcup_{i=1}^{k} ||\beta_1|_{z_i}|$ is called a candidate $\beta_1$ structure of target $\alpha_1$.

To ensure $\alpha_2$ in-Kernel, all $\beta_1$ slices must be valid (refer to Def. 1). The target $\alpha_1$ must belong to some candidate $\beta_1$ structure (defined to Def. 2). For example, the candidate $\beta_1$ structures of $\alpha_1$ with 3 orbitals is $\{\{3, 3\}, \{2, 2, 2\}\}$. To be specific, for any target $\alpha_1$ of 3 orbitals, through $\pi \circ \rho$, its $\beta_1$ either consists of two valid slices each of 3 bits, or of 3 valid slices each of 2 bits.

Generating target $\alpha_1$ includes constructing all $\alpha_1$ under candidate $\beta_1$ structures and filtering them with constraints imposed by in-Kernel characteristic of both $\alpha_1$ and $\alpha_2$. With an example shown in Fig. 3, the whole process is explained at length. The two bits of an orbital at $\alpha_1$ are denoted by $p_i, q_i$ where $i \in \{1, 2, 3, 4\}$. Note that the notations of bits and orbitals are used just for clear description. The bits of one orbital have no distinction in themselves, so are the four orbitals.

**Generating target $\alpha_1$ under candidate $\beta_1$ structure.** To efficiently construct $\alpha_1$, all slice patterns of valid $\beta_1$ slices with $s$ bits where $s \in \{2, 3, 4, 5, 6\}$, are stored in a lookup table in advance.

1. Through $\pi \circ \rho$, the candidate $\beta_1$ structure $\{3, 3, 2\}$ determines how the 8 bits of the 4 orbitals are organized at $\beta_1$. More specifically, three bits $p_1, p_2$, and $p_3$ from three distinct orbitals are mapped to the same slice $\beta_1|_{z_1}$. One bit $q_4$ of the fourth orbital is mapped to slice $\beta_1|_{z_2}$ with $q_1$ and $q_2$. The left two bits $q_3$ and $p_4$ are mapped to another valid $\beta_1|_{z_3}$ slice.

2. Under the candidate $\beta_1$ structure, for any valid pattern of slice $\beta_1|_{z_1}$ chosen from the lookup table, $p_1', p_2'$ and $p_3'$ are determined. Accordingly, $q_1$, $q_2$ and $q_3$ can be decided from the $\alpha_1$ orbital relations. With a determined $q_3, p_4''$ can be decided through the relations imposed by the validity of the slice $\beta_1|_{z_2}$. Similarly, $q_4$ can be decided with $\alpha_1$ orbital relation. When all the 8 bits of the 4 orbitals are determined, an $\alpha_1$ is constructed.
3. For any constructed $\alpha_1$, filter it with the constraint imposed by the candidate $\beta_1$ structure. To fulfill this $\beta_1$ structure, the $\rho$-offsets of the 8 bits must satisfy

$$z'_1 = z_1 + \text{offset}[p_1]$$
$$= z_2 + \text{offset}[p_2],$$
$$= z_3 + \text{offset}[p_3]$$

$$z'_2 = z_1 + \text{offset}[q_1]$$
$$= z_2 + \text{offset}[q_2],$$
$$= z_4 + \text{offset}[q_4]$$

$$z'_3 = z_3 + \text{offset}[q_3]$$
$$= z_4 + \text{offset}[p_4],$$

which can be further abstracted with

$$v_1 = v_2$$
$$v_3 = v_1 + v_4,$$

where $v_i = (\text{offset}[q_i] - \text{offset}[p_i]) \mod 64$ with $i \in \{1, 2, 3, 4\}$. Here offset[$q_i$] and offset[$p_i$] stand for the $\rho$-offset value of the bit position $q_i$ and $p_i$. With this filter the $\alpha_1$ states that strictly follow the $\beta_1$ structure can be obtained.

4. For any remaining $\alpha_1$ states, filter it with constraints imposed by $\alpha_2$ in-Kernel. In this case, we should further check whether slice $\beta_1|z'_2$ is valid or not.

5. The constructed $\alpha_1$ states which pass the above two filters are the target $\alpha_1$.

Totally, candidate $\beta_1$ structures of $\alpha_1$ with up to 6 orbitals are constructed and filtered. The associated analytical models are listed at Appendix A.

**Remark 1.** Although the methods for constructing target $\alpha_1$ of different candidate $\beta_1$ structures are similar, the above steps are exclusive for $\{3, 3, 2\}$ structure. In practice, different candidate $\beta_1$ structures exhibit slightly different features that inspire dedicated treatments. Especially, even for the same structure, there might be several distinct sub-structures that show diverse bit organizations at $\beta_1$. For example, as shown in Table 1 and Fig. 10, candidate structure $\{3, 3, 2, 2\}$ consists of three different sub-structures. The five $\alpha_1$ orbitals are distributed to two 3-bit $\beta_1$ slices and two 2-bit $\beta_1$ slices through $\pi \circ \rho$-effect. In sub-structure $\{3, 3, 2, 2\}(a)$, the three bits of the second 3-bit $\beta_1$ slice come from the first, second, and third $\alpha_1$ orbital while in sub-structure $\{3, 3, 2, 2\}(b)$, the three bits come from the first, second and forth $\alpha_1$ orbital.

When all target $\alpha_1$ are collected, the satisfactory $\alpha_1$ states are generated by further checking whether the requirement on $\text{Rows}_3$, i.e., $\text{Rows}_3 = \text{Row}_{\alpha_1} + \text{Row}_{\beta_1} + \text{Row}_{\beta_2} \leq \left\lceil \frac{T_3}{2} \right\rceil$ is met or not. Extending the satisfactory $\alpha_1$ forward by one round, all $|K|K|$ trails ($\alpha_1, \alpha_2$) are successfully obtained.

### 4.2 Exhaustive search algorithm of $|K|K|$ trail cores

Rather than enumerate state $\alpha_1$, the techniques used in this work construct in-Kernel $\alpha_1$ from candidate $\beta_1$ structures. The $|K|K|$ Trail Cores Search Method consists of three phases: $\alpha_1$ Construction, Filtering, and Trail Extension as shown in Algorithm 1.
Phase 1. \( \alpha_1 \) Construction. First prepare all the theoretical candidate \( \beta_1 \) structures for \( m \) orbitals. For each candidate \( \beta_1 \) structure, start from the \( \beta_1 \) slice with the maximum number of active bits. To be precise, the \((x, y, z)\) coordinates of active bits from the starting \( \beta_1 \) slice are determined, whose pairing bits of the same orbital in \( \alpha_1 \) are known as well. In the end, coordinates of the \( 2m \) bits at \( \alpha_1 \) are determined either from the valid \( \beta_1 \) slices or from the orbital relations at \( \alpha_1 \).

Phase 2. Filtering. When \( \alpha_1 \) are constructed according to the bit relations determined by the theoretically deduced \( \beta_1 \) structures, further check whether the \( m \) orbitals actually follow the structures or not. Filter \( \alpha_1 \) with constraints that guarantee the \( m \) orbitals are mapped to valid \( \beta_1 \) slices. The constraints can be expressed by 1) relationship equations of the offsets relations \( v_i = (\text{offset}[q_i] - \text{offset}[p_i]) \mod 64 \) with \( i \in \{1, 2, ..., m\} \) of the \( m \) orbitals and 2) direct valid \( \beta_1 \) slice patterns stored in lookup tables. Store the \( \alpha_1 \) that pass the filtering phase.

Phase 3. Trail Extension. Through the filtering phase, all target \( \alpha_1 \) are collected. To generate valid 3-round trail cores of \(|K|K| \) feature, target \( \alpha_1 \) states are extended forward by one round with the constraint on Row3 which is set as \( \lceil \frac{T}{2} \rceil \). The trail extension methods are the same as in [MDV17] whose time complexity is neglected.

---

Algorithm 1: \(|K|K| \) Trail Cores Search Method

**Input:** The candidate \( \beta_1 \) structure  
**Output:** The valid \(|K|K| \) 3-round trail cores

1: while (not last slice pattern of valid \( \beta_1 \) slices) do  
2: (Phase 1 \( \alpha_1 \) Construction:)  
3: Construct \( \beta_1|z'_i \)  
4: Apply \( \rho^{-1} \circ \pi^{-1} \) to get corresponding bits in \( \alpha_1 \)  
5: Construct the related untried orbitals in \( \alpha_1 \)  
6: Using the existing orbitals to construct the other bits in \( \beta_1|z'_{i>1} \)  
7: Get the constructed \( \alpha_1 \) structure  
8: (Phase 2 Filter:)  
9: if \( \alpha_1 \) passes the offsets filter  
10: if \( \beta_1 \) valid and the weight requirement met  
11: satisfactory \( \alpha_1 \) are obtained  
12: end if  
13: else  
14: go to line 4  
15: end if  
16: end while

The time complexity of each candidate \( \beta_1 \) structure is decided by how \( \alpha_1 \) are constructed. Take trail cores with \( \alpha_1 \) of 3 orbitals whose candidate \( \beta_1 \) structures are \( \{2, 2, 2\}, \{3, 3\} \) as an example. For \( \beta_1 \) structure \( \{3, 3\} \), firstly, there are 1000 = \( 2^{9.97} \) valid slices. For one bit of a valid slice, there are 4 candidate orbitals at \( \alpha_1 \). Thus for each valid slice of 3 bits, there are \( 4^3 \) candidate \( \alpha_1 \) patterns. Therefore, the time complexity of \( \{3, 3\} \) is \( 2^{15.97} \).

The theoretical time complexities of all the associated analytical models are listed at Table 1. In practical programming, the searching space can be pruned by checking whether some simple constraints are met or not.

With a total complexity of around \( 2^{92} \), \(|K|K| \) 3-round trail cores of weight up to 53 are exhaustively generated. The time unit is the process of filtering a constructed \( \alpha_1 \) state. All the cases in Table 1 are searched, and the experimental results are displayed in Section 7.


| Table 1: Time complexity of $|K|/K$ trail core search. |
| --- | --- | --- | --- |
| # orbitals in $\alpha_1$ | Candidate $\beta_1$ structure | #time complexity of a case | #time complexity in total |
| 3 orbitals | 1. (3, 3) | 23°.97 | 23°.97 |
| 2 orbitals | 2. (2, 2, 2) | 23°.97 | 23°.97 |
| 4 orbitals | 1. (4, 4) | 23°.97 | 23°.97 |
| 3 orbitals | 2. (3, 3, 2) | 23°.97 | 23°.97 |
| 5 orbitals | 3. (4, 2, 2) | 23°.97 | 23°.97 |
| 4 orbitals | 4. (2, 2, 2, 2) | 23°.97 | 23°.97 |
| 5 orbitals | 1. (4, 4) | 23°.97 | 23°.97 |
| 6 orbitals | 2. (5, 3, 2) | 23°.97 | 23°.97 |
| 5 orbitals | 3. (4, 2, 2) | 23°.97 | 23°.97 |
| 5 orbitals | 4. (2, 2, 2, 2) | 23°.97 | 23°.97 |
| 6 orbitals | 5. a) (3, 3, 2, 2) | 23°.97 | 23°.97 |
| 6 orbitals | 6. b) (3, 3, 2, 2) | 23°.97 | 23°.97 |
| 6 orbitals | 7. c) (3, 3, 2, 2) | 23°.97 | 23°.97 |
| 6 orbitals | 8. (2, 2, 2, 2, 2) | 23°.97 | 23°.97 |

Discussion.

- With this algorithm, all 3-round trail cores of 6 orbitals and part trails of 7 orbitals are obtained. But we cannot cover all cases of 7 orbitals as theoretically there are too many candidate structures and the time complexity increases vigorously.

- The limitation of this exhaustive search algorithm is that it is oriented to orbitals. All 2-round trail cores of $m \leq 6$ orbitals are completely covered. The minimal propagation weight of 3/4/5/6-orbital $|K|/K$ trails are 33/39/48/53 respectively. As $|K|/K$ trails of 7 orbitals cannot be covered and there is no theoretical proof for a satisfactory lower bound, say 55 or 56 as well, we claim exhaustiveness over 53 which is experimentally proved.

- The weights of 7-orbital trails must be greater than 53. Let’s assume the minimal weight of 7-orbital trails is 53. Compared to 6-orbital trails, the 7-th orbital must share rows with two other orbitals at $\alpha_1$. In other words, for the 7-orbital trails of minimal weight 53, there is an $\alpha_1$ slice that has 3 orbitals. Otherwise, the total weight of the 7-orbital trails will exceed 53. On the other hand, the 3 orbitals at the $\alpha_1$ slice will be distributed to 6 slices at $\beta_1$. Thus, the structure of the 7-orbital trails are either $\{2, 2, 2, 2, 4\}$ or $\{2, 2, 2, 2, 3, 3\}$. For either of the structures, the previous 6-orbital structure is $\{2, 2, 2, 2, 2, 2\}$ which implies a weight increase at $\beta_1$. Therefore, compared to the 6-orbital trails, there must be weight increase for the 7-orbital trails.

Based on the above considerations, exhaustive search under threshold weight 53 are claimed. Without caring exhaustive search, 3-round trail cores of 3, 4, 5, 6 orbitals are generated. Those paths can be used in collision attack, preimage attack, and differential distinguisher attack. Particularly, in [GLL+19] one of the newly obtained 3-round trail cores of 5 orbitals are used to successfully identify a 5-round collision on SHA3-256.
5 Generating |N|K and |N|N Trail Cores

To obtain valid 3-round trail cores (α₁, α₂) with at least α₁ out-Kernel, valid 2-round trail cores α₂ that allow the existence of valid 3-round trail cores are collected and extended forward by one round. As in [MDV17], parity-bare states which represent all out-Kernel α₁ under the same parity are enumerated to generate valid 2-round trail cores. A propagation weight T₂ is set to list the parity-bare states of candidate parities.

In this section, we illustrate primary properties used to develop the ideal improvement assumption under which the theoretical representatives of subspaces are generated. Based on those concepts, out-Kernel α₁ are efficiently covered through viability check of subspace \( V_{α₁} \). Valid 3-round trail cores can be obtained through extending all the valid α₁ forward by one round. Ultimately, the strategy of generating valid 3-round trail cores of |N|K and |N|N feature is presented.

5.1 Propagation property of out-Kernel \( α₁ \)

With observations on differential propagation of out-Kernel α₁, β₁ does exhibit properties that can be exploited in the searching phase. Lemma 2 illustrates the elemental properties used to develop exhaustive search algorithms of |N|K and |N|N trails. The analytical model is shown in Eq 4.

\[
α₁ \xrightarrow{λ} β₁ \xrightarrow{λ} α₂ \xrightarrow{λ} β₂
\]  

(4)

Definition 3. Given any difference α (or β), a group is comprised of \( l \) contiguous active slices of which the neighbors are nonactive, denoted by \( α|_g \) or \( β|_g \).

Lemma 2. Given a group pattern \( β₁|_g \), assuming \( χ \)-compatible is not considered, the corresponding groups \( α₂|_g \) that have the minimal propagation weight at \( β₂ \) must be in-Kernel when \( l \leq 6 \).

Proof. When checking out-Kernel state α₁ which is either parity-bare state or state generated by adding orbitals to parity-bare state, through λ operation, the corresponding \( β₁ \) is composed of several group patterns, i.e., \( β₁ := \bigcup_{k=1}^{n} β₁|_g^k \), where \( n \) is the number of groups.

For each \( β₁ \) group \( β₁|_g \), there are a set of \( χ \)-compatible groups \( α₂|_g \) which are either in-Kernel or out-Kernel. Experimentally, the more runs the parity of \( χ \)-compatible groups \( α₂|_g \) have, the more active bits they have which incur heavier propagation weight at \( β₂ \).

For any group \( β₁|_g \), without considering \( χ \)-compatibility, suppose it has in-Kernel \( α₂|_g \) groups. For any \( β₁ \) slice of group \( β₁|_g \), there are a number of in-Kernel α₂ slices. The assumed in-Kernel α₂ slices are prepared following the rule that

- if a \( β₁ \) slice only has one active row, all α₂ slices of one orbital that have one bit in the corresponding active row are counted in;
- if a \( β₁ \) slice has multiple active rows, all the α₂ slices that have the least number of orbitals which are distributed over the corresponding active rows are counted in.

On the other hand, suppose any group \( β₁|_g \) has optimal out-Kernel \( α₂|_g \) groups, i.e., the group patterns under 1-run parity which corresponds to the active \( β₁ \) rows. When all the in-Kernel and out-Kernel \( α₂|_g \) groups are prepared, compare the corresponding propagation weight at \( β₂ \).

It is experimentally proved that there is conditional monotonic relation between the number of active bits \( ||β₂|_g^l ||^3 \) and the number of active rows \( Row|β₂ \) when \( l \leq 6 \). The

\[^3β^l \] is introduced in Eq 1.
propagation weight of the supposed optimal out-Kernel patterns is larger than that of the assumed in-Kernel patterns.

Lemma 3. Without considering $\chi$-compatibility, given any $\beta_1$, the ideal $\alpha_2$ of minimal propagation weight at $\beta_2$ must be in-Kernel.

Proof. The $n$-group $\beta_1$ can be reorganized according to the decrease of the length of each group, i.e., when checking propagation weight $w(\beta_2)$, the group with the largest length is first studied. For each group $\beta_1[g^k]$, there are two kinds of $\chi$-compatible groups $\alpha_2[g^k]$, i.e., in-Kernel and out-Kernel where $k \in \{1, \cdots, 6\}$. Following Lemma 2, for all the $\beta_1$ group patterns, without considering $\chi$-compatibility, prepare the corresponding in-Kernel and out-Kernel $\alpha_2[g^k]$ pattern.

The computation of propagation weight of $w(\beta_2)$ starts from the first group $g^1$. According to Lemma 2 the in-Kernel states will lead to lower weight. When the second group $g^2$ is involved, there might be overlaps on rows with $g^1$ which greatly complicates the whole situation. To address this problem, an experiment of exhaustively adding all assumed in-Kernel and out-Kernel $\alpha_2[g^2]$ to in-Kernel $\alpha_2[g^1]$ and computing the associated propagation weight is conducted. The length of group $g^2$ is at most 5. It shows that the in-Kernel $\alpha_2[g^2]$ always have less propagation weight. When taking the third, fourth and etc groups into consideration, as the group length decreases, in-Kernel $\alpha_2[g^k]$ patterns promise lower total $w(\beta_2)$.

Based on the above analysis, it is concluded that without considering $\chi$-compatibility, given any $\beta_1$, the ideal $\alpha_2$ of minimal propagation weight at $\beta_2$ must be in-Kernel.

Recall that the ultimate goal is to search 3-round trail cores that satisfy $w^{rev}(\alpha_1) + w(\beta_1) + w(\beta_2) \leq T_3$. Given any $\alpha_1$ (or $\beta_1$), the propagation weight $w^{rev}(\alpha_1)$ and $w(\beta_1)$ are direct. Combining the newly introduced property Lemma 3, $w(\beta_2)$ can be predicted under some assumptions which will be explained in the following section.

5.2 The ideal improvement assumption

To search valid 2-round trail cores, it is essential to predict how the out-Kernel $\alpha$ propagates forward (or backward). To clearly explain the strategy and methods, definitions are given in advance.

Definition 4. The derivative subspace of the original 2-round trail cores. Given any out-Kernel state $\alpha$, by adding $n$ orbitals to it, where $n \in \{0, 1, 2, \cdots\}$, a set of out-Kernel states (denoted by $\alpha'$) that have the same parity is generated. Such a set is called the derivative subspace $V_{\alpha}$ of the original out-Kernel state $\alpha$.

Instinctively, if $\alpha_1$ (or $\beta_1$) can predict or evaluate the 3-round propagation weight of all out-Kernel states in its derivative subspace to some extent, valid 2-round trail cores can be efficiently constructed. According to Def. 4, each parity-bare state $\alpha_1$ represents a derivative subspace of 2-round trail cores under specific parities. However it is impossible to decide whether there exist valid 2-round trail cores in the subspace directly from the parity-bare state.

From another perspective, the out-Kernel states $\alpha_1$ are classified with parities $p$ under which each parity-bare state stands for a derivative subspace $V_{\alpha_1}$. The search space of out-Kernel 2-round trail cores is divided into disjoint derivative subspaces of the parity-bare states. A naive method to generate valid 2-round trail cores from each subspace $V_{\alpha_1}$ is to examine all $\alpha'_1$ (where $\alpha'_1 \in V_{\alpha_1}$) with the constraint on propagation weight, i.e., $w^{rev}(\alpha'_1) + w(\beta'_1) \leq T_3 - 2$. The naive plan is obviously impractical.
On the other hand, if there exists some kind of representative 2-round trail core of $V_\alpha$, the derivative subspace as well as the whole search space can be efficiently covered. With Observation 1, neither the original 2-round trail core $\alpha_1$ nor other constructed patterns $\alpha'_1$ can effectively represent $V_\alpha$ in terms of $Row_3$ where $Row_3 = Row_{\alpha_1} + Row_{\beta_1} + Row_{\beta_2}$ is the number of active rows of 3-round trail cores. Based on Observation 2 where the decrement on $Row_3$ is possible by adding orbitals to $\alpha_1$, the theoretical representative $\alpha'_1$ (refer to Def. 6) is proposed to represent the subspace. The ideal improvement assumption of original 2-round trail core $\alpha_1$ based on which the theoretical representative of $V_\alpha$ is defined will be illustrate in this section.

**Observation 1.** Given any derivative subspace $V_{\alpha_1}$, comparing to the original $\alpha_1$, the constructed 2-round trail cores $\alpha'_1$ have a larger number of active rows (i.e., $Row_{\alpha_1} + Row_{\beta_1}$). However, $\alpha'_1$ may have less active rows in $\beta_2$ (i.e., $Row_{\beta_2}$) than $\alpha_1$ due to the round operations. Therefore while $\alpha_1$ is not valid it is possible to exist valid $\alpha'_1$ in subspace $V_{\alpha_2}$.

**Observation 2.** Given a subspace $V_{\alpha_1}$, when $\alpha_1$ is not valid it is possible that there exist some valid $\alpha'_1$, which implies adding orbitals to $\alpha_1$ really decreases $Row_3$ or more precisely $Row_{\beta_2}$. At this case, although adding orbitals to $\alpha_1$ may increase $Row_{\alpha_1}$ or $Row_{\beta_1}$, the decrement on $Row_{\beta_2}$ can compensate the cost so that a valid $\alpha'_1$ is possible.

**Definition 5.** Common and preferred slice. A $\beta_1$ slice $\beta_1|_z$ is common if none of its $\chi$-compatible slices $\alpha_2|_z$ is in-Kernel. Otherwise $\beta_1|_z$ is called a preferred slice.

**The ideal improvement of $\alpha_1$ in terms of $Row_3$.** According to the analysis in Section 5.1, any $\alpha_1$ that has common $\beta_1$ slices (refer to Def. 5) can be improved with a better $Row_{\beta_2}$ by transforming common slices preferred. Here the "improvement" or the later "improve" means the decrement of the number of active rows, or more precisely a lower propagation weight. Each common slice can be improved by adding a specific number of bits to it which introduces the same number of orbitals to $\alpha_1$. For example, a slice of only one bit is a common slice. To improve it we can add one active bit to the bit positions of the same column as the previous active bit. For each of the added orbitals, through the $\pi \circ \rho$-effect, the other bit is distributed either to an active slice or to an empty slice. In the most ideal case, the other bit of the added orbital also improves a common slice to be preferred. Actually, given an original 2-round trail core $\alpha_1$ of $V_{\alpha_1}$, an theoretically optimal state in regarding to $Row_3$ can be deduced with the following three assumptions.

$\beta_1$ is ideally improved to obtain optimal $Row_{\beta_2}$. Assume each common slice $\beta_1|_z$ is transformed preferred by only adding one bit to it. The other bit of the corresponding orbital improves another common slice simultaneously. If there are an odd number of common slices, the impact of the other bit of the left orbital is simply not considered. With this assumption, no new $\beta_1$ slice is introduced while the least number of orbitals are taken to improve $\alpha_1$. When all common slices $\beta_1|_z$ become preferred, the improved $\beta'_1$ has an optimal $Row_{\beta_2}$ according to the conclusion shown in 5.1.

$Row_{\beta_1}$ is counted with the least increase of $\#row$. For each common slice $\beta_1|_z$, there are several kinds of improvements. For example, a bit of an unactive row and a bit of an active row may both improve a common slice. When counting $Row_{\beta_1}$, choose the one with the least increase of $\#row$ among all the possible improvements. With this method, $Row_{\beta_1}$ is minimal under the circumstance of improving common $\beta_1|_z$.

$Row_{\alpha_1}$ is counted with the least increase of $\#row$. When counting the new $Row_{\alpha_1}$ after improving $\beta_1$, choose the added orbital combinations with the least $\#row$ increase.
The optimal state is called the theoretical representative $\alpha'_r$ of $V_{\alpha_1}$ (defined in below 6). As shown in Lemma 4, comparing to all 2-round trail cores in $V_{\alpha_1}$, theoretical representative has the minimal propagation weight $w_{r3}$, where $w_{r3} = w_{\text{rev}}(\alpha'_1) + w(\beta'_1) + w(\beta'_2)$.

**Definition 6. Theoretical representative of derivative subspace.** Under the ideal improvement assumption, given any out-Kernel 2-round trail core $\alpha$, the ideally generated state is called theoretical representative of the derivative subspace $V_{\alpha}$, denoted by $\alpha'_1$.

**Lemma 4.** The theoretical representative $\alpha'_1$ of a derivative subspace $V_{\alpha_1}$ represents all the 2-round trail cores, i.e, both $\alpha_1$ and $\alpha'_1 \in V_{\alpha_1}$, in terms of Row3.

**Proof.** When $\alpha_1$ has no common slices $\beta_1|z$, the theoretical representative $\alpha'_1$ is identical to $\alpha_1$. When $\alpha_1$ has at least one common slice, $\alpha'_1$ has the minimal Row3 according to the ideal improvement assumption. Besides, as $\alpha'_1$ is ideally improved from $\alpha_1$ it naturally has a lower Row3 than the other $\alpha'_1$ except those $\alpha'_1$ that follow the ideal improvement. Therefore, the theoretical representative always provide the possibly optimal Row3 for all 2-round trail cores in $V_{\alpha_1}$. □

**Remark 2.** The theoretical representatives of subspaces do not necessarily exist in practice, i.e., $\alpha'_1 \notin V_{\alpha_1}$ is acceptable, as the ideal assumption may not comply with the actual $\pi \circ \rho$-effect.

### 5.3 The 3-round trail core search

The theoretical representative provides a criteria to evaluate whether there are valid 2-round trail cores in the subspace. As shown in Lemma 4, if the theoretical representative of $V_{\alpha_1}$ is valid, there possibly are valid 2-round trail cores in the subspace. On the other hand, if the theoretical representative is not valid, there definitely are not any valid 2-round trail cores in the subspace. In this section, we explain at length how to generate valid 2-round trail cores with the ideal improvement assumption. We first illustrate the viability check of a derivative subspace $V_{\alpha_1}$.

#### 5.3.1 Generating viable 2-round trail cores

Given a subspace $V_{\alpha_1}$, the process of constructing its theoretical representative with the ideal improvement assumption and checking the validity of the theoretical representative is called viability check. A subspace that passes the viability check is viable (defined below). For any viable subspaces, the following problem is to precisely identify the valid 2-round trail cores. In the following sections, we use viable subspace $V_{\alpha}$ and viable state $\alpha$ (or 2-round trail cores) indiscriminately. The viability check of a subspace $V_{\alpha}$ equals to the viability check of a state $\alpha$.

**Definition 7. Viable 2-round trail cores.** The 2-round trail cores $\alpha$ of which the theoretical representatives $\alpha'_1$ of its derivative subspace can pass the viability check, i.e., $w_{r3}' \leq T_3$, are called viable. The derivative subspace of a viable 2-round trail core is also called a viable subspace.

Viability check only shows whether it is possible that there exist valid 2-round trail cores in the subspace. Actually, the viable subspaces do not necessarily promise valid 2-round trail cores. With Lemma 5, the valid 2-round trail cores are obtained by checking the validity of viable trail cores.

**Lemma 5.** The valid 2-round trail cores must be in the space of viable 2-round trail cores.
Proof. With the previous illustration, there are definitely not any valid 2-round trail cores in a non-viable subspace. In a viable subspace, following the definition of viable 2-round trail core (Def. 7), the valid 2-round trail cores must also satisfy the constraints of the possibly valid trail cores. Therefore, the space of viable states contains all the valid 2-round trail cores.

The strategy used in locating all viable 2-round trail cores in a viable subspace is introduced in the following paragraph.

**Locate all viable 2-round trail cores of a viable subspace** $V_{\alpha_1}$. Given a viable subspace $V_{\alpha_1}$, the 2-round trail cores in it contain the original $\alpha_1$ as well as a great number of $\alpha'_1$ which are constructed by adding orbitals to $\alpha_1$. To collect all viable 2-round trail cores of a viable derivative subspace, a strategy of adding one orbital to viable original states each time is widely used.

- The original 2-round trail core $\alpha_1$ are viable which could possibly be valid. Viable 2-round trail cores are stored for further trail extension.

- As for the 2-round trail cores $\alpha'_1$ in $V_{\alpha_1}$, rather than enumerate all $\alpha'_1$, only $\alpha'_1$ with one orbital added to $\alpha_1$ are checked at first. If the sub-subspace $V_{\alpha'_1}$ is viable, the corresponding $\alpha'_1$ is a viable 2-round trail core. Otherwise, the sub-subspace can be safely discarded. Thus $\alpha'_1$ with only one orbital added to $\alpha_1$ is covered. Similarly, to cover $\alpha''_1$ with two orbitals added to $\alpha_1$, add one orbital to the previously collected viable $\alpha'_1$ and check the viability of the new derivative subspaces $V_{\alpha''_1}$. In this way, every time the viable 2-round trail cores with $n$ orbitals added to $\alpha_1$ are collected, add one orbital to them and check the viability of the newly generated subspaces. This process will terminate when none of states generated by adding one orbital to the newly obtained viable 2-round trail cores can pass the viability check.

The parity-bare states of all candidate parities are enumerated to operate the viability check after which a set of viable 2-round trail cores are collected. With the method presented above, we generate the viable 2-round trail cores in each of of the viable derivative subspaces. In the end, all viable 2-round trail cores of the search space are collected.

![Diagram](image)

**Figure 4:** The process of 3-round trail core search for trails with $|N|K$ and $|N|N$ feature.

5.3.2 Search strategy of 3-round trail cores

Up to now, the ideal improvement assumption and the theoretical representatives of derivative subspaces are introduced with which the search space of 2-round trail cores can
be efficiently covered and filtered out. All viable 2-round trail cores are collected through the viability check of subspaces. At this part, we give a comprehensive description on how to generate valid 3-round trail cores with the techniques explained in the previous sections.

1. Prepare parities \( p \) of 1-run, 2-run and even 3-run, and enumerate the associated parity-bare states. With rough estimation on the propagation weight of \( \alpha_1 \) and \( \beta_1 \), only states under parities of at most 3-runs can meet the threshold weight.

2. Given any parity-bare state \( \alpha_1 \), check the viability of its derivative subspace \( V_{\alpha_1} \).

3. Given any viable subspace \( V_{\alpha_1} \), collect all the viable 2-round trail cores in the subspace.

4. Extend all viable 2-round trail cores forward by one round with threshold weight \( T_3 \) to obtain all valid 3-round trail cores.

**The exhaustiveness of the search space.** To fully cover the search space of \( |N|K|, |N|N \) case and \( |K|N \) case, i.e., the 2-round trail cores, we enumerate the parity-bare states under all candidate parities as in [MDV17]. To be more specific, candidate parities of 1-run, 2-run, and 3-run are prepared.

- For 1-run parities, the length is up to 7. The time complexity of checking the viability of the parity-bare states is around \( 5 \times (2^4)^9 = 2^{38.3} \). It shows that there are no viable states for runs of length greater than 5.

- For 2-run case, parities of lengths (1,1) and (1,2) are listed where (1,1) (or (1,2)) represents the lengths of the two runs are both 1 (or 1 and 2). The time complexity of checking the viability of the parity-bare states are around \( 2^{10.6} \times (2^4)^6 = 2^{34.6} \) and \( 2^{11.6} \times (2^4)^7 = 2^{39.6} \) respectively.

  For 2-run parities of lengths (2,3), (2,2) and (1,3), we only prepare those with affected odd columns (as shown in the first example of Fig 5) of which the time complexity are less than \( 2^{35} \). In fact, for most of the states under those parities, especially for those of more than 30 active bits in \( \alpha \) and \( \beta_0 \), the weights of the 2-round trail cores already exceed the threshold weight \( T_3 \). Thus we do not need to conduct the viability check for those states.

  And there are no viable states under (2,3), (2,2) and (1,3) parities.

![Figure 5: Examples of 2-run parities with and without affected odd columns](image)

- For 3-run parities of length (1,1,1), only those with affected odd columns are prepared. The time complexity of checking the viability of the parity-bare states under those
parities are around $2^{36}$. The weights of all 2-round trail cores under those parities are greater than the threshold weight $T_3$ implying there are no viable states under 3-run parities.

With the increase on lengths of the runs, the number of active bits in $\alpha$ and $\beta^0$ increases accordingly. To be specific, wherever the positions of affected columns, the 2-run (or 3-run) parities have $2 \times 2 \times 5 + 2 \times \sum_{i=1}^{k} n_i$ (or $3 \times 2 \times 5 + 2 \times \sum_{i=1}^{k} n_i$) active bits in $\alpha$ and $\beta^0$ where $k$ is the number of odd columns minus the number of affected odd columns (i.e., the number of unaffected odd columns), and $n_i$ is the number of active bits of the $i$-th unaffected odd column. While a greater number of active bits does not always mean a heavier weight, it is so when the number of active bits are small enough.

Although for different parities the active bits are placed at different positions, the increase of active bits at $\alpha$ and $\beta^0$ will not decrease the weights under this context. Thus with the experimental verification, the longer the lengths of runs, the heavier the weights under those parities. In total, we prepare as many candidate parities as possible. When there is no viable states under a parity pattern, e.g., the $(1,1,1)$ 3-run parities, we just quit and do not check other cases under this category that have more active bits in $\alpha$ and $\beta^0$. In this way, all parities that may contain viable states are covered.

Figure 4 shows the complete process. The enumeration of parity-bare states and trail extension methods are directly taken from [MDV17]. In practice, 3-round trail cores with $T_3 = 53$ are generated with a time complexity of around $2^{40}$ whose time unit is the viability check of a 2-round trail core. The search results are summarized in Section 7.

5.4 The list of assumptions used to exhaustively cover the search space

The exhaustive search algorithms of 3-round trail cores with $|N|N|$, $|N|K|$ and $|K|N|$ features are based on several assumptions, i.e., the ideal improvement assumption and the assumptions used to fully cover the search space. To clarify the logic behind, we summarize them in this section.

1. The assumptions used to fully cover the search space

According to the search strategy, the search space is divided into subspaces derived from parity-bare states which are generated under parities. Parities are organized by runs, for example, 1-run, 2-run and 3-run parities. As discussed in Section 5.3.2, the number of active bits of $\alpha$ and $\beta^0$ of parity-bare states is $m \times 2 \times 5 + 2 \times \sum_{i=1}^{k} n_i$, where $m$ is the number of runs of parities, $k$ is the number of unaffected odd columns, and $n_i$ is the number of active bits of the $i$-th unaffected odd column. To enumerate all candidate parities, we have the following assumptions.

- The more runs of a parity, the larger the number of active bits of $\alpha$ and $\beta^0$ of parity-bare states under the parity.
- With the increase of length and number of runs, the number of active bits increases accordingly. Under this context, it is experimentally verified that the increase of the number of active bits will not decrease the propagation weight $w^{rev}(\alpha) + w(\beta)$.
- Therefore, when enumerating parities organized by runs, if all the parity-bare states of one parity are not viable, we quit checking other parities of longer length or more runs.

Starting from 1-run parity, we enumerate all possible parities with these assumptions.

2. The ideal improvement assumption

As described in Section 5.2, the ideal improvement assumption is used to determine whether a subspace is viable or not. In other words, with this assumption we decide whether it is possible to exist valid 3-round
trail cores in a subspace. To that end, an ideal representative of the subspace in terms of 3-round propagation weight is built upon the assumption that all the most ideal cases happen at the same time when improving the original 2-round trail core.

6 Generating |K|N| Trail Cores

|K|N| trail core search shares the same strategy with |N|K| and |N|N| trail core search. Concepts and definitions illustrated in Section 5 are equally applied to |K|N| trail core search. In searching 3-round trail cores, the principal technique is to compute Row\(\alpha\) of theoretical representatives of derivative subspaces \(V_{\alpha_2}\), i.e., the viability check. In this section, the ideal improvement assumption under the |K|N| situation is introduced to construct theoretical representatives. With the dedicated ideal compensating assumption, the algorithm of counting Row\(\alpha\) is explained in detail. Finally, the general process of generating |K|N| 3-round trail cores with threshold propagation weight \(T_3\) is provided.

6.1 The ideal improvement assumption in compensating \(\alpha_2\)

3-round trail cores with |K|N| feature require \(\alpha_1\) in-Kernel. With the analytic model shown in Eq. 5 where \(\lambda^{-1} = \rho^{-1} \circ \pi^{-1}\), the theoretical representatives of subspaces \(V_{\alpha_2}\) are generated and checked. In each subspace \(V_{\alpha_2}\), suppose there are at least one 2-round trail cores that have in-Kernel \(\alpha_1\), i.e., either \(\alpha_2\) or states generated by adding orbitals to \(\alpha_2\) have in-Kernel \(\alpha_1\). Basically, it is assumed that the least number of orbitals are added to \(\alpha_2\) to ensure theoretical representative with in-Kernel \(\alpha_1\). Additionally, the added orbitals also impose the least number of row increase on \(Row_{\alpha_2} + Row_{\beta_2}\). Some definitions are given before introducing the ideal assumptions.

\[
\alpha_1 \xleftarrow{\lambda^{-1}} \beta_1 \xleftarrow{\pi^{-1}} \alpha_2 \xrightarrow{\lambda} \beta_2
\]  

Definition 8. Orbital and Non-Orbital bit and row of difference \(\beta_1\). Given a difference \(\beta_1\), among all the active rows, through \(\rho^{-1} \circ \pi^{-1}\), if one bit of an active row \(\beta_1|_r\) is distributed to the same column at \(\alpha_1\) with another bit from another active row \(\beta_1|_p\), the bit is called an orbital bit. If an active row \(\beta_1|_r\) contains at least one orbital bit, it is called an orbital row. Otherwise, it is a non-orbital row.

Definition 9. Sharing bit and Sharing row of difference \(\beta_1\). Given a difference \(\beta_1\), among all the active rows, through \(\rho^{-1} \circ \pi^{-1}\), if one bit of an active row \(\beta_1|_r\) is distributed to the same row at \(\alpha_1\) with another bit from another active row \(\beta_1|_p\), the bit is called a sharing bit. If an active row \(\beta_1|_r\) contains at least one sharing bit, it is called a sharing row.

Definition 10. Compensating row of a non-orbital row. For any non-orbital \(\beta_1\) row, there are around \(5 \times 4 = 20\) \(\alpha_2\) rows that can form \(\alpha_1\) orbitals with it. Such \(\alpha_2\) rows are called the compensating rows of the non-orbital row.

The process of adding orbitals to \(\alpha_2\) to make the related non-orbital \(\beta_1\) rows orbital is called compensating the non-orbital rows. Similar to the concept of improving a common slice in the last chapter, here compensating a row means decreasing the total number of active rows \(Row_{\beta_3}\).

\(^4\)The theoretical representative \(\alpha_2^\ast\) of \(V_{\alpha_2}\) promises the minimal propagation weight \(w_3^\ast\) of the subspace, where \(w_3^\ast = w_3^{rev}(\alpha_1^\ast) + w(\beta_1^\ast) + w(\beta_2^\ast).\)
The ideal compensating assumption. Given an out-Kernel state $\alpha_2$, to efficiently determine whether there are valid 2-round trail cores in its derivative subspace, the validity of the theoretical representative ought to be checked. The assumptions used to compute $\text{Row}_{\alpha_3}$ of the theoretical representative of $V_{\alpha_2}$ are listed below.

Superset of $\beta_1$ active row. To ensure $\alpha_1$ in-Kernel with the minimal cost of orbitals added to $\alpha_2$, without considering the compatibility of $\chi^{-1}$, assume all the $(2^3 - 1)$ nonzero $\beta_2$ row value are valid. With this assumption, active rows at $\beta_1$ are classified into two categories, i.e., orbital and non-orbital rows (refer to Def 8). The orbital rows naturally form orbitals at $\alpha_1$ through $\rho^{-1} \circ \pi^{-1}$, while the non-orbital rows need to be compensated to be orbital rows by adding orbitals to $\alpha_2$. The $\alpha_1$ orbitals that correspond to orbital $\beta_1$ rows through $\pi \circ \rho$ are called natural orbitals as there is no need to add extra orbitals to $\alpha_2$ to make such $\alpha_1$ orbitals possible. Comparatively, the $\alpha_1$ orbitals that corresponds to non-orbital $\beta_1$ rows are called compensating orbitals because such $\beta_1$ rows need to be compensated to be orbital at $\alpha_1$ (through adding orbitals to $\alpha_2$).

For any active rows of $\alpha_2$, as the superset of the corresponding $\beta_1$ row is considered, the $\chi^{-1}$ layer can be bypassed.

The optimal $\text{Row}_{\alpha_1}$. Practically, $\alpha_1$ of the theoretical representative is comprised of a set of natural orbitals and compensating orbitals. When selecting the combination of $\alpha_1$ orbitals from both orbital and non-orbital $\beta_1$ rows, it is possible that the involved $\beta_1$ rows are also sharing rows (refer to Def 9). Therefore, there are cases that either the natural orbitals or the compensating orbitals share some $\alpha_1$ rows. In that case the $\alpha_1$ orbitals of sharing rows are always selected. Under the above assumption, the optimal $\text{Row}_{\alpha_1}$ of the theoretical representative can be obtained.

The least increase on the number of rows to $\text{Row}_{\alpha_2} + \text{Row}_{\beta_2}$. To compensate the non-orbital $\beta_1$ rows, orbitals are added to $\alpha_2$. At this phase, whatever $\alpha_2$ orbitals are chosen, suppose the optimal $\alpha_1$ generated in the last step always holds. The orbitals added to $\alpha_2$ must at first compensate all the non-orbital $\beta_1$ rows. In other words, the added $\alpha_2$ orbitals at least lie in one compensating row (refer to Def 10). Then the finally chosen orbital combination must have the least influence on $\alpha_2$ and $\beta_2$ in regard of the number of active rows. The number of active rows of $\alpha_2$ and $\beta_2$ of the theoretical representative is computed with this assumption.

The ideal compensating assumption described above ensures a theoretical representative of subspace $V_{\alpha_2}$ with $|K|N$ feature. To check the viability of $V_{\alpha_2}$, the ideal $\text{Row}_{\alpha_3}$ of the theoretical representative is computed with the optimal $\text{Row}_{\alpha_1}$ and $\text{Row}_{\alpha_2} + \text{Row}_{\beta_2}$. We introduce the detailed algorithm used in computing $\text{Row}_{\alpha_3}$ of theoretical representative under the ideal improvement assumption in the next section.

6.2 Counting $\text{Row}_{\alpha_3}$ of theoretical representatives

$\text{Row}_{\alpha_3}$ of the theoretical representative stands for all states in subspace $V_{\alpha_2}$ in regard to the constraints imposed by $|K|N$ feature and threshold propagation weight $T_3$. If $\text{Row}_{\alpha_3} > \left\lceil \frac{T_3}{2} \right\rceil$, there is no valid 2-round trail cores in $V_{\alpha_2}$. The subspace can be safely discarded. The search space can be pruned effectively in this way.

With the ideal assumption, $\text{Row}_{\alpha_3}$ is the sum of $\text{Row}_{\alpha_1}$ and $\text{Row}_{\alpha_2} + \text{Row}_{\beta_2}$. The exact algorithm is listed below.

Counting $\text{Row}_{\alpha_1}$. According to the previous analysis, there are two kinds of orbitals in $\alpha_1$, i.e., the natural orbitals that come from the active $\beta_1$ rows of original $\alpha_2$, and
the compensating orbitals that are generated through adding orbitals to \( \alpha_2 \). In counting \( \text{Row}_{\alpha_1} \), natural orbitals and compensating orbitals are considered separately.

- **Natural orbitals.** The natural \( \alpha_1 \) orbitals that correspond to \( \beta_1 \) orbital rows which have only one orbital bit must be counted in computing \( \text{Row}_{\alpha_1} \). Some orbital \( \beta_1 \) rows contain multiple orbital bits which implies they correspond to multiple \( \alpha_1 \) orbitals. In that case, if any of the \( \alpha_1 \) orbitals has been counted, the other \( \alpha_1 \) orbitals of the \( \beta_1 \) rows can be discarded. Otherwise, if none of the associating \( \alpha_1 \) orbitals have been counted, choose the one with the least increase on \( \text{Row}_{\alpha_1} \). With this rule, the theoretically optimal \#row of natural orbitals is obtained.

- **Compensating orbitals.** The compensating \( \alpha_1 \) orbitals of a non-orbital \( \beta_1 \) row might share \( \alpha_1 \) rows with natural orbitals or with other compensating orbitals. In that case, always choose the compensating orbital that shares rows with natural orbitals over the one that shares rows with other compensating orbitals. If the non-orbital row has no sharing row, i.e., its compensating orbitals neither share rows with natural orbitals nor with other compensating orbitals, simply increase \#row with 2.

**Counting \( \text{Row}_{\alpha_2} + \text{Row}_{\beta_2} \) with added orbitals.** As explained earlier, when considering \( \text{Row}_{\alpha_2} + \text{Row}_{\beta_2} \) suppose the theoretically optimal \( \text{Row}_{\alpha_1} \) always holds whatever the orbital combinations are added to \( \alpha_2 \). At \( \alpha_2 \), orbitals with compensating rows are added to \( \alpha_2 \) to make the non-orbital \( \beta_1 \) rows orbital. The combinations of added \( \alpha_2 \) orbitals with the least increase on \#row to \( \text{Row}_{\alpha_2} \) and \( \text{Row}_{\beta_2} \) are chosen.

In compensating the non-orbital \( \beta_1 \) rows, there are theoretical cases that one \( \alpha_2 \) compensating row compensates multiple non-orbital \( \beta_1 \) rows. There are also theoretical cases that one \( \alpha_2 \) slice contains several compensating rows. In general, there are some situations to be considered when choosing the added orbitals to \( \alpha_2 \).

- If there are \( \alpha_2 \) slices in which the number of compensating \( \alpha_2 \) rows is less than that of the compensated non-orbital \( \beta_1 \) rows, such compensating \( \alpha_2 \) orbitals always have the least increase on the number of active rows to \( \text{Row}_{\alpha_2} + \text{Row}_{\beta_2} \). If such best case exists, the related \( \alpha_2 \) rows will be chosen over all the other compensating rows.

- There are cases that an \( \alpha_2 \) slice contains multiple compensating rows each of which compensates a different non-orbital \( \beta_1 \) row. If the corresponding non-orbital \( \beta_1 \) rows can not be compensated in the best situation, it should be compensated under this circumstance.

- In the worst case, there is only one compensating \( \alpha_2 \) row that can only compensate one non-orbital \( \beta_1 \) row in a \( \alpha_2 \) slice. If a non-orbital \( \beta_1 \) row cannot be compensated with the last two cases, it has to be compensated in this way. The corresponding \( \alpha_2 \) orbitals of the compensating \( \alpha_2 \) rows will be counted in computing \( \text{Row}_{\alpha_2} + \text{Row}_{\beta_2} \).

At last, compute the \#row of \( \alpha_2 \) and \( \beta_2 \) with the orbitals of compensating rows added to \( \alpha_2 \) by the above rule.

The key technique in our search strategy is to determine \( \text{Row}_3 \) of the theoretical representative of \( V_{\alpha_2} \). With the above algorithm of computing \( \text{Row}_{\alpha_1} \) and \( \text{Row}_{\alpha_2} + \text{Row}_{\beta_2} \), the complete process of exhaustively searching 3-round trail cores of \(|K|N|\) characteristic is illustrated in next part.

### 6.3 Search strategy of \(|K|N|\) 3-round trail cores

Following the same idea in Section 5, the search space of all out-Kernel \( \alpha_2 \) is divided into a series of subspaces \( V_{\alpha_2} \) where the original states \( \alpha_2 \) are the parity-bare states under
candidate parities. With ideal improvement assumption, viability check is conducted to the subspaces to collect viable \( \alpha_2 \). Ultimately all viable \( \alpha_2 \) are collected and extended backward by one round to generate valid 3-round trail cores of \(|K|N|\) feature. Depicted in Fig 6, the complete search steps are introduced in the following paragraphs.

![Parity-bare States Diagram](image)

**Figure 6**: The process of 3-round trail core search for trails with \(|K|N|\) feature.

1. To cover the space of out-Kernel \( \alpha_2 \), prepare parities \( p \) of 1-run, 2-run and 3-run. Enumerate parity-bare states under candidate parity.

2. Conduct viability check to subspaces \( V_{\alpha_2} \) where \( \alpha_2 \) are parity-bare states. The viability check is to check the validity of the theoretical representative of the derivative subspace by checking whether \( \text{Row}_3 \leq \lceil \frac{T_3}{2} \rceil \) is met or not.

3. For each viable derivative subspace \( V_{\alpha_2} \), by adding one orbital each time and checking the viability accordingly, collect all the viable 2-round trail cores in the subspace. The method is the same as in Section 5, i.e.,

   (a) For all viable \( \alpha_2 \), add one orbital to it to construct new 2-round trail cores \( \alpha'_2 \). Conduct viability check to new generated \( V_{\alpha'_2} \) and collect the viable \( \alpha'_2 \) accordingly.

   (b) Repeat the above process on the freshly viable \( \alpha'_2 \) until all the new 2-round trail cores that are constructed from adding one orbital to the current viable 2-round trail core cannot pass the viability check.

4. Extend all viable \( \alpha_2 \) backward by one round to generate valid 3-round trail cores with \(|K|N|\) feature.

In our experiments, 3-round trail cores of weight up to 53 are exhaustively searched with a time complexity of \( 2^{45} \). Similar to \(|N|K|\) and \(|N|N|\) case, parities of at most 3-runs are prepared after rough estimation on propagation weight. The search results are reported in Section 7.

### 7 Search Result And Bounds for Trails Covering More Rounds

#### 7.1 Summary of the search result

According to the methods and algorithms introduced in Section 4, 5 and 6, 3-round trail cores with threshold propagation weight \( T_3 \) are exhaustively searched. More specifically,
Table 2: Summarized result of 3-round trail cores search.

| | $|K|K|$ | $|N|K|$ | $|N|N|$ | $|K|N|$ |
|---|---|---|---|---|
| $T_3$ | 53 | 53 | 53 | 53 |
| Time Complexity | $2^{42}$ | $2^{40}$ | $2^{40}$ | $2^{45}$ |
| Minimal Weight | 35 | 46 | 48 | 32 |

$|K|K|$ trail cores, $|N|K|$ and $|N|N|$ trail cores, and $|K|N|$ trail cores of propagation weight up to 53 are generated with time complexity $2^{42}$, $2^{40}$, and $2^{45}$ respectively. The results of exhaustive 3-round trail core search are summarized in Table 2.

The propagation weight distribution of 3-round trail cores under distinct categories are listed in Fig 7. There is a trend that when $\alpha_1$ is in-Kernel it is more likely to propagate to valid 3-round trail cores.

7.2 Bounds for 4/5/6-round trails

With all 3-round trail cores of threshold propagation weight $T_3 = 53$ at hand, we can update the lower bound on propagation weight of 4/5/6-round trails. Using the same idea as in [MDV17], lower bound for 4/5/6-round trails can be updated to 56, 58 and 108 respectively compared to the previous 48, 50 and 92.

8 Conclusion

Based on whether $\alpha_1$ and $\alpha_2$ are in-Kernel, 3-round trail cores are classified into four categories, i.e., $|K|K|$, $|K|N|$, $|N|K|$ and $|N|N|$ trail cores. In this work, we study the propagation properties of different kinds of 3-round trail cores and deduce dedicated exhaustive search algorithm for each of them. For $|K|K|$ trail cores, candidate in-Kernel
α₁ are constructed and filtered based on β₁ structures. For trail cores with at least one out-Kernel α₁, idealized differentials of subspaces are constructed to effectively prune the search space.

In conclusion, 3-round trail cores with threshold weight 53 are obtained. With exhaustively searched 3-round trail cores, lower bound on propagation weight of 4/5/6-round trails can be improved to 56/58/108 accordingly.

Acknowledgments

The first two authors are supported by the National Key Research and Development Program of China under Grant 2017YFB0802704, and the National Natural Science Foundation of China under Grant 61972249. The last author is supported by the National Research Foundation, Prime Minister’s Office, Singapore, under its Strategic Capability Research Centres Funding Initiative, Nanyang Technological University under grant M4082123, and Singapore’s Ministry of Education under grants M4012049, M4012153, and M4020466. We especially thank Prof. Guo Jian of Nanyang Technological University who offers us advice and guidance for this work. We thank Gilles Van Assche for helping us improve this paper. We also thank all the fellows and peers who give us advice, support and encouragement.

References


[DDS14a] Itai Dinur, Orr Dunkelman, and Adi Shamir. Collision Attacks on Up to 5 Rounds of SHA-3 Using Generalized Internal Differentials. In Shiho Moriai,


A Analytic Model In $|K|K|$ Trail Search

The candidate $\beta_1$ structures of 3, 4, and 5 orbitals are listed in this section. The $\beta_1$ structures of 6 and even 7 orbitals can be deduced in the same way. As there are too many structures, we omit them.

The $n$ orbitals in $\alpha_1$ are denoted as $p_1, q_1, \cdots, p_n, q_n$ with coordinates $(x_{p_i}, y_{p_i}, z_{p_i})$ and $(x_{q_i}, y_{q_i}, z_{q_i})$ respectively. Note it is possible that two orbitals are in the same slice $z$ in $a$, i.e., there may exist $z_i$ and $z_j$ so that $z_i = z_j$ ($i, j \in \{1, 2, \cdots, n\}$). The $2n$ bits are transformed to bits $p'_1, q'_1, \cdots, p'_n, q'_n$ at $\beta'_1$ through $\rho$, and to bits $p''_1, q''_1, \cdots, p''_n, q''_n$ at $\beta_1$ through $\pi$.

\[
\begin{array}{c|c|c}
\alpha_1 & \beta'_1 & \beta_1 \\
\hline
p_1 & p_2 & p_3 \\
q_1 & q_2 & q_3 \\
z_1 & z_2 & z_3 \\
\hline
\end{array}
\]

(a) A candidate $\beta_1$ structure $\{3, 3\}$.

\[
\begin{array}{c|c|c}
\alpha_1 & \beta'_1 & \beta_1 \\
\hline
p_1 & p_2 & p_3 \\
q_1 & q_2 & q_3 \\
z_1 & z_2 & z_3 \\
\hline
\end{array}
\]

(b) A candidate $\beta_1$ structure $\{2, 2, 2\}$.

Figure 8: Candidate $\beta_1$ structures of 3 orbitals.
Figure 9: candidate $\beta_1$ structures of 4 orbitals
Figure 10: Candidate $\beta_1$ structure of 5 orbitals.
(a) A candidate $\beta_1$ structure $\{4, 2, 2, 2\}$.

(b) A candidate $\beta_1$ structure $\{2, 2, 2, 2, 2\}$.

Figure 11: Candidate $\beta_1$ structure of 5 orbitals.