Refining Probabilistic-Linearization TIDA for 5-Round SHA3-384 Collision Attacks

Luhan Yan[1], Zhenzhen Bao[1,2,3] ✉, and Huina Li[1] ✉

[1] Institute for Network Sciences and Cyberspace, Tsinghua University, Beijing, China
ylh23@mails.tsinghua.edu.cn, {zzbao,lihuina}@mail.tsinghua.edu.cn
[2] Zhongguancun Laboratory, Beijing, China
[3] State Key Laboratory of Cryptography and Digital Economy Security, Tsinghua University, Beijing, 100084, China

Abstract. Since the SHA-3 family was standardized by NIST in 2015, its collision resistance has been extensively studied. The previous best-known collision attack on 5-round SHA3-384 is based on internal differentials and the probabilistic-linearization variant of two-block Target Internal Differential Algorithm (TIDA). Its connector replaces deterministic affine restrictions that guarantee S-box differential validity by higher-dimensional affine relaxations, reducing the number of linear constraints and preserving more degrees of freedom. The price is that solutions of the linearized system are only probabilistically valid. In particular, once the first message block fixes the inner part variables, the remaining affine solution space cannot be treated as a set of independent trials; only part of its freedom effectively contributes to the connector probability. This paper refines the probabilistic-linearization framework in two ways. We first propose SA-PIDS, a simulated-annealing-based search for affine-subspace assignments, improving the trade-off among capacity constraints, the product-density estimate, and the remaining solution-space dimension. We then give a structural evaluation of the actual connector probability, explaining how the effective remaining freedom in the solution space after fixing the first block could be used and counted. Using the same target internal differential characteristic as the previous 5-round SHA3-384 attack, our refinements reduce the theoretical complexity from $2^{170.73}$ to $2^{164.11}$.

Keywords: SHA-3 · Collision Attacks · Internal Differentials · TIDA

1 Introduction

SHA-3 is the latest NIST-standardized Secure Hash Algorithm family specified in FIPS 202 [14] in 2015 after the SHA-3 competition, whose winner was Keccak, designed by Bertoni, Daemen, Peeters and Van Assche [2]. The standard specifies four fixed-output hash functions, SHA3-224, SHA3-256, SHA3-384, and SHA3-512, and two extendable-output functions, SHAKE128 and SHAKE256. All these standardized instances are based on the Sponge construction instantiated with the Keccak-f[1600] permutation with 24 rounds. As a standardized hash

and extendable-output function family, SHA-3 is an important target for security analysis. In particular, collision attacks on reduced-round KECCAK/SHA-3 are a standard way to evaluate its security margin.

A central difficulty in attacking a SHA-3 hash function, as opposed to the underlying permutation alone, is the input constraint imposed by the Sponge construction. In a SHA3-d instance, the 1600-bit state is divided into an outer part and an inner part. Message blocks are absorbed only into the outer part of the state, while the inner part is not directly controlled by the attacker. Thus, a differential characteristic of the KECCAK permutation does not immediately yield a collision attack on the hash function. Before KECCAK became the SHA-3 standard, practical collision attacks on the full hash function were known only for a very small number of rounds. The attacks of Naya-Plasencia *et al.* [10] used carefully chosen low-weight differential characteristics starting from the initial state, and reached two rounds for collisions and three rounds for near-collisions. Extending these attacks was difficult because, although a low-weight characteristic can be extended backward by one round while preserving valid probability, the inverse linear layer of KECCAK diffuses the resulting starting difference into a high-weight difference that is hard to realize from valid initial hash states.

Dinur *et al.* [3] overcame this obstacle by introducing the Target Difference Algorithm (TDA). The key insight is to avoid treating the first round as another probabilistic differential transition. They start from a low-Hamming-weight, high-probability differential characteristic for the later rounds and extend it one round backward to obtain a prescribed target difference. Due to the inverse diffusion of the KECCAK linear layer, this target difference has high weight and is unlikely to be reached by random message pairs. TDA instead constructs a one-round algebraic connector, namely a procedure that finds message pairs whose difference after the first round equals the prescribed target difference. This connector exploits the algebraic simplicity of one KECCAK round, in particular the quadratic χ layer, which is a parallel application of 5-bit S-boxes, and turns the unlikely matching event into a first-round constraint-solving problem. More precisely, it is split into a difference phase and a value phase. In the difference phase, the possible input differences of active S-boxes are restricted to low-dimensional affine subspaces compatible with the prescribed output differences; after being pulled back through the linear layer, these restrictions become linear equations on the initial difference. In the value phase, once the relevant S-box input and output differences are fixed, the corresponding state-value constraints are again linear and can be solved by linear algebra. Thus, the first round is handled algebraically, while the remaining rounds are handled by the usual probabilistic differential propagation. This hybrid algebraic-differential strategy enabled the first practical 4-round collisions on KECCAK-224 and KECCAK-256.

Subsequent progress in collision attacks based on standard differential cryptanalysis mainly came from extending and refining the connector. Qiao *et al.* [12] extended the one-round linear-algebraic connector of Dinur *et al.* to two rounds by fully linearizing an additional χ layer. By restricting S-box input values to

suitable affine subspaces, the relevant S-box outputs become affine functions on those subspaces, allowing the additional connector conditions to be incorporated into linear systems. This extension, however, consumes many degrees of freedom. Song *et al.* [13] and Guo *et al.* [6] reduced this cost by non-full S-box linearization, where only the output bits needed by the connector are linearized. This saved degrees of freedom and enabled practical 5-round collision attacks on `SHA3-224` and `SHA3-256`. Nevertheless, value-linearization techniques become less effective for high-capacity instances such as `SHA3-384` and `SHA3-512`, where the smaller rate leaves fewer message-controlled degrees of freedom for satisfying the added linear conditions. For `SHA3-384`, Huang *et al.* [8] addressed this limitation in the standard differential setting by using two-block messages together with a SAT-based connector and a deduce-and-sieve procedure, obtaining collision attacks for 4-round `SHA3-384` with a practical complexity.

A separate line of work changed what the differential characteristic tracks from pairwise XOR differences between two states to internal differences within one state. Internal differentials were introduced by Peyrin in attacks on `ECHO` and `Grøstl` [11], and Dinur *et al.* [4] generalized and applied them to collision attacks on Keccak. They also developed an analogous variant of TDA, called the Target Internal Difference Algorithm (TIDA). The key motivation is twofold. First, standard differential characteristics of the Keccak permutation have extremely small probabilities, which limited progress especially for larger-output, high-capacity instances. Second, in standard differential cryptanalysis of hash functions, a high-probability characteristic directly gives a collision attack only when it ends with zero output difference, which is a restrictive requirement for Keccak. Internal differentials provide a more flexible subset-based viewpoint: instead of tracking the XOR difference between two independent states, they track structural relations inside one state and try to map inputs satisfying a prescribed internal-difference relation into smaller output subsets with unexpectedly high probability. A birthday search inside these smaller subsets can then be cheaper; this is the squeeze attack. In Keccak, this viewpoint is enabled by periodic symmetries along the z-axis, which are preserved by most round operations, while the sparse round constants perturb them. Using this approach, Dinur *et al.* obtained practical collisions for 3-round Keccak-384 and Keccak-512, and theoretical attacks on 4-round Keccak-384 and 5-round Keccak-256.

Zhang *et al.* [15] further improved internal-differential attacks by introducing conditional internal differentials and an improved two-block TIDA procedure. Conditional internal differentials translate the transition conditions of the first two χ-layers into linear constraints on the affine space of candidate second-block values produced by the two-block TIDA procedure, so that selected messages conform to the first two rounds of the internal-differential characteristic with probability 1. Together with the improved TIDA procedure, this led to collision attacks on all six `SHA-3` functions reduced to up to five rounds. Their subsequent work introduced probabilistic linearization [16] to address the degree-of-freedom loss caused by deterministic TIDA connector construction. In the deterministic linearization, the input difference of each active S-box is typically constrained

to a low-dimensional affine subspace, which introduces many linear equations. Probabilistic linearization relaxes this requirement. For an active S-box with prescribed output difference δ_{out}, it selects a higher-dimensional affine subspace $U \subseteq \mathbb{F}_2^5$ for the possible input difference δ_{in}. This imposes fewer linear constraints, but not every $\delta_{\text{in}} \in U$ is a valid input difference leading to δ_{out}. Thus, probabilistic linearization trades deterministic validity for fewer equations and an additional probability factor. With suitable choices of the affine input-difference subspaces and the target internal-differential characteristic, the reduction in linear constraints can outweigh this probability loss. Using this idea, Zhang *et al.* obtained the first theoretical collision attack on 5-round `SHA3-384`, with complexity $2^{170.73}$.

`SHA3-384` is a boundary case for the probabilistic-linearization TIDA framework. Its smaller rate leaves substantially fewer message-controlled degrees of freedom for satisfying the connector constraints than `SHA3-224` and `SHA3-256`, while, unlike `SHA3-512`, it has a concrete 5-round probabilistic-linearization TIDA baseline to improve. Thus, it is the natural target for evaluating whether the current probabilistic-linearization TIDA can be further optimized.

In this state-of-the-art probabilistic-linearization TIDA procedure, the connector starts from a target internal difference α_1^*. For each active S-box, where "active" means that the prescribed output difference is nonzero, the connector selects an affine subspace for the corresponding χ-input difference. Each selected subspace has a local differential density, namely the fraction of its elements that are valid input differences for the prescribed output difference; the product of these local densities over all active S-boxes is denoted by p_1^*. The selected subspaces also impose local linear constraints, which are pulled back through the Keccak linear layer to a global linear system E_Δ on the initial-difference variables. This affine-subspace-selection and linear-system-construction step is referred to as the Probabilistic Input Difference System (PIDS) procedure. After Gaussian elimination, E_Δ is separated into a capacity-only subsystem $E_\mathcal{C}$ and a remaining subsystem $E_\mathcal{R}$. In the two-block TIDA procedure, the first message block M_0 is sampled until the inner part of the induced internal difference satisfies $E_\mathcal{C}$. Once M_0 is fixed, the remaining system $E_\mathcal{R}$ leaves an affine solution space for the unfixed variables. Because probabilistic linearization uses affine relaxations of the exact S-box differential constraints, this affine space may contain no valid χ-input-difference assignment. We denote by p_1 the probability that it contains at least one such valid assignment. If this event occurs, the second block M_1 is solved from the corresponding transition constraints; otherwise, M_0 must be resampled. Thus, the connector cost is governed by the rank of $E_\mathcal{C}$, the product-density estimate p_1^*, the dimension and structure of the remaining affine solution space, and the subsequent solvability for M_1.

Conceptually, probabilistic linearization moves TIDA from a deterministic linear-algebraic connector toward a partially probabilistic connector. It keeps the connector linear by allowing higher-dimensional affine subspaces for active S-box input differences, thereby preserving more degrees of freedom. The price is that the remaining S-box differential validity is no longer enforced by the

linear system: a solution of E_Δ may still fail to be a valid χ-input difference. This leaves two coupled issues: how to choose the affine subspaces so that the resulting system is favorable, and how to evaluate how much of the remaining affine solution space can actually be used.

The first issue is an optimization problem in constructing E_Δ. The search space of affine-subspace assignments is large, and the effect of a local S-box choice is visible only after the corresponding constraints are pulled back through the Keccak linear layer and reduced together by Gaussian elimination. The Greedy-PIDS procedure of [16] makes such choices locally, preferring candidates that do not increase the current capacity-only subsystem. However, a locally favorable choice may still lead to a suboptimal global trade-off: after being combined with the choices for other active S-boxes, it may increase the final rank of $E_\mathcal{C}$, shrink the affine solution space left after fixing M_0, or yield an unfavorable product-density estimate p_1^*. Therefore, PIDS construction is a global multi-objective optimization problem rather than a sequence of independent local decisions.

The second issue is more intrinsic to probabilistic linearization itself. The product-density estimate p_1^* measures the local density of valid χ-input differences inside the selected affine subspaces. However, the actual TIDA procedure does not use the remaining solution space as independent trials. The first block M_0 fixes the capacity-part variables, and the attack then searches only inside the corresponding remaining affine solution space. Thus, the relevant event is not whether a random solution of E_Δ is valid, but whether this particular remaining space contains at least one valid input-difference assignment. Valid assignments may cluster over the same capacity-part choices or spread across many such choices. Therefore, the true probability p_1 depends on the effective remaining degrees of freedom, not only on p_1^* or on the raw dimension of the remaining space. Prior work used p_1^* as a conservative proxy in some cases and relied on direct experimental estimation of p_1 in the 5-round `SHA3-384` case. A general way to quantify this effective remaining freedom is therefore needed for reliably applying probabilistic linearization.

Our Contributions. The two issues above motivate the two technical contributions of this work. For the first issue, we propose SA-PIDS, a simulated-annealing-based algorithm for selecting affine subspaces of active S-box input differences. Instead of making locally greedy choices, SA-PIDS treats the construction of E_Δ as a global combinatorial optimization problem. Its evaluation function jointly considers the rank of the capacity-only subsystem $E_\mathcal{C}$, the product-density estimate p_1^*, and the dimension of the affine solution space of the system after fixing M_0. This allows the search to escape local optima and to find better trade-offs for the TIDA connector.

For the second and more intrinsic issue, we refine the evaluation of the actual connector probability p_1. We first formalize why the local product-density estimate p_1^* and the raw dimension of the remaining solution space do not determine the probability sampled by the two-block TIDA procedure. We then model how valid input-difference assignments project onto the capacity-part choices fixed by M_0. This yields the relation $p_1 = p_1^* \cdot \frac{|\mathcal{S}_\mathcal{R}|}{\bar{\kappa}}$, where $|\mathcal{S}_\mathcal{R}|/\bar{\kappa}$ is the effective remaining

freedom and $\bar{\kappa}$ is the average projection multiplicity of valid assignments. Since $\bar{\kappa}$ is infeasible to count directly, we estimate it using harmonic-mean sampling, which corrects the bias caused by larger fibers being sampled more frequently.

Combining these two improvements, and using the same target internal differential characteristic as the previous probabilistic-linearization attack on 5-round SHA3-384, we obtain an improved connector and a refined probability evaluation. The resulting theoretical collision attack improves the previous $2^{170.73}$ bound of Zhang *et al.* [16] to $2^{164.11}$.

Organization. The rest of the paper is organized as follows. Section 2 introduces the necessary background, including the specification of SHA-3 and the fundamental concepts of internal differential analysis. Section 3 reviews the essential framework of internal differential collision attacks and the probabilistic linearization method, which serves as our baseline. Section 4 presents one of our core contributions on improved PIDS. Section 5 provides the formal evaluation of p_1, including the derivation process of the formula and the specific evaluation methods. Section 6 demonstrates the application of our method by providing a detailed account of the updated collision attack on 5-round SHA3-384, including the complexity calculations. We conclude this paper in Section 7.

Table 1. Summary of the best known collision attacks on SHA-3

Function	Rate	Capacity	Output	Claimed collision security (bits)	Best collision attacks			
					Rounds	Complexity	Attack method	Reference
SHA3-224	1152	448	224	112	5	$2^{96.67}$	Internal differential	[16]
					5	Practical	Differential	[13]
SHA3-256	1088	512	256	128	5	$2^{96.67}$	Internal differential	[16]
					5	Practical	Differential	[6]
SHA3-384	**832**	**768**	**384**	**192**	4	2^{76}	Internal differential	[15]
					4	$2^{59.64}$	Differential	[8]
					5	$2^{170.73}$	Internal differential	[16]
					5	$2^{164.11}$	**Internal differential**	**Section 6**
SHA3-512	576	1024	512	256	3	Practical	Internal differential	[4]
					4	$2^{225.29}$	Internal differential	[16]
SHAKE128	1344	256	d	$\min(d/2, 128)$	5	$2^{96.67}$	Internal differential	[16]
					5	Practical	Differential	[12]
					6	$2^{123.5}$	Differential	[7]
SHAKE256	1088	512	d	$\min(d/2, 256)$	4	2^{76}	Internal differential	[15]
					5	$2^{163.28}$	Internal differential	[16]
					6	$2^{232.29}$	Internal differential	[16]

2 Preliminaries

In this section, we first briefly describe the sponge construction, the Keccak hash function, and the SHA-3 instances. Subsequently, we give the related concepts of internal differentials and some notations to be used in this paper.

2.1 Description of SHA-3

Sponge Construction. The sponge construction [1] is an iterated construction for building cryptographic hash functions with variable-length input and arbitrary output length based on a fixed-length permutation. The sponge construction operates on a state of $b = r + c$ bits, which is divided into two parts: the outer part consisting of the first r bits, and the inner part consisting of the last c bits. Here the value r is called the bitrate and the value c the capacity. First, the input message M is first padded with a *multi-rate padding* rule and split into r-bit blocks, then the b bits of the state are initialized to zero.

The sponge construction proceeds in two phases. In the absorbing phase, the message blocks are processed iteratively: each block is XORed into the first r bits of the state, and then the fixed permutation f is applied to the entire b-bit state. After all blocks are processed, the construction switches to the squeezing phase, where the d-bit digest is generated iteratively by outputting the first r bits of the state as a block and then applying f to the state. This process repeats until the desired output length d is reached.

Keccak. Keccak [2] was selected as the winner of the `SHA-3` competition in 2012, it was then formally standardized in 2015 by NIST as `SHA-3` standard [14]. Keccak functions instantiate the sponge construction using a fixed-length permutation, called Keccak-$f[b]$, where the state width b is typically 1600 bits for the `SHA-3` instances. Let denote each bit of the internal state by $\mathbf{A}[x][y][z]$, where $0 \le x < 5$, $0 \le y < 5$ and $0 \le z < w$. The internal state of Keccak-$f[b]$ is organized into a $5 \times 5 \times w$-bit three-dimensional array, where $\mathbf{A}[x][y][\cdot]$ is a lane, $\mathbf{A}[x][\cdot][\cdot]$ is a sheet, $\mathbf{A}[\cdot][y][\cdot]$ is a plane, and $\mathbf{A}[\cdot][\cdot][z]$ is a slice. w is the length of each lane. The Keccak-$f[b]$ permutation consists of n_r rounds, where $n_r = 12 + 2\log_2 w$, each applying five step mappings in the order listed below. All index arithmetic on the x and y coordinates of the state array $\mathbf{A}[x][y][z]$ is performed modulo 5, and on the z coordinate modulo w (the lane size).

$$\theta: \quad \mathbf{A}[x][y][z] \leftarrow \mathbf{A}[x][y][z] + \sum_{y'=0}^{4} \mathbf{A}[x-1][y'][z] + \sum_{y'=0}^{4} \mathbf{A}[x+1][y'][z-1],$$

$$\rho: \quad \mathbf{A}[x][y][z] \leftarrow \mathbf{A}[x][y][z - T(x,y)], \text{ where } T(x,y) \text{ is predefined constant},$$

$$\pi: \quad \mathbf{A}[x][y][z] \leftarrow \mathbf{A}[x'][y'][z], \text{ with } \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} x' \\ y' \end{pmatrix},$$

$$\chi: \quad \mathbf{A}[x][y][z] \leftarrow \mathbf{A}[x][y][z] + (\mathbf{A}[x+1][y][z] + 1) \cdot \mathbf{A}[x+2][y][z],$$

$$\iota: \quad \mathbf{A} \leftarrow \mathbf{A} + RC[i_r], \text{ where } RC[\cdot] \text{ is the round constant}.$$

SHA-3 Instances. The `SHA-3` family consists of four fixed-output cryptographic hash functions and two extendable-output functions (XOFs). The four fixed-output instances, denoted by `SHA3-`d, are defined from Keccak[c] by appending the two-bit suffix '01' to the message, where $b = 1600$, $c = 2d$,

and $d \in \{224, 256, 384, 512\}$. The two XOF instances, namely `SHAKE128` and `SHAKE256`, have capacities $c = 256$ and $c = 512$, respectively, and support an arbitrary output length d. For these two instances, the original message M is appended with the four-bit suffix '1111'. After that, the *multi-rate* padding rule of KECCAK is applied. Their parameters and corresponding security strengths against collision attacks are summarized in Table 1.

2.2 Internal Differentials

In contrast to classic differential analysis that studies standard differential characteristics with high probability by tracking the differences between different messages, internal differential analysis [4,15,16] considers internal differential characteristics between different parts of the internal state. This approach applies to KECCAK because most operations in its round function preserve periodicity along the z-axis, as noted in [2]. We define the relevant notions as follows.

Definition 1 (Symmetric State [4]). *A state* $\mathbf{A}$ *is called a symmetric state if it has period* i *in the* z*-axis. Namely, given a rotation index* $i < w$ *such that the state* $\mathbf{A}$ *satisfies,* $\mathbf{A}[x][y][z] = \mathbf{A}[x][y][(z + i) \bmod w]$*. A necessary condition for such symmetry is that* i *divides* w*.*

Definition 2 (Consecutive Slice Set [4]). *We set* $i = 32$ *in this paper. For* $i = 32$*, a symmetric state* $\mathbf{A}[x][y][z]$ *is composed of two repetitions of slices* $0 - 31$*. Each sequence of slices* $(0 - 31, 32 - 63)$ *is called a consecutive slice set or CSS in short.*

For a symmetric state $\mathbf{A}$, we can express it as $\mathbf{A} = (\hat{\mathbf{A}}, \hat{\hat{\mathbf{A}}})$, where $\hat{\mathbf{A}}$ and $\hat{\hat{\mathbf{A}}}$ are the first and second CSS's respectively.

Since the round constants in KECCAK are not periodic, the ι operation introduces a difference between the two CSSs, which then propagates through subsequent steps. All other step mappings, however, are translation-invariant along the z-axis: they preserve the equality of the two CSSs for symmetric states (see Theorem 1).

Theorem 1. *A state symmetric with period* i *remains symmetric with period* i *after applying the step mappings* θ*,* ρ*,* π*, and* χ*.*

By selecting one CSS as the *representative state*, the differences between the remaining CSS's and this representative are defined as the *internal difference*. In other words, the other CSS can be computed by XORing the representative state with the internal difference. Now we can define a set of messages that satisfies an internal difference.

Definition 3 (Internal Difference [4]). *Let* i *be a period and let* $\mathbf{v}$ *be a state. The set obtained by adding all symmetric states with period* i *to* $\mathbf{v}$ *is called an internal differential and is denoted by*

$$[i, \mathbf{v}] = \{\mathbf{v} + \mathbf{u} \mid \mathbf{u} \text{ is symmetric with period } i\}.$$

If $\mathbf{v} = \mathbf{0}$*, then* $[i, \mathbf{0}]$ *is called the zero internal difference. For* $\mathbf{v} \neq \mathbf{0}$*, the set* $[i, \mathbf{v}]$ *is a coset of* $[i, \mathbf{0}]$*, and* $\mathbf{v}$ *is called a representative state of this internal difference.*

In this paper, the canonical representative state $\mathbf{v}$ is chosen to satisfy $\mathbf{v}[x][y][z] = 0, z \in \{32, 33, \ldots, 63\}$, which is uniquely determined for each internal difference when $i = 32$. For a state $\mathbf{v}$, the canonical representative state of the internal difference is referred to as its internal difference and is denoted by $\Delta(\mathbf{v})$. Then, for a round function R, an internal differential is a pair of internal differences (α_1, α_2), whose transition probability is defined as

$$\Pr\big(\Delta(R(\mathbf{v})) = \alpha_2 \mid \Delta(\mathbf{v}) = \alpha_1\big).$$

The propagation of internal differences can be described in the same way as standard differential characteristics. Since the step mappings θ, ρ, π, and ι are linear, That is, for any $T \in \{\theta, \rho, \pi, \iota\}$, $T([i, \mathbf{v}]) = [i, T(\mathbf{v})]$. In particular, $\theta([i, \mathbf{v}]) = [i, \theta(\mathbf{v})]$, $\rho([i, \mathbf{v}]) = [i, \rho(\mathbf{v})]$, $\pi([i, \mathbf{v}]) = [i, \pi(\mathbf{v})]$ and $\iota([i, \mathbf{v}]) = [i, \iota(\mathbf{v})]$.

The zero internal difference remains invariant under θ, ρ, π, and χ. For the nonlinear layer χ, a non-zero internal difference may propagate to multiple possible internal differences, similarly to the case of standard differences. The corresponding transition rules are described in Supplementary Material C.

Definition 4 (Internal Differential Characteristic [4]). *Let* $L = \pi \circ \rho \circ \theta$ *denote the composition of the linear step mappings before* χ*. Then the internal differential transition in the* j*-th round is written as*

$$\alpha_{j-1} \xrightarrow{L} \beta_{j-1} \xrightarrow{\chi} \alpha_j^* \xrightarrow{\iota} \alpha_j, \qquad j \geq 1.$$

An internal differential characteristic is a sequence of such transitions through several rounds.

For an n_r-round collision attack, the internal differential characteristic is typically extended up to the input of the χ layer in round $n_r - 1$. After this χ layer, each possible non-zero internal difference gives rise to an output subset after the remaining round function, called a *collision subset*. The final collision search is then performed independently within these collision subsets.

2.3 Notations

Notation	Description
n_r	Number of attacked rounds
c	Capacity of a sponge function
r	Rate of a sponge function
b	Width of a KECCAK permutation in bits, $b = r + c$
d	Length of the digest in bits

p	Number of fixed bits in the initial state due to padding
i	Period of a symmetric state
$\theta, \rho, \pi, \chi, \iota$	The five mappings that comprise one KECCAK round
L	Linear layer of KECCAK, *i.e.* $L = \pi \circ \rho \circ \theta$
R	One-round operation of KECCAK, *i.e.* $R = \iota \circ \chi \circ \pi \circ \rho \circ \theta$
$\Delta(\cdot)$	Internal difference of a state
α_j	Initial internal difference of the j-th round
β_j	Input internal difference of the χ-layer in the j-th round
α^*_{j+1}	Output internal difference of the χ-layer in the j-th round
E_Δ	Linear system obtained from affine constraints on χ-input differences, written in the initial-difference variables
$E_\mathcal{C}$	Capacity-only subsystem of E_Δ after Gaussian elimination
$E_\mathcal{R}$	Remaining subsystem of E_Δ after removing E_C
p_1^*	Product-density estimate of the selected affine input-difference subspaces
p_1	Actual probability that the remaining affine solution space after fixing M_0 contains at least one valid input-difference assignment
M_0, M_1	First and second message blocks in the two-block TIDA procedure

3 Overview of the Attack Framework

In this section, we will give an overview of the collision attacks on `SHA-3`. We first describe the squeeze attack and the variant of birthday attack, and then review the basic framework of collision attacks. Finally, we revisit the generalized target internal difference algorithm using the probabilistic linearization method.

3.1 Squeeze Attack and Variant of Birthday Attack

The basic idea of a squeeze attack is to exploit the non-random behavior of the hash function so that a structured subset of inputs is mapped into a smaller output subset with relatively high probability, thereby enabling a more efficient collision search algorithm.

Let the hash function map an input space S to an output space D. A squeeze attack considers an input subset $S' \subseteq S$ and an output subset $D' \subseteq D$. Let p be the probability that a randomly chosen input from S' is mapped into D', and let $q = |D'|/|D|$ be the relative size of D' in the full output space. If the characteristic induces non-random behavior such that $p^2 > q$, then the expected number of inputs required to find a collision in D' is reduced from the generic birthday bound $\sqrt{|D|}$ to $\sqrt{q|D|}/p$.

This idea naturally appears in internal-differential attacks on `SHA-3`, where internal differences define structured input spaces and corresponding output subsets. Dinur *et al.* [4] used this viewpoint in their collision attacks based on internal differentials. Zhang *et al.* [15] further formulated a variant of the birthday attack tailored to this setting. In their framework, the internal differential partitions

the input space into 2^k disjoint subsets $S_1, \ldots, S_{2^k}$, and each subset corresponds to an output subset D_j of size 2^m, called a collision subset. When an input is sampled uniformly from $S' = \bigcup_j S_j$, its output falls into a particular collision subset D_j with probability 2^{-k}. Therefore, the expected number of inputs required to find a collision is reduced to $2^{(k+m)/2}$, instead of the generic birthday complexity over the full output space.

3.2 The Framework of Collision Attacks Using Internal Differentials

A collision attack using internal differentials can be viewed as a squeeze attack. The adversary first generates sufficiently many messages that follow a prescribed internal differential characteristic, and then performs collision search within the collision subsets determined at the end of the characteristic. In this view, a collision subset is a subset of the output space D, and the transition probability of the internal differential characteristic corresponds to the probability p that an input reaches this output subset.

We follow Zhang *et al.*'s framework [16] in our collision attacks, as illustrated in Fig. 1. The overall framework consists of three stages:

- **TIDA Stage**: For the target internal difference α_1, the input difference system E_Δ is established by probabilistic linearization and filter out the first message block M_0 that make E_Δ consistent. After that, select α_0 from the solution space of each E_Δ that can be legally propagated to α_1.
- **Collecting Message Stage**: For each $\beta_0 = \Delta(L(\alpha_0))$ obtained in the previous stage, we solve the differential transition systems $E_{\beta_0 \to \alpha_1^\star}$ and $E_{\beta_1 \to \alpha_2^\star}$, and obtain several subspaces of the second message block M_1 passing the first two rounds by conditional internal differentials. After that, compute after $(n_r - 1)$ rounds functions from the subspaces and store these outputs into different sets.
- **Searching Stage**: Perform brute-force search on the outputs of collision subset sequentially after one round function, until a collision is found.

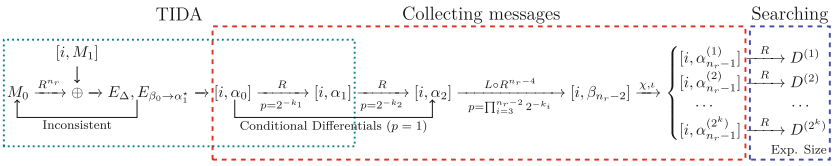

Fig. 1. The framework of n_r-round collision attack [16]

3.3 The Expected Size of Collision Subset

The complexity of Searching Stage is determined by the size of the collision subsets. For an internal differential characteristic $\Delta = (\Delta_0, \ldots, \Delta_4)$, where each

Δ_j consists of 32 bits, the expected size of the corresponding collision subset is defined as $N_\Delta = P_\Delta^{-1}$, where $P_\Delta = \prod_{j=0}^{31} P_{\delta^{(j)}}^{(3)}$ is the probability that two distinct states from the equivalence class $[32, \Delta]$ collide in the first three output lanes after the χ operation. Here, $\delta^{(j)} = (\Delta_0[j], \ldots, \Delta_4[j])$ denotes the 5-bit difference at slice position j, and $P_\delta^{(3)}$ is the precomputed collision probability of the corresponding S-box. The geometric mean of these probabilities is $P_3 = 0.052$, as given in Observation 1 of [16]. When multiple collision subsets are involved, we use the geometric expected size $(P_3^{32})^{-1}$ as a uniform estimate. Thus, for w collision subsets, the overall collision search complexity is estimated as $\left(w \cdot (P_3^{32})^{-1}\right)^{1/2}$. This probabilistic method refines previous analyses [15] that assigned the same search cost to different digest lengths, such as 512-bit and 640-bit outputs, and provides a tighter upper bound on the collision search complexity. The expected sizes of the collision subsets for output lengths up to 640 bits are summarized in Table 4 of [16].

3.4 Generalized TIDA by Probabilistic Linearization

In previous TIDA [4,15], the input difference of each active S-box is constrained in a two-dimensional affine subspace. Although using a 2-dimensional affine subspace provides a more precise characterization of the input differences, it also introduces more constraints into the equation system E_Δ. For `SHA-3` instances with larger capacity, the capacity-related constraints in E_Δ will become more restrictive. Therefore, imposing 2-dimensional affine-subspace constraints on active S-box input differences makes the system harder to satisfy and increases the cost of enumerating M_0. Therefore, to reduce the number of equations in the system, Zhang *et al.* [16] introduced a generalized TIDA by probabilistic linearization technique[1] for constructing a 1-round connector. They considered the higher-dimensional affine subspaces, specifically dimensions 3 and 4. This approach is motivated by the observation that, for a given S-box output difference, the corresponding input differences are not uniformly distributed over these higher-dimensional affine subspaces.

Given a non-zero output difference δ_{out} of the Keccak S-box and for the t-dimensional affine space U, the proportion of input differences of δ_{out} in U is called the *difference density* of U with respect to δ_{out}, and is defined as $\#\{\delta \in U | \delta \to \delta_{\text{out}}\}/|U|$. For δ_{out}, the t-dimensional affine subspace with maximum difference density is called the *max density subspace table* (MDST). Describing such high-dimensional affine subspaces requires fewer equations: one equation for a 4-dimensional affine subspace and two equations for a 3-dimensional affine subspace. This reduces the total number of equations in E_Δ. This process is implemented as the (Probabilistic Input Difference System) PIDS algorithm to produce a probabilistic input difference system E_Δ. A greedy version of PIDS (refer to Algorithm 4) is further applied to minimize the number of equations in the inner part of E_Δ.

[1] We refer to this procedure as probabilistic-linearization TIDA, or simply TIDA in the remainder of this paper.

Probabilistic-Linearization TIDA. This procedure (see Algorithm 3) employs the E_Δ system constructed by PIDS (line 2). Due to the two-message-block technique, E_Δ is divided into two subsystems after applying Gaussian elimination: the capacity-only subsystem $E_\mathcal{C}$, and the remaining subsystem $E_\mathcal{R}$ (line 3). Then, randomly selects a first message block M_0 such that the internal difference of the inner part of $R^{n_r}(M_0||0^c)$ conforms to $E_\mathcal{C}$ (lines 6–12). For each M_0 satisfying $E_\mathcal{C}$, the system $E_{\mathcal{R},\mathcal{C}^*}$ is updated from $E_\mathcal{R}$ by instantiating the capacity variables. Solving the updated $E_{\mathcal{R},\mathcal{C}^*}$ then yields its solution space U_C, which may contain a solution that can generate a valid input difference (line 13). Randomly select a solution from U_C and generate the corresponding input difference β_0. If a compatible input difference is found, the *differential transition system* $E_{\beta_0 \to \alpha_1^*}$ (*i.e.*, E_M in this attack) will be established by adding the transition conditions according to the differential characteristics of the active S-boxes (referred to Property 2), as well as the constraints on the initial message. The system $E_{\beta_0 \to \alpha_1^*}$ will produce a solution space of M_1, where the solution in the space is compatible with both the initial and the target internal difference (lines 14–21).

Complexity Analysis of TIDA. The complexity of the TIDA procedure contains three parts: (1) the probability of enumerating M_0 satisfying $E_\mathcal{C}$, denoted by $p_0 = 2^{-|E_\mathcal{C}|}$, which depends on the number of equations in $E_\mathcal{C}$; (2) the probability that the solution space U_C contains a solution that can generate a compatible input difference β_0, denoted by ${p_1}^2$; (3) the probability that difference transition system $E_{\beta_0 \to \alpha_1^*}$ has a solution, denoted by p_2. The final complexity 2^t of the TIDA procedure is $2^t = 2^{-\log_2 p_0 - \log_2 p_1 - \log_2 p_2}$.

3.5 The Complexity of the Framework

The total time complexity of the attack is determined by the following parts.

1. The complexity of TIDA Stage, denoted by 2^{d_1}. Suppose the time to run TIDA once is 2^t and the average size of the space W_1 output by TIDA is 2^m. Then the complexity will be evaluated by $2^t \cdot \max\{2^{n-m}, 1\}$.
2. The complexity of Collecting Message Stage, denoted by 2^{d_2}. For each message, we need to check whether it conforms to the internal differential characteristics of the $n_r - 2$ rounds. The complexity of this phase is $2^{d_2} = 2^n$, where the n is defined later.
3. The complexity of Searching Stage for a collision, denoted by 2^{d_3}. Denote the j-round transition condition number by k_j. In the last round, there will be $2^{k_{n_r-1}}$ collision subsets, and each collision subset has an expected size of 2^s. The complexity of searching is $2^{d_3} = 2^{(k_{n_r-1}+s)/2}$. Meanwhile, the message size that is sufficient to launch an available attack is $2^n = 2^{k_2+\cdots+k_{n_r-2}+(k_{n_r-1}+s)/2}$.

[2] In Algorithm 3, p_1 is approximately estimated by difference density p_1^* produced by PIDS. A further discussion about p_1 and p_1^* will be presented in Section 5.

The final complexity of the framework is given as Eq. (1):

$$2^{d_1} + 2^{d_2} + 2^{d_3} = 2^t \cdot \max\{2^{n-m}, 1\} + 2^n + 2^{(k_{n_r-1}+s)/2}. \quad (1)$$

4 An Improved PIDS Algorithm

In this section, we improve the construction of the Probabilistic Input Difference System (PIDS) used in the TIDA. We first recall the PIDS construction problem and the Greedy-PIDS baseline, and then present our simulated-annealing-based variant, SA-PIDS.

4.1 PIDS as a Global Optimization Problem

The PIDS procedure of [16] advances TIDA by constructing, via probabilistic linearization, a linear equation system E_Δ from the target internal difference α_1^*. Given a target internal difference α_1^*, it first adds the deterministic linear constraints imposed by non-active S-boxes at the χ-layer whose output difference is prescribed by α_1^*. For each active S-box, it selects an affine subspace for the corresponding input difference. These selected affine subspaces impose linear constraints on the corresponding input-difference variables. After propagation backward through the Keccak linear layer, these local constraints form the global system E_Δ on the initial difference α_0.

Gaussian elimination decomposes E_Δ into a capacity-only subsystem $E_\mathcal{C}$ and a remaining subsystem $E_\mathcal{R}$. The equations in $E_\mathcal{C}$ involve only inner part variables, while the equations in $E_\mathcal{R}$ involve rate-part variables together with capacity-part variables. In the two-block TIDA procedure, the first message block M_0 is sampled until its induced capacity-part difference satisfies $E_\mathcal{C}$. Thus, the number of independent equations in $E_\mathcal{C}$ controls the expected cost of finding a suitable M_0.

The affine-subspace selection also affects the probability of obtaining a valid input difference. Let p_1^* denote the product of the local differential densities of the selected affine subspaces. This quantity estimates the fraction of solutions of E_Δ that correspond to valid χ-layer input-difference transitions. If p_1^* is too small, then even after an M_0 satisfying $E_\mathcal{C}$ is found, the remaining system $E_{\mathcal{R},\mathcal{C}^*}$, which is initialized by $\Delta_\mathcal{C}^*$ produced by M_0, may contain no valid input-difference assignment, and another M_0 has to be tested. Hence, p_1^* also affects the expected number of trials of M_0, although it is only an estimate of the actual probability p_1 analyzed later in Section 5.

The affine-subspace selection problem is therefore a trade-off. Choosing an affine subspace with higher dimension imposes fewer linear equations, which may keep $E_\mathcal{C}$ small and leave a larger remaining solution space, but usually decreases the local differential densities. Conversely, choosing lower-dimensional subspaces can increase p_1^*, but introduces more equations that may propagate through the Keccak linear layer, increase the rank of $E_\mathcal{C}$, or shrink the solution space left

after fixing M_0. Thus, PIDS construction is a multi-objective optimization problem involving the capacity-only constraint rank, the product-density estimate p_1^*, and the remaining solution space.

Greedy-PIDS [16], shown as Algorithm 4 in Supplementary Material, handles this problem by processing active S-boxes sequentially. For each active S-box, it chooses a candidate affine subspace from the maximum-difference-density subspace table (MDST) and prefers candidates whose added constraints do not increase the current number of independent equations in $E_{\mathcal{C}}$. This rule is effective and keeps the search manageable, but it is inherently local. The constraints introduced by different active S-boxes interact only after the KECCAK linear layer and Gaussian elimination, so a choice that is favorable for one S-box may lead to a worse global system. In particular, the final E_{Δ} depends on the processing order of active S-boxes and candidate subspaces, and the greedy criterion does not jointly optimize the capacity-only constraint rank, p_1^*, and the remaining affine solution space.

Moreover, the candidate space explored in previous attack is restricted in order to keep p_1^* from becoming too small. The search mainly uses high-density four-dimensional affine subspaces from MDST, with additional manual fine-tuning by introducing lower-dimensional subspaces and removing some constraints, as was done in [16]. These choices are reasonable for efficiency, but they leave open whether better global combinations exist among the broader set of affine subspace assignments.

This motivates our variant based on the simulated annealing (SA) algorithm, termed SA-PIDS, which considers affine-subspace selection as a global combinatorial optimization issue. Instead of accepting the first locally favorable choice, SA-PIDS searches over subspace assignments more broadly and evaluates each candidate according to its combined effect on $E_{\mathcal{C}}$, p_1^*, and the remaining solution space.

4.2 The SA-PIDS Algorithm

We use a simulated-annealing-based stochastic search to mitigate the premature commitments made by Greedy-PIDS, and call the resulting algorithm SA-PIDS. Simulated annealing [9,5] is not the only possible search strategy for PIDS, but it is a natural fit for the present optimization problem. The search space is finite and discrete: for each active S-box, one chooses one affine subspace from a finite candidate set. The objective is also evaluable: once a complete assignment is fixed, one can construct E_{Δ}, compute the rank of $E_{\mathcal{C}}$, the product-density estimate p_1^*, and the dimension of the remaining affine solution space. At the same time, the objective is not locally separable, because constraints introduced by different active S-boxes interact after being propagated backward through the KECCAK linear layer and Gaussian elimination. Therefore, a locally favorable choice may have a poor global effect. Simulated annealing is useful in this setting because it keeps the search lightweight while allowing non-temporarily improving moves, so that the algorithm can escape local choices that would trap a purely greedy procedure.

At a high level, SA-PIDS consists of four components: a warm start from Greedy-PIDS, a single-S-box replacement neighborhood, a Metropolis acceptance rule, and an exponential cooling schedule. The warm start initializes the search from a feasible greedy assignment; the neighborhood move explores alternative affine-subspace assignments; the Metropolis rule helps avoid premature commitment to locally favorable choices; and the cooling schedule gradually shifts the search from exploration to local refinement.

State, Neighborhood, and Acceptance Rule. SA-PIDS can be viewed as a stochastic neighborhood search over complete affine-subspace assignments. Let $\mathcal{A}$ be the set of active S-box indices in the target internal difference α_1^*. For each $a \in \mathcal{A}$, let $\mathcal{C}_a = \mathrm{MDST}[\delta_{\mathrm{out}}^{(a)}]$ be the candidate set of affine subspaces for the input difference of the a-th active S-box, where $\delta_{\mathrm{out}}^{(a)}$ is its prescribed output difference and MDST denotes the maximum-difference-density subspace table. A state of SA-PIDS is a map

$$\mathcal{S} : \mathcal{A} \to \bigcup_{a \in \mathcal{A}} \mathcal{C}_a, \qquad \mathcal{S}(a) \in \mathcal{C}_a.$$

Equivalently, $\mathcal{S}$ records one selected affine subspace for each active S-box.

A neighboring state is obtained by replacing the selected subspace of one active S-box. More precisely, SA-PIDS chooses an index $a \in \mathcal{A}$ and replaces $\mathcal{S}(a)$ by another candidate subspace from $\mathcal{C}_a$, while keeping all other choices unchanged. This is a local move in the assignment space, but its effect is evaluated globally by reconstructing E_{Δ}, reducing it, and recomputing the score.

Let $\mathsf{Score}(\mathcal{S})$ be the evaluation score, with smaller values indicating better assignments. For a neighboring state $\mathcal{S}'$, set $\Delta = \mathsf{Score}(\mathcal{S}') - \mathsf{Score}(\mathcal{S})$. If $\Delta < 0$, the neighboring state is accepted. If $\Delta \geq 0$, it is accepted with probability $\exp(-\Delta/T)$, where T is the current temperature. Thus, at high temperature, SA-PIDS may accept temporarily worse assignments and explore non-greedy combinations; as T decreases, the search gradually becomes more conservative. The algorithm keeps the best assignment encountered throughout the search and returns it after the cooling schedule terminates.

Evaluation Function. For a complete assignment $\mathcal{S}$, the corresponding system $E_{\Delta}(\mathcal{S})$, the decomposition into $E_{\mathcal{C}}(\mathcal{S})$ and $E_{\mathcal{R}}(\mathcal{S})$, and the estimate of the product-density $p_1^*(\mathcal{S})$ are determined. We score $\mathcal{S}$ by

$$\mathsf{Score}(\mathcal{S}) = s_1 - \log_2 p_1^* - s_2, \tag{2}$$

where $s_1 = \mathrm{rank}(E_{\mathcal{C}})$ is the number of independent capacity-only equations, and 2^{s_2} is the size of the affine solution space of the remaining system after the capacity variables are fixed, and can be calculated without fixing the capacity variables.

The coefficients in Eq. (2) are chosen in the logarithmic scale of the dominant M_0-sampling cost. A sampled M_0 satisfies the capacity-only constraints with probability approximately 2^{-s_1}. Ignoring the refinement modeled later in

Section 5.2, the probability that the remaining affine space contains a valid input-difference assignment is estimated by $p_1^* 2^{s_2}$. Hence, the estimated success probability per sampled M_0 is proportional to $2^{-s_1+\log_2 p_1^*+s_2}$[3]. Minimizing Eq. (2) is therefore equivalent to minimizing the dominant exponent of the expected number of M_0 trials under this approximation. Note that the score is not used as the final complexity estimate; it is only a search objective for SA-PIDS. The final attack complexity is evaluated later using the actual probability p_1.

The Algorithm 1 gives the evaluation procedure. The operation BuildSystem adds the deterministic constraints for non-active S-boxes and the affine-subspace constraints selected by $\mathcal{S}$ for active S-boxes, pulls these constraints back through the Keccak linear layer, and returns the resulting system E_Δ together with the product-density estimate p_1^*.

Algorithm 1 Evaluate

Require: Target internal difference α_1^*; candidate assignment $\mathcal{S}$.
Ensure: Score s, system E_Δ, estimate p_1^*.
1: $(E_\Delta, p_1^*) \leftarrow \text{BuildSystem}(\alpha_1^*, \mathcal{S})$.
2: Reduce E_Δ by Gaussian elimination and decompose it into $E_\mathcal{C}$ and $E_\mathcal{R}$.
3: $s_1 \leftarrow \text{rank}(E_\mathcal{C})$.
4: $s_2 \leftarrow$ dimension of the affine solution space of the remaining system after fixing the capacity variables.
5: $s \leftarrow s_1 - \log_2 p_1^* - s_2$.
6: **return** (s, E_Δ, p_1^*).

Full Description of SA-PIDS. The Algorithm 2 gives the full SA-PIDS procedure. The algorithm is initialized with the output of Greedy-PIDS[4]. This warm start provides a feasible affine-subspace selection with a reasonable capacity-only subsystem, while the subsequent annealing search is allowed to move away from the greedy choices. Starting from this initial solution, the algorithm iteratively improves the assignment through a simulated annealing process: in each inner iteration, a neighboring solution is generated by randomly replacing the affine subspace of a single active S-box with another candidate from $\mathcal{C}_a \setminus \{\mathcal{S}(a)\}$, allowing flexible jumps across subspaces of different dimensions. The new solution is evaluated and its score difference Δ determines acceptance: improvements ($\Delta < 0$) are always accepted, while degradations are accepted with probability $\exp(-\Delta/T)$ under the Metropolis criterion, and the best solution found so far is always preserved. After I_0 such perturbations at the current temperature, the

[3] The additional factor p_2, *i.e.*, the probability that the differential transition system $E_{\beta_0 \to \alpha_1^*}$ is consistent, depends solely on the target difference α_1^* and can therefore be omitted from the score calculation here.

[4] The output of the original Algorithm 4 does not contain the affine subspace selection $\{\mathcal{S}\}$. However, it can be slightly modified to record the affine subspace selected for each active S-box during execution, which will not be elaborated here.

Algorithm 2 SA-PIDS

Require: α_1^*, MDST, T_0, I_0, I_1, $\lambda \in (0,1)$
Ensure: Best E_Δ^{best}, $p_1^{*,\text{best}}$, $\mathcal{S}^{\text{best}}$
Let $\mathcal{A}$ be the active χ-S-box indices in α_1^*
Set $\mathcal{C}_a \leftarrow \text{MDST}[\delta_{\text{out}}(a)]$ for all $a \in \mathcal{A}$
$\mathcal{S} \leftarrow \textsc{Greedy-PIDS}(\alpha_1^*, \text{MDST})$
$(s, E_\Delta, p_1^*) \leftarrow \textsc{Evaluate}(\alpha_1^*, \mathcal{S})$
$(E_\Delta^{\text{best}}, p_1^{*,\text{best}}, \mathcal{S}^{\text{best}}, s^{\text{best}}) \leftarrow (E_\Delta, p_1^*, \mathcal{S}, s)$
$T \leftarrow T_0$
for $i = 1$ to I_1 **do**
  for $j = 1$ to I_0 **do**
    $\mathcal{S}' \leftarrow \mathcal{S}$; pick random a from $\{a \in \mathcal{A} : |\mathcal{C}_a| > 1\}$; pick random U' from $\mathcal{C}_a \setminus \{\mathcal{S}(a)\}$; set $\mathcal{S}'(a) \leftarrow U'$ and keep $\mathcal{S}'(b) = \mathcal{S}(b)$ for all $b \neq a$
    $(s', E'_\Delta, p_1^{*\prime}) \leftarrow \textsc{Evaluate}(\alpha_1^*, \mathcal{S}')$
    $\Delta \leftarrow s' - s$
    if $\Delta < 0$ **or** rand() $< \exp(-\Delta/T)$ **then**
      $(E_\Delta, p_1^*, \mathcal{S}, s) \leftarrow (E'_\Delta, p_1^{*\prime}, \mathcal{S}', s')$
      if $s > s^{\text{best}}$ **then**
        $(E_\Delta^{\text{best}}, p_1^{*,\text{best}}, \mathcal{S}^{\text{best}}, s^{\text{best}}) \leftarrow (E_\Delta, p_1^*, \mathcal{S}, s)$
      end if
    end if
  end for
  $T \leftarrow \lambda T$
end for
return $(E_\Delta^{\text{best}}, p_1^{*,\text{best}}, \mathcal{S}^{\text{best}})$

temperature is reduced via $T \leftarrow \lambda T$; as T approaches zero, the probability of accepting inferior solutions approaches zero too, smoothly transitioning the search from global exploration to local exploitation until it converges to a high-quality affine-subspace selection.

4.3 Experimental Comparison

In this section, we conduct an experiment to compare the results obtained by SA-PIDS and Greedy-PIDS. In our experiments, we set the initial temperature to $T_0 = 500$, the number of iterations at each temperature to $I_0 = 50$, the number of temperature levels to $I_1 = 500$, and the cooling rate to $\lambda = 0.9$[5]. Hence, each run evaluates $I_0 I_1$ neighboring assignments in addition to the initial Greedy-PIDS assignment.

We take the affine subspace selection scheme in [16] as the baseline and verify the effectiveness of the proposed SA-PIDS algorithm. In [16], the main objective of the Greedy-PIDS algorithm was to make the rank of $E_\mathcal{C}$ as small as possible. Through this greedy strategy and manual fine-tuning, the resulting rank of $E_\mathcal{C}$

[5] It is a relatively good set of parameters that we obtained through testing different combinations of parameters. Please refer to the code provided in Section 6.

was 90, with the corresponding differential probability $p_1^* = 2^{-76.97}$ and the dimension of the solution space of $E_{\mathcal{R}}$ being 10.

For the same target difference α_1^* (*i.e.*, α_1 in Table 5 after the ι operation), use the SA-PIDS algorithm to generate a selection of affine subspaces. The evaluation function of this algorithm comprehensively considers three indicators: the rank of $E_{\mathcal{C}}$, the differential probability p_1^*, and the dimension of the solution space of $E_{\mathcal{R}}$. Note that, due to the analysis in Section 5.3, a threshold should be set for the dimension of the solution space of $E_{\mathcal{R}}$ here. If the threshold is exceeded, the evaluation function will return a poor score. Running the algorithm yields the affine subspace selection shown in Table 6, which is applied in the updated attack in Section 6, and the threshold is empirically set to 28. Based on this selection, construct E_{Δ} and perform Gaussian elimination, obtaining a rank of $E_{\mathcal{C}}$ reduced to 85, corresponding to $p_1^* = 2^{-91.19}$, while the dimension of the solution space of $E_{\mathcal{R}}$ increases to 28.

Table 3 summarizes the comparison of the key indicators obtained by the two algorithms. The comprehensive score in the table $s = s_1 - \log_2 p_1^* - s_2$, where s_1 denotes $|E_{\mathcal{C}}|$ and s_2 denotes the dimension of the solution space of $E_{\mathcal{R}}$. A smaller s indicates a better result.

Table 3. Comparison between Greedy-PIDS and SA-PIDS

Indicator	Greedy-PIDS	SA-PIDS	Indicator	Greedy-PIDS	SA-PIDS
s_1	90	**85**	s_2	10	**28**
$\log_2 p_1^*$	-76.97	-91.19	Score	156.97	**148.19**

Since the constraints for non-active S-boxes are fixed, the rank of $E_{\mathcal{C}}$ admits a theoretical lower bound, *i.e.*, the point where none of the active-S-box constraints raise it further. For the α_1^* used in this experiment, this bound is 82. SA-PIDS achieves a rank closer to this optimum than Greedy-PIDS.

Comparing the two sets of data in Table 3, although the probability p_1^* obtained by SA-PIDS is numerically worse than that of Greedy-PIDS, it succeeds in achieving a trade-off among the indicators by introducing a multi-objective evaluation function: it not only reduces the rank of $E_{\mathcal{C}}$ to the theoretical lower bound, but also increases the solution space dimension of $E_{\mathcal{R}}$ from 10 to 28, thereby obtaining a better comprehensive score. The content discussed in Section 5 is exactly how to use the dimension of the solution space of $E_{\mathcal{R}}$ to refine the complexity estimate.

5 Revised Complexity Analysis

In Section 4.2, our proposed SA-PIDS optimizes the selection of affine subspaces. However, there still remains a statistical gap between the calculated differential densities p_1^* and the actual probability p_1. The value of p_1^* is derived by multiplying the differential densities of the chosen affine subspaces, representing the

probability of finding a valid input difference β_0 assuming uniform sampling over the entire solution space of E_{Δ}. However, the actual TIDA procedure operates differently: it first fixes the capacity-part difference $\Delta_{\mathcal{C}}^*$ by randomly sampling M_0, which restricts the search to a specific subsystem $E_{\mathcal{R},\mathcal{C}^*}$. Consequently, the true probability p_1 is strictly defined as the probability that the solution space of $E_{\mathcal{R},\mathcal{C}^*}$ contains at least one solution that derives a valid input difference. Substituting p_1^* for p_1 leads to a biased complexity estimation. To resolve this, this section rigorously models the theoretical relationship between p_1^* and p_1, and proposes a corresponding experimental evaluation method.

5.1 Nonlinear Equation System Perspective

We first introduce an algebraic perspective based on nonlinear equation systems. Given the output difference δ_{out} of an S-box, the set of all valid input differences $\mathcal{D}_{\delta_{\text{out}}} = \{\delta_{\text{in}} \in \mathbb{F}_2^5 \mid \text{DDT}(\delta_{\text{in}}, \delta_{\text{out}}) > 0\}$ is inherently nonlinear. This fundamental property explains why previous works had to resort to approximations: either restricting the search to some affine subspace of the true input difference set (as in classical linearization), or adopting larger high-dimensional affine subspaces at the cost of probabilistic acceptance (as in probabilistic linearization). Therefore, we can construct a Boolean function $f_{\delta_{\text{out}}} : \mathbb{F}_2^5 \to \mathbb{F}_2$ such that $f_{\delta_{\text{out}}}(a) = 1$ if and only if $a \in \mathcal{D}_{\delta_{\text{out}}}$. This Boolean function is fully defined by its truth table: for all $2^5 = 32$ possible input differences a, if a is a valid input difference for δ_{out} then $f_{\delta_{\text{out}}}(a) = 1$, otherwise 0.

Then for all active S-boxes, such Boolean functions can be constructed to characterize the corresponding sets of valid input differences. By substituting the variable constraints induced by the linear layer into these Boolean functions, we obtain a nonlinear equation system E_N. Each solution of E_N corresponds to an initial internal difference α_0 that induces a valid input difference β_0 for the target internal difference α_1^*.

In fact, this is exactly the equation system required by TIDA procedure. However, solving such a nonlinear system directly is difficult. Previous works therefore constructed a linear equation system to avoid solving the nonlinear one explicitly, but using affine subspaces to approximate the valid input difference sets of active S-boxes inevitably introduces a probabilistic factor into the attack complexity. It is noted that in the following analysis, we do not solve such a nonlinear equation system. Actually, it is a tool used for theoretical description.

5.2 Theoretical Modeling for Evaluating p_1

Recall the TIDA stage using two message blocks. First, the PIDS algorithm uses probabilistic linearization to add linear constraints on the input differences of each S-box for the target difference α_1^* (adding constraints of an affine subspace for active S-boxes, and adding five linear constraints to force the input difference to be zero for non-active S-boxes). All these linear constraints are mutually compatible and describe an affine subspace over the whole space of β_0. Via the linear layer of the Keccak permutation, elements of this affine subspace

of β_0 correspond bijectively to the elements of an affine subspace of the initial difference α_0, *i.e.*, the affine subspace described by the equation system E_Δ.

After Gaussian elimination, E_Δ is decomposed into a subsystem $E_\mathcal{C}$ containing only capacity-part variables and the remaining subsystem $E_\mathcal{R}$. As described in Section 5.1, there is a nonlinear equation system E_N that precisely describes all valid α_0 and the compatible corresponding β_0. Denote the solution spaces of E_Δ and E_N by $\mathcal{S}_\Delta$ and $\mathcal{S}_N$, respectively. Let $\mathcal{V} = \mathcal{S}_\Delta \cap \mathcal{S}_N$. Then the proportion of elements in $\mathcal{S}_\Delta$ that yield a valid input difference is $|\mathcal{V}|/|\mathcal{S}_\Delta|$, which is exactly p_1^*. Since the variables of different S-boxes are mutually independent, when the variables in the system are the S-box input differences (*i.e.*, β_0), the probability that an element of $\mathcal{S}_N$ also lies in $\mathcal{S}_\Delta$ can be computed by multiplying the differential densities of the chosen affine subspaces for each S-box. In fact, this property remains valid after variable substitution when the unknowns are expressed in terms of the initial difference α_0. Formally,

$$p_1^* = \frac{|\mathcal{V}|}{|\mathcal{S}_\Delta|} = \frac{|\mathcal{S}_\Delta \cap \mathcal{S}_N|}{|\mathcal{S}_\Delta|} = \prod_{j \text{ in active S-boxes}} \frac{\#\{\delta \in U_j \mid \delta \to \delta_{\text{out}}^{(j)}\}}{|U_j|}, \tag{3}$$

where U_j denotes the affine subspace added to the input difference of the j-th active S-box.

The state of the internal difference Δ can be partitioned into an outer part and an inner part[6], *i.e.*, $\Delta = (\Delta_\mathcal{R}, \Delta_\mathcal{C}) \in \mathbb{F}_2^r \times \mathbb{F}_2^c$. Let $\mathcal{S}_\mathcal{C}$ be the solution space of $E_\mathcal{C}$, *i.e.*, $\mathcal{S}_\mathcal{C} = \{\Delta_\mathcal{C} \in \mathbb{F}_2^c \mid A_c \Delta_\mathcal{C}^T = b_c^T\}$, where A_c is the coefficient matrix of $E_\mathcal{C}$ and b_c^T is the constant vector (written as a column vector). For the system $E_\mathcal{R}$, since it contains variables from both inner and outer parts, it can be written as $(A_r||A_{rc})(\Delta_\mathcal{R}||\Delta_\mathcal{C})^T = b_r^T$, where A_{rc} consists of the last c columns of the coefficient matrix and A_r the first r columns. For an arbitrary fixed $\Delta_\mathcal{C}^* \in \mathbb{F}_2^c$, substituting it into $E_\mathcal{R}$ and yields a system concerning only $\Delta_\mathcal{R}$, denoted as $E_{\mathcal{R},\mathcal{C}^*}$, which has the form $A_r \Delta_\mathcal{R}^T = b_r^T \oplus A_{rc} \Delta_\mathcal{C}^{*T}$. Since its coefficient matrix is always A_r, the dimension of its solution space does not depend on the particular value of $\Delta_\mathcal{C}^*$. Denote the solution space of $E_{\mathcal{R},\mathcal{C}^*}$ by $\mathcal{S}_{\mathcal{R},\mathcal{C}^*}$, and its constant size by $|\mathcal{S}_\mathcal{R}|$.

During the attack, we enumerate the first message block M_0 to obtain a solution $\Delta_\mathcal{C}^*$ of $E_\mathcal{C}$, which determines an affine subspace of the solution space of the original system E_Δ, namely the solution space $\mathcal{S}_{\mathcal{R},\mathcal{C}^*}$ of $E_{\mathcal{R},\mathcal{C}^*}$. At this point, verifying whether the resulting initial internal difference yields a valid input difference is actually performed within $\mathcal{S}_{\mathcal{R},\mathcal{C}^*}$. Consequently, the probability p_1 that a given $\Delta_\mathcal{C}^*$ makes $\mathcal{S}_{\mathcal{R},\mathcal{C}^*}$ contain some $\Delta_\mathcal{R}^*$ such that $(\Delta_\mathcal{R}^*, \Delta_\mathcal{C}^*)$ is a valid input difference can be formally expressed as:

$$p_1 = \Pr\{\exists \Delta_\mathcal{R}^* \in \mathcal{S}_{\mathcal{R},\mathcal{C}^*} : (\Delta_\mathcal{R}^*, \Delta_\mathcal{C}^*) \in \mathcal{S}_N \mid \Delta_\mathcal{C}^* \in \mathcal{S}_\mathcal{C}\}. \tag{4}$$

Define the natural projection map from $\mathcal{S}_\Delta$ to $\mathcal{S}_\mathcal{C}$ by *proj*: $\mathcal{S}_\Delta \to \mathcal{S}_\mathcal{C}, (\Delta_\mathcal{R}, \Delta_\mathcal{C}) \mapsto \Delta_\mathcal{C}$. For any fixed $\Delta_\mathcal{C}^* \in \mathcal{S}_\mathcal{C}$, its fiber in $\mathcal{S}_\Delta$ is exactly the solution space $\mathcal{S}_{\mathcal{R},\mathcal{C}^*}$ of

[6] The rate and capacity refer to the symmetric state (i.e., after state-size reduction). For `SHA3-384` with $i = 32$, they are 412 and 388 bits, respectively; the padding bits are counted as part of the capacity.

the updated system $E_{\mathcal{R},\mathcal{C}^*}$. By properties of affine spaces, the cardinality of this fiber is the constant $|\mathcal{S}_\mathcal{R}|$. The affine space decomposition theorem yields:

$$|\mathcal{S}_\Delta| = |\mathcal{S}_\mathcal{R}||\mathcal{S}_\mathcal{C}|. \tag{5}$$

If the $\Delta_\mathcal{C}^*$ derived from M_0 satisfies that there exists $\Delta_\mathcal{R}^* \in \mathcal{S}_\mathcal{R}$ such that $\mathbf{v} = (\Delta_\mathcal{R}^*, \Delta_\mathcal{C}^*)$ is a solution of E_N, *i.e.*, $\mathbf{v} \in \mathcal{V}$, then this $\Delta_\mathcal{C}^*$ must be an image of some element of $\mathcal{V}$ under the projection *proj* onto the capacity variables. Denote the projection of $\mathcal{V}$ onto the capacity variables by $\mathcal{V}_\mathcal{C}$. This means that the probability p_1 is actually determined by the proportion of $\mathcal{V}_\mathcal{C}$ in the solution space $\mathcal{S}_\mathcal{C}$, *i.e.*,

$$p_1 = \frac{|\mathcal{V}_\mathcal{C}|}{|\mathcal{S}_\mathcal{C}|}. \tag{6}$$

Since *proj* is surjective, several $\mathbf{v} \in \mathcal{V}$ may share the same projection, *i.e.*, correspond to the same element in $\mathcal{V}_\mathcal{C}$, and denote this multiplicity by κ. For different elements of $\mathcal{V}_\mathcal{C}$, the corresponding κ may be different, but from a global perspective there is a fixed numerical relationship between the sizes of $\mathcal{V}_\mathcal{C}$ and $\mathcal{V}$. Define

$$\bar{\kappa} = \frac{|\mathcal{V}|}{|\mathcal{V}_\mathcal{C}|}, \tag{7}$$

where $\bar{\kappa} \geq 1$, and $\bar{\kappa}$ can be regarded as the arithmetic mean of the multiplicities κ for all elements of $\mathcal{V}_\mathcal{C}$.

Combining Eq. (3), (5), (6) and (7), we will derive

$$p_1 = \frac{|\mathcal{V}_\mathcal{C}|}{|\mathcal{S}_\mathcal{C}|} = \frac{|\mathcal{V}|}{|\mathcal{S}_\Delta|} \cdot \frac{|\mathcal{S}_\Delta|}{\bar{\kappa}|\mathcal{S}_\mathcal{C}|} = p_1^* \cdot \frac{|\mathcal{S}_\mathcal{R}|}{\bar{\kappa}}. \tag{8}$$

Eq. (8) establishes a direct quantitative relationship between p_1^* and p_1, transforming the estimation of p_1 into the estimation of $\bar{\kappa}$.

Remark 1 (Effective remaining degrees of freedom). Eq. (8) shows that the remaining affine solution space after fixing M_0 cannot be used as $|\mathcal{S}_\mathcal{R}|$ independent trials. The reason is that the success of a sampled M_0 is determined by its inner part projection: several valid input-difference assignments may project to the same capacity choice and therefore correspond to the same successful M_0. The average number of valid assignments per successful projection is $\bar{\kappa}$. Hence, the useful contribution of the remaining affine space is the quotient $|\mathcal{S}_\mathcal{R}|/\bar{\kappa}$, or, in logarithmic form, $d_{\text{eff}} = \log_2 |\mathcal{S}_\mathcal{R}| - \log_2 \bar{\kappa}$. We call d_{eff} the effective remaining degrees of freedom. This quantity is the part of the remaining affine solution space that actually contributes to increasing the connector probability p_1. The product-density estimate p_1^* accounts for the local density of valid S-box input differences, but it does not capture how these valid assignments are distributed over capacity-side projections. Thus, the core probability-evaluation problem introduced by probabilistic linearization is to estimate this projection-discounted degree of freedom.

In the probabilistic-linearization attack of [16], this projection-discounted freedom was not isolated as a separate structural quantity. As for 6-round

SHAKE256, p_1^* was used as a lower-bound proxy for p_1; for 5-round SHA3-384, p_1 was estimated experimentally for the concrete instance. Our analysis explains both phenomena: p_1^* is the global density term, while $|\mathcal{S}_{\mathcal{R}}|/\bar{\kappa}$ measures how much of the remaining solution space is effectively usable after fixing M_0.

5.3 The Estimation for $\bar{\kappa}$

The application of Eq. (8) reduces the direct evaluation of p_1 to estimating the only unknown $\bar{\kappa}$. In general, directly evaluating $\bar{\kappa}$ is infeasible due to the massive set of input differences. Therefore, we design an experiment to estimate $\bar{\kappa}$. Denote the experimentally estimated value by $\hat{\kappa}$.

The following experimental method is designed to estimate the coefficient $\bar{\kappa}$:

1. During the construction of E_{Δ}, record the set of valid input differences within the affine subspace selected for each active S-box.
2. For each active S-box, independently and uniformly select a value from its valid input difference set, combine them to form β_0. Compute the corresponding α_0 via the inverse linear transformation L^{-1} and extract its inner part $\Delta_{\mathcal{C}}^*$. Repeat this step m times to obtain m samples of $\Delta_{\mathcal{C}}^*$.
3. For each $\Delta_{\mathcal{C}}^*$, substitute it into $E_{\mathcal{R}}$ to obtain $E_{\mathcal{R},\mathcal{C}^*}$. Solve this linear system, and based on the dimension of the solution space $\mathcal{S}_{\mathcal{R},\mathcal{C}^*}$ of $E_{\mathcal{R},\mathcal{C}^*}$, choose:
   - If the dimension of $\mathcal{S}_{\mathcal{R},\mathcal{C}^*}$ is small, exhaustively enumerate all solutions and accurately count the number $N^{(\Delta_{\mathcal{C}}^*)}$ of solutions that yield a valid input difference;
   - If the dimension of $\mathcal{S}_{\mathcal{R},\mathcal{C}^*}$ is too large to enumerate the whole solution space of $E_{\mathcal{R},\mathcal{C}^*}$, randomly select M solutions (*e.g.*, $M = 2^{28}$) and record the number of hits H. Then an estimate of $N^{(\Delta_{\mathcal{C}}^*)}$ is $\hat{N}^{(\Delta_{\mathcal{C}}^*)} = H \cdot |\mathcal{S}_{\mathcal{R}}|/M$.
4. Collect all $N^{(\Delta_{\mathcal{C}}^*)}$ (or $\hat{N}^{(\Delta_{\mathcal{C}}^*)}$) corresponding to the sampled $\Delta_{\mathcal{C}}^*$, and compute the arithmetic mean $\hat{\kappa}_{\text{avg}} = \frac{1}{m} \sum_{\text{sampled } \Delta_{\mathcal{C}}^*} N^{(\Delta_{\mathcal{C}}^*)}$, and the harmonic mean $\hat{\kappa}_{\text{harm}} = \left(\frac{1}{m} \sum_{\text{sampled } \Delta_{\mathcal{C}}^*} 1/N^{(\Delta_{\mathcal{C}}^*)} \right)^{-1}$.

The experiment involves two levels of sampling: *outer sampling*, which randomly selects compatible input differences β_0 to derive α_0 and the associated $\Delta_{\mathcal{C}}^*$, and *inner sampling*, which, for each $\Delta_{\mathcal{C}}^*$, draws solutions from the corresponding $E_{\mathcal{R},\mathcal{C}^*}$ and counts how many, denoted $N^{(\Delta_{\mathcal{C}}^*)}$, yield a valid input difference. The core issue is how to estimate the true $\bar{\kappa}$ as accurately as possible with these limited samplings.

For the inner sampling, when the dimension of the solution space of $E_{\mathcal{R},\mathcal{C}^*}$ is small, one can traverse the entire solution space to obtain the exact $N^{(\Delta_{\mathcal{C}}^*)}$; if the dimension is large, exhaustive traversal is too costly, and one may resort to random sampling of solutions of $E_{\mathcal{R},\mathcal{C}^*}$ to estimate $N^{(\Delta_{\mathcal{C}}^*)}$. However, such an estimate can suffer from large variance. For each $\Delta_{\mathcal{C}}^*$, there always exists a solution in the solution space of the corresponding $E_{\mathcal{R},\mathcal{C}^*}$ that can produce a valid input difference[7]. However, since it is impossible to traverse this solution

[7] This can be clearly drawn by reviewing how $\Delta_{\mathcal{C}}^*$ are generated during the procedure in the experiment above.

space, when such a solution accounts for a small proportion of the entire solution space, random sampling may not be able to sample such a solution.

To resolve this conflict, we can adopt a coordinated strategy between algorithm and evaluation: embed a dimension restriction mechanism directly into the `Evaluate` function of the SA-PIDS algorithm. Specifically, within the `Evaluate` function, add a dimension pre-check: if it is detected that the dimension of the resulting $E_{\mathcal{R},\mathcal{C}^*}$ exceeds a predefined threshold, immediately return a poor score. This mechanism implicitly prunes high-dimensional solution spaces during the search phase, guiding the algorithm to avoid cases that are hard to evaluate accurately and ensuring that the final output differential characteristics all satisfy the condition for precise inner statistics.

For the outer sampling, we must note the following fact: $\Delta_{\mathcal{C}}^*$ is obtained by randomly selecting a valid input difference β_0 and applying the inverse linear layer, and $\Delta_{\mathcal{C}}^*$ itself determines the number $N^{(\Delta_{\mathcal{C}}^*)}$ of valid input differences in the chosen affine subspace (the solution space of $E_{\mathcal{R},\mathcal{C}^*}$). This means that if a certain $\Delta_{\mathcal{C}}^*$ has a larger $N^{(\Delta_{\mathcal{C}}^*)}$, then in the initial random sampling of valid input differences, that $\Delta_{\mathcal{C}}^*$ is more likely to be sampled. This would cause the arithmetic mean computed from the experiment to be significantly biased (since we tend to sample $\Delta_{\mathcal{C}}^*$ with larger $N^{(\Delta_{\mathcal{C}}^*)}$). The phenomenon that "larger quantity leads to higher probability of being sampled" makes the final arithmetic mean overly influenced by high-value samples, resulting in a systematically larger error. This is precisely why we use the harmonic mean in the above experiment to eliminate such scale bias. A more rigorous analysis is given below.

Suppose the elements of $\mathcal{V}_{\mathcal{C}}$ are ${\Delta_{\mathcal{C}}^*}^{(1)}, {\Delta_{\mathcal{C}}^*}^{(2)}, \cdots, {\Delta_{\mathcal{C}}^*}^{(|\mathcal{V}_{\mathcal{C}}|)}$, and the corresponding numbers of valid input differences in the solution space of $E_{\mathcal{R},\mathcal{C}^*}$ are $N_1, N_2, \cdots, N_{|\mathcal{V}_{\mathcal{C}}|}$. Note that since the elements of $\mathcal{V}_{\mathcal{C}}$ are all distinct, the solutions in solution spaces $\mathcal{S}_{\mathcal{R},\mathcal{C}^*}$ are also distinct, so $|\mathcal{V}| = N_1 + N_2 + \cdots + N_{|\mathcal{V}_{\mathcal{C}}|}$, then $\bar{\kappa} = |\mathcal{V}|/|\mathcal{V}_{\mathcal{C}}| = (N_1 + N_2 + \cdots + N_{|\mathcal{V}_{\mathcal{C}}|})/|\mathcal{V}_{\mathcal{C}}|$.

According to the outer sampling in the experiment, the probability that a particular ${\Delta_{\mathcal{C}}^*}^{(j)}$ is drawn equals the ratio of those input differences that induce ${\Delta_{\mathcal{C}}^*}^{(j)}$ to the total set of input differences, *i.e.*, proportional to N_j: $\Pr[\text{draw } {\Delta_{\mathcal{C}}^*}^{(j)}] = N_j/(N_1 + N_2 + \cdots + N_{|\mathcal{V}_{\mathcal{C}}|})$.

If the experiment samples m values of $\Delta_{\mathcal{C}}^*$ in the outer sampling, the experimental estimate $\hat{\kappa}$ should accurately be expressed as $\hat{\kappa} = \sum_j \Pr[\text{draw } {\Delta_{\mathcal{C}}^*}^{(j)}] \cdot N_j$.

In the experiment, computing the arithmetic mean implicitly assumes that every $\Delta_{\mathcal{C}}^*$ is drawn with equal probability, *i.e.*, $\Pr[\text{draw } {\Delta_{\mathcal{C}}^*}^{(j)}] = 1/|\mathcal{V}_{\mathcal{C}}|$. Only under this assumption does the arithmetic mean become meaningful. But this is clearly not the case in practice.

The harmonic mean naturally resolves this problem. Consider the theoretical expectation of the reciprocal $1/N_j$ under the actual sampling distribution:

$$\mathbb{E}\left[\frac{1}{N_j}\right] = \sum_{j=1}^{|\mathcal{V}_{\mathcal{C}}|}\left(\Pr[\text{draw } {\Delta_{\mathcal{C}}^*}^{(j)}] \cdot \frac{1}{N_j}\right) = \sum_{j=1}^{|\mathcal{V}_{\mathcal{C}}|}\left(\frac{N_j}{|\mathcal{V}|} \cdot \frac{1}{N_j}\right) = \sum_{j=1}^{|\mathcal{V}_{\mathcal{C}}|}\frac{1}{|\mathcal{V}|} = \frac{|\mathcal{V}_{\mathcal{C}}|}{|\mathcal{V}|}. \tag{9}$$

Taking the reciprocal of this expectation yields exactly:

$$\left(\mathbb{E}\left[\frac{1}{N_j}\right]\right)^{-1} = \frac{|\mathcal{V}|}{|\mathcal{V}_\mathcal{C}|} = \bar{\kappa}. \tag{10}$$

By the Law of Large Numbers, as the number m of outer samples grows sufficiently large, the experimental harmonic mean $\hat{\kappa}_{\text{harm}}$ converges to this expected value, making it a consistent estimator of the true $\bar{\kappa}$. Substituting this estimate into Eq.(8) yields $\hat{p_1}$, which accurately reflects the true probability and thereby prevents systematic deviations in the attack complexity evaluation. Furthermore, even for limited sample sizes, $\hat{\kappa}_{\text{harm}}$ consistently exhibits less deviation from the true $\bar{\kappa}$ compared to $\hat{\kappa}_{\text{avg}}$.

5.4 Experimental Verification

We verified the above methods against the estimates in [16]. There, p_1 is estimated by splitting the active S-boxes into an analytically tractable group (dependent only on capacity bits) and a group handled via random sampling (additionally influenced by free variables); the overall p_1 is the product of the two factors. For their 5-round `SHA3-384` attack, this gave $p_1 = 2^{-70.15}$ and a final complexity of $2^{170.73}$. The method is limited by the differential density of the chosen affine subspaces: when density is low, sampling enough valid input differences becomes infeasible.

Using the same configuration and the experimental framework of Section 5.3, we exhaustively explored the 10-dimensional solution space of $E_\mathcal{R}$ in the inner sampling, and performed 2^{20} outer samples. The resulting harmonic mean was $\hat{\kappa}_{\text{harm}} = 2^{3.25}$. Taking this as the estimate of $\bar{\kappa}$ and applying Eq.(8), we obtained $p_1 = p_1^*|\mathcal{S}_\mathcal{R}|/\bar{\kappa} = 2^{-76.97+10-3.25} = 2^{-70.22}$, which is close to the value reported in [16].

In summary, our method gives an accurate estimate of p_1 while being independent of the affine-subspace density, making it applicable in more flexible settings and, together with other techniques in this paper, capable of revealing even better attack complexity.

6 Updated Collision Attacks on 5-Round `SHA3-384`

In this section, combining the techniques described above, we present an improved theoretical collision attack on 5-round `SHA3-384`, superior to the previous work [16]. The attack uses the same internal differential characteristic as in [16], shown in Table 5. The period of this trail is $i = 32$, and the numbers of differential transition conditions per round are $(k_2, k_3, k_4) = (25, 18, 16)$. For the target difference α_1^*, there are 77 active S-boxes and 83 non-active S-boxes.

To apply the optimization techniques from Section 5.2, the dimension of the solution space of $E_\mathcal{R}$ (produced after eliminating E_Δ inside the evaluation function) should be considered as an additional factor in the `Evaluate` function,

with a low score returned when this dimension exceeds a certain threshold, so as to facilitate the estimation of $\bar{\kappa}$ using the experiment in Section 5.3. Taking into account our computational capability, we set the dimension threshold to 28, *i.e.*, the SA-PIDS algorithm ensures that the solution space dimension of $E_{\mathcal{R}}$ after elimination does not exceed 28.

After applying SA-PIDS, we obtain a selection of affine subspaces for the active S-boxes as listed in Table 6. Each equation choice for an active S-box is expressed in hexadecimal as (U, p), where U is the coefficient vector and p the constant term. The algorithm outputs $p_1^* = 2^{-91.19}$ and a system E_{Δ} with 469 equations. After Gaussian elimination, E_{Δ} is decomposed into $E_{\mathcal{R}}$ with 384 equations and $E_{\mathcal{C}}$ with 85 equations, giving $p_0 = 2^{-85}$.

For the estimation of $\bar{\kappa}$ in Eq. (8), following the experimental steps of Section 5.3, we randomly sampled 2^{14} valid input differences β_0 for α_1^* and their corresponding initial differences α_0. For each stored α_0, we substituted its inner part values $\Delta_{\mathcal{C}}^*$ into $E_{\mathcal{R}}$ to obtain the updated system $E_{\mathcal{R},\mathcal{C}^*}$. The dimension of the solution space of each $E_{\mathcal{R},\mathcal{C}^*}$ is $412 - 384 = 28$. We exhaustively traversed the solutions of each $E_{\mathcal{R},\mathcal{C}^*}$ and computed the corresponding harmonic mean $\hat{\kappa}_{\text{harm}} = 2^{5.34}$. Adopting it as the estimate for $\bar{\kappa}$, each $\Delta_{\mathcal{C}}^*$ yields on average $2^{5.34}$ valid input differences, hence $p_1 = p_1^* \cdot 2^{28-5.34} = 2^{-68.53}$. As for p_2, we adopt the evaluation from [16], *i.e.*, $p_2 = 2^{-10.58}$.

Next, to find a compatible input difference, each β_0 must be checked against the DDT. According to the above steps, there are about $2^{79.11}$ choices of M_0 that satisfy $E_{\mathcal{C}}$. For each such M_0, we obtain the updated system $E_{\mathcal{R},\mathcal{C}^*}$ with a solution space of dimension 28. Thus the total number of β_0 candidates to verify is $2^{107.11}$, each verification costing 2^q time. The complexity of this process is $2^{107.11} \cdot 77 \cdot 2^q$. Finally, the complexity of the TIDA stage is $2^{85+68.53+10.58} + 2^{113.38} \cdot 2^q \approx 2^{164.11}$.

The message-collection and collision-search phases are analyzed in the same way as in [16]. Once a valid input difference β_0 and its corresponding initial difference α_0 are found via the above procedure, we can set up the differential transition system $E_{\beta_0 \to \alpha_1^*}$ to constrain the initial message space that conforms to this differential propagation. The solution space of $E_{\beta_0 \to \alpha_1^*}$ has size 2^{412-k_0}, where each solution corresponds to a message M_1 such that the internal state satisfies the initial difference α_0 and deterministically propagates to the second-round difference α_1. Since the expected solution space size is sufficient, the attack needs to perform the TIDA stage only once. Solving the differential transition system, we expect to collect $2^{25+18+(200.75+16)/2} = 2^{151.38}$ initial messages, so that a collision is expected after 5 rounds.

In summary, the total complexity of this attack is $2^{164.11} + 2^{151.38} \approx 2^{164.11}$. A comparison of the complexity at each stage with the previous work is given in Table 4.

The source code is available at `https://github.com/KeccakInternalDiff/keccak-internal-differential.git`.

Remark 2. The present improvements target the TIDA connector phase, in particular the PIDS construction and the evaluation of p_1. They are therefore most

Table 4. Comparison with previous attack results

Metric	Previous work [16]	This work	Metric	Previous work [16]	This work
$\lvert E_C \rvert$	90	**85**	Dimension of the solution space of $E_{\mathcal{R}}$	10	**28**
$-\log_2 p_1^*$	76.97	91.19	Verification of β_0 ($\log_2$)	92.19	107.11
$-\log_2 p_1$	70.15	**68.53**	Message collection ($\log_2$)	151.38	151.38
$-\log_2 p_2$	10.58	10.58	Final complexity ($\log_2$)	170.73	**164.11**

useful for instances whose overall attack complexity is dominated by this phase. This is the case for the 5-round `SHA3-384` attack of [16], which relies on probabilistic linearization and Greedy-PIDS. For `SHA3-224`, `SHA3-256`, and `SHAKE128`, the best reduced-round collision attacks already use different standard differential connector techniques, and when using the internal differential, the TIDA phase is not the dominant bottleneck. For `SHA3-512`, the smaller rate $r = 576$ leaves even fewer degrees of freedom, and the known 4-round attack does not provide the same 5-round probabilistic-linearization TIDA baseline to optimize. For `SHAKE256`, the bottleneck of the 6-round attack lies mainly in the message-generation and collision-search stages rather than in the PIDS step. Therefore, `SHA3-384` is the appropriate target for the improvements developed in this paper.

7 Conclusions

This paper investigates collision attacks on round-reduced `SHA-3` via internal differential analysis. We propose SA-PIDS, an enhanced PIDS algorithm that integrates simulated annealing, which significantly outperforms the previous greedy strategy. Additionally, we update the method for calculating the key probability p_1. Applying these improvements to 5-round `SHA3-384` yields updated attack complexities. Extending similar improvements to `SHA3-512` or `SHAKE256` will require addressing additional bottlenecks beyond PIDS, such as the much tighter degree-of-freedom budget or the complexity of the final message-generation and collision-search stages.

Acknowledgements

We would like to thank all anonymous reviewers for their helpful comments. This research is supported by the National Natural Science Foundation of China (No. 62372262), the National Key R&D Program of China (No. 2023YFA1011200), the Fundamental and Interdisciplinary Disciplines Breakthrough Plan of the Ministry of Education of China (No. JYB2025XDXM114), and the High Performance Computing Center of Tsinghua University.

References

1. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Cryptographic sponge functions (2011)

2. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: The KECCAK reference (2011), http://keccak.noekeon.org/Keccak-reference-3.0.pdf, document version 3.0
3. Dinur, I., Dunkelman, O., Shamir, A.: New attacks on Keccak-224 and Keccak-256. In: Canteaut, A. (ed.) FSE 2012. LNCS, vol. 7549, pp. 442–461. Springer, Berlin, Heidelberg (Mar 2012). https://doi.org/10.1007/978-3-642-34047-5_25
4. Dinur, I., Dunkelman, O., Shamir, A.: Collision attacks on up to 5 rounds of SHA-3 using generalized internal differentials. In: Moriai, S. (ed.) FSE 2013. LNCS, vol. 8424, pp. 219–240. Springer, Berlin, Heidelberg (Mar 2014). https://doi.org/10.1007/978-3-662-43933-3_12
5. Geman, S., Geman, D.: Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. IEEE Transactions on Pattern Analysis and Machine Intelligence **PAMI-6**(6), 721–741 (1984). https://doi.org/10.1109/TPAMI.1984.4767596
6. Guo, J., Liao, G., Liu, G., Liu, M., Qiao, K., Song, L.: Practical collision attacks against round-reduced SHA-3. Journal of Cryptology **33**(1), 228–270 (Jan 2020). https://doi.org/10.1007/s00145-019-09313-3
7. Guo, J., Liu, G., Song, L., Tu, Y.: Exploring SAT for cryptanalysis: (quantum) collision attacks against 6-round SHA-3. In: Agrawal, S., Lin, D. (eds.) ASIACRYPT 2022, Part III. LNCS, vol. 13793, pp. 645–674. Springer, Cham (Dec 2022). https://doi.org/10.1007/978-3-031-22969-5_22
8. Huang, S., Ben-Yehuda, O.A., Dunkelman, O., Maximov, A.: Finding collisions against 4-round SHA-3-384 in practical time. IACR Trans. Symm. Cryptol. **2022**(3), 239–270 (2022). https://doi.org/10.46586/tosc.v2022.i3.239-270
9. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **220**(4598), 671–680 (1983). https://doi.org/10.1126/science.220.4598.671, https://www.science.org/doi/abs/10.1126/science.220.4598.671
10. Naya-Plasencia, M., Röck, A., Meier, W.: Practical analysis of reduced-round Keccak. In: Bernstein, D.J., Chatterjee, S. (eds.) INDOCRYPT 2011. LNCS, vol. 7107, pp. 236–254. Springer, Berlin, Heidelberg (Dec 2011). https://doi.org/10.1007/978-3-642-25578-6_18
11. Peyrin, T.: Improved differential attacks for ECHO and Grøstl. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 370–392. Springer, Berlin, Heidelberg (Aug 2010). https://doi.org/10.1007/978-3-642-14623-7_20
12. Qiao, K., Song, L., Liu, M., Guo, J.: New collision attacks on round-reduced Keccak. In: Coron, J.S., Nielsen, J.B. (eds.) EUROCRYPT 2017, Part III. LNCS, vol. 10212, pp. 216–243. Springer, Cham (Apr / May 2017). https://doi.org/10.1007/978-3-319-56617-7_8
13. Song, L., Liao, G., Guo, J.: Non-full sbox linearization: Applications to collision attacks on round-reduced Keccak. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part II. LNCS, vol. 10402, pp. 428–451. Springer, Cham (Aug 2017). https://doi.org/10.1007/978-3-319-63715-0_15
14. of Standards, N.I., Technology: SHA-3 standard: Permutation-based hash and extendable-output functions (Aug 2015). https://doi.org/10.6028/NIST.FIPS.202, https://doi.org/10.6028/NIST.FIPS.202
15. Zhang, Z., Hou, C., Liu, M.: Collision attacks on round-reduced SHA-3 using conditional internal differentials. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part IV. LNCS, vol. 14007, pp. 220–251. Springer, Cham (Apr 2023). https://doi.org/10.1007/978-3-031-30634-1_8

16. Zhang, Z., Hou, C., Liu, M.: Probabilistic linearization: Internal differential collisions in up to 6 rounds of SHA-3. In: Reyzin, L., Stebila, D. (eds.) CRYPTO 2024, Part IV. LNCS, vol. 14923, pp. 241–272. Springer, Cham (Aug 2024). https://doi.org/10.1007/978-3-031-68385-5_8

Supplementary Material

A Internal Differential Characteristic Used in the Attack

In this section, we provide the internal differential characteristics used in the collision attack for 5-round SHA3-384 in Table 5, which was provided in [16].

Table 5. The 1-3.5 round internal differential characteristic used in the attack on 5-round SHA3-384

|----------a---4b|--------8184--2-|--------8496--31|--------8782---c|--------8382--1a|	
|--------8-a---49|---------194--68|--------8-94--31|--------8682---4|--------8382--1a|	
|--------8-a---49|---------184--6-|--------8484--31|--------8782---4|--------8382--1a|	α_1
|--------8-a----9|---------18---6-|--------8494--31|--------8382---4|--------8382--1a|	
|--------8-8---49|---------184--6-|--------849---35|--------8782---4|--------838---18|	
$\downarrow L$	
|--------8------2|-----------8--1|--------8-------|-----------8---|--------8---8---|	
|--------8-------|---------------|---------------|--------8-------|--------8---8---|	
|--------------81|---------------1|---------------|---------------|--------------8-|	β_1
|---------------|---------------|---------------|---------------|---------------|	
|------------8---|------------8---|---------------|-----------8---|---------------|	
$\downarrow \chi$	
|--------8------2|--------------1|---------------|---------------|-----------8---|	
|--------8-------|---------------|---------------|---------------|-----------8---|	
|-------------8--|--------------1|---------------|---------------|---------------|	α_2^*
|---------------|---------------|---------------|---------------|---------------|	
|------------8---|---------------|---------------|---------------|---------------|	
$\downarrow \iota$	
|--------8--8-8--|--------------1|---------------|---------------|-----------8---|	
|--------8-------|---------------|---------------|---------------|-----------8---|	
|-------------8--|--------------1|---------------|---------------|---------------|	α_2
|---------------|---------------|---------------|---------------|---------------|	
|------------8---|---------------|---------------|---------------|---------------|	
$\downarrow L$	
|--------8--8-8--|---------------|---------------|---------------|---------------|	
|---------------|--------------8|-------------4--|---------------|---------------|	
|---------------2|---------------|---------------|---------------|---------------2|	β_2
|-------------4--|--------------8|-------------4--|---------------|---------------|	
|---------------|---------------|---------------|---------------|---------------|	
$\downarrow \chi$	
|--------8--8-8--|---------------|---------------|---------------|---------------|	
|---------------8|--------------8|-------------4--|---------------|---------------|	
|---------------2|---------------|---------------|---------------|---------------|	α_3^*
|---------------|--------------8|-------------4--|---------------|---------------|	
|---------------|---------------|---------------|---------------|---------------|	
$\downarrow \iota$	
|---------------a|---------------|---------------|---------------|---------------|	
|---------------8|--------------8|-------------4--|---------------|---------------|	
|---------------2|---------------|---------------|---------------|---------------|	α_3
|---------------|--------------8|-------------4--|---------------|---------------|	
|---------------|---------------|---------------|---------------|---------------|	
$\downarrow L$	
|---------------a|------------8---|---------------|---------------|---------------|	
|---------------|---------------|--------------1-|-----------1----|---------------|	
|---------------|-----------1----|---------------|---------------|---------------|	β_3
|---------------|-------------8-|---------------|---------2------|---------------|	
|---------------|---------------|---------------|---------------|---------------|	

B Selection of Affine Subspaces

Table 6. The selection of equations for each active S-box in the attack on 5-round SHA3-384

In.	δ_{out}	Equation	Prob.	In.	δ_{out}	Equation	Prob.	In.	δ_{out}	Equation	Prob.	In.	δ_{out}	Equation	Prob.
0	0x04	(0x0d, 1)	6/16	36	0x14	(0x00, 0)	10/32	82	0x06	(0x00, 0)	9/32	121	0x18	(0x01, 0) (0x12, 1)	6/8
1	0x11	(0x11, 1) (0x14, 0)	6/8	37	0x06	(0x14, 1)	9/16	85	0x01	(0x01, 1) (0x0a, 0)	6/8	122	0x04	(0x02, 1) (0x04, 1)	6/8
2	0x08	(0x00, 0)	9/32	38	0x03	(0x00, 0)	9/32	87	0x1f	(0x00, 0)	11/32	127	0x1d	(0x0d, 0)	9/16
3	0x19	(0x15, 0) (0x12, 1)	5/8	49	0x18	(0x05, 0) (0x12, 1)	6/8	88	0x1a	(0x00, 0)	12/32	128	0x05	(0x1e, 1)	7/16
4	0x14	(0x00, 0)	10/32	50	0x06	(0x14, 1)	9/16	89	0x18	(0x12, 1)	9/16	130	0x0c	(0x00, 0)	9/32
5	0x06	(0x14, 1)	9/16	52	0x06	(0x14, 1)	9/16	90	0x0c	(0x09, 1)	9/16	131	0x11	(0x00, 0)	9/32
6	0x01	(0x01, 1) (0x14, 0)	6/8	53	0x01	(0x00, 0)	9/32	95	0x1d	(0x00, 0)	11/32	132	0x14	(0x00, 0)	10/32
17	0x1c	(0x00, 0)	10/32	55	0x1f	(0x00, 0)	11/32	96	0x05	(0x00, 0)	10/32	133	0x06	(0x14, 1)	9/16
18	0x06	(0x01, 1) (0x14, 1)	6/8	56	0x12	(0x00, 0)	10/32	97	0x10	(0x04, 1)	6/16	134	0x03	(0x0a, 1)	9/16
20	0x04	(0x04, 1)	9/16	57	0x18	(0x01, 0) (0x12, 1)	6/8	98	0x08	(0x00, 0)	9/32	145	0x08	(0x08, 1)	9/16
21	0x01	(0x01, 1)	9/16	58	0x08	(0x12, 0) (0x08, 1)	6/8	99	0x11	(0x05, 1) (0x02, 0)	6/8	146	0x02	(0x02, 1)	9/16
23	0x1f	(0x00, 0)	11/32	63	0x1d	(0x01, 1) (0x0c, 1)	7/8	100	0x14	(0x00, 0)	10/32	148	0x04	(0x00, 0)	9/32
24	0x1a	(0x00, 0)	12/32	64	0x05	(0x00, 0)	10/32	101	0x06	(0x14, 1)	9/16	151	0x1f	(0x13, 1)	8/16
25	0x18	(0x12, 1)	9/16	65	0x10	(0x00, 0)	9/32	102	0x02	(0x02, 1)	9/16	152	0x1a	(0x00, 0)	12/32
26	0x0c	(0x00, 0)	9/32	66	0x08	(0x00, 0)	9/32	113	0x18	(0x00, 0)	9/32	153	0x18	(0x01, 0) (0x12, 1)	6/8
31	0x1e	(0x16, 0)	9/16	67	0x11	(0x05, 1)	9/16	114	0x04	(0x04, 1)	9/16	154	0x0c	(0x09, 1)	9/16
32	0x05	(0x00, 0)	10/32	68	0x14	(0x00, 0)	10/32	116	0x04	(0x00, 0)	9/32	159	0x1d	(0x00, 0)	11/32
33	0x10	(0x00, 0)	9/32	69	0x06	(0x14, 1)	9/16	117	0x01	(0x01, 1)	9/16				
34	0x08	(0x08, 1)	9/16	70	0x03	(0x0a, 1)	9/16	119	0x1f	(0x00, 0)	11/32				
35	0x13	(0x00, 0)	10/32	81	0x18	(0x00, 0)	9/32	120	0x1a	(0x00, 0)	12/32				

C Properties of the Keccak S-box

In this section, we will provide some properties of the S-box in Keccak that may be useful in the attack procedure. Denote the input bits of an S-box by $a_0, \ldots, a_4$ and the output bits by $b_0, \ldots, b_4$.

Property 1. By constraining input bits, the output of χ can be partially or fully linearized:

1. Fixing one input bit to linearize two output bits.
2. Fixing a linear relationship between two consecutive input bits (*e.g.*, $a_i + a_{i+1} = 1$) to linearize one output bit.

3. To fully linearize all 5 output bits, the input must be restricted to a 2-dimensional affine subspace. There are 80 such subspaces, characterized by one of the following constraints:

$$a_i = c_0,\ a_j = c_1,\ a_k = c_2 \quad \text{(for three non-consecutive bits)}, \tag{11}$$
$$a_i = c_0,\ a_{i+3} = c_1,\ a_{i+1} + a_{i+2} = c_2 \quad \text{(for four consecutive bits)}. \tag{12}$$

Here, $c_0, c_1, c_2 \in \{0, 1\}$ are constants. Note that the continuity of the bit indices is based on the translation equivalence.

Remark 3. Note that the Property 1.3 has already mentioned in [12,13,6]. However, they only pointed out that there are 80 such 2-dimensional affine subspaces, but did not give a way to find these affine subspaces or calculate their number.

Property 2. Let δ_{in} and δ_{out} be the input and output differences of an S-box, respectively. Then:

1. For a fixed input difference δ_{in}, the output difference δ_{out} is determined by a set of linear conditions on the input values. The number of independent linear conditions q satisfies $2 \le q \le 4$, and each possible δ_{out} occurs with probability 2^{-q}. Specifically,

$$\delta_{\text{out}} = \mathtt{S}(x) \oplus \mathtt{S}(x \oplus \alpha) = C \cdot x \oplus \eta, \tag{13}$$

where

$$C = \begin{pmatrix} \delta_2 & \delta_1 & & & \\ & \delta_3 & \delta_2 & & \\ & & \delta_4 & \delta_3 & \\ \delta_4 & & & \delta_0 & \\ \delta_1 & \delta_0 & & & \end{pmatrix}, \eta = \mathtt{S}(\delta_{in}) = \begin{pmatrix} \delta_0 \oplus (\delta_1 \oplus 1)\delta_2 \\ \delta_1 \oplus (\delta_2 \oplus 1)\delta_3 \\ \delta_2 \oplus (\delta_3 \oplus 1)\delta_4 \\ \delta_3 \oplus (\delta_4 \oplus 1)\delta_0 \\ \delta_4 \oplus (\delta_0 \oplus 1)\delta_1 \end{pmatrix}.$$

2. For a fixed output difference δ_{out}, the set of possible input differences δ_{in} contains between 5 and 17 distinct 2-dimensional affine subspaces. Shown as Table 7. No higher-dimensional subspaces exist.
3. Given both input and output differences, the set of input values satisfying the differential characteristic forms an affine subspace. Its dimension is determined by the number of independent linear conditions derived from the Differential Distribution Table (DDT).

Remark 4. The Property 2.1 is determined by the algebraic degree of the S-box and has already mentioned in [15]. The Property 2.2 is firstly mentioned in [3]. The Property 2.3 can be directly derived from the Property 2.1, and can exactly evaluate the probability of differential transition.

Table 7. Distribution of S-box input differentials

Output difference	Number of input differences	Number of 2D affine subspaces
01 02 04 08 10	9	9
03 06 0c 18 11	9	9
05 0a 14 09 12	10	5
07 0e 1c 19 13	10	5
0b 16 0d 1a 15	12	17
0f 1e 1d 1b 17	11	12
1f	11	10

D The TIDA Procedure

Algorithm 3 Probabilistic-Linearization TIDA [16]

Require: Target internal difference α_1, target number of rounds n_r, and MDST
Ensure: the first block M_0, the value subspace W_1 of the second M_1, initial internal difference α_0

$E_\Delta = \emptyset$, $\alpha_1^* = \alpha_1 \oplus RC[1]$ and $p_1^* = 1$
$(E_\Delta(\Delta_\mathcal{R}, \Delta_\mathcal{C}), p_1^*) = \text{PIDS}(\alpha_1^*, p_1^*)$
Reduce $E_\Delta(\Delta_\mathcal{R}, \Delta_\mathcal{C}) = E_\mathcal{R}(\Delta_\mathcal{R}, \Delta_\mathcal{C}) \cup E_\mathcal{C}(\Delta_\mathcal{C})$
$W_1 = \text{NULL}$
repeat
    $\Delta_\mathcal{C}^* = (\delta'_{r/2-p}, \ldots, \delta'_{b/2-1})$
    repeat
        Randomly select M_0 and compute $\Delta(R^{n_r}(M_0))$
        for $j = r/2 - p$ to $b/2 - 1$ **do**
            $\delta'_j = \Delta(R^{n_r}(M_0))[j]$ ▷ the j-th bit of $\Delta(R^{n_r}(M_0))$
        end for
    until $\Delta_\mathcal{C}^*$ is a solution of $E_\mathcal{C}$
    Solve $E_\mathcal{R}(\Delta_\mathcal{R}, \Delta_\mathcal{C}^*)$ and obtain its solution space U_C
    repeat
        Randomly choose and delete a solution α_0 of U_C
        $\beta_0 = \Delta(L(\alpha_0))$
        if β_0 is the input difference of α_1^* **then**
            Obtain the differential transition system $E_{\beta_0 \to \alpha_1^*}$
            Solve $E_{\beta_0 \to \alpha_1^*}(R^{n_r}(M_0) \oplus (X||0^c))$ on X, and get its solution space W_1 if it has solutions
        end if
    until W_1 is NULL or $U_C = \emptyset$
until W_1 is NULL
return (M_0, W_1, α_0)

Remark 5. The algorithm starts with $p_1^* = 1$. For each active S-box, PIDS selects an affine subspace and multiplies its differential density into p_1^*, repeating until all active S-boxes have been handled.

E The Greedy-PIDS Algorithm

Algorithm 4 Greedy-PIDS [16]

Require: Internal difference α_1^* and MDST.
Ensure: Linear equation system E_Δ, probability p_1^*.

$E_\Delta = \emptyset$, $E'_\Delta = \emptyset$, $E_\mathcal{C}^* = \emptyset$, $p_1^* = 1$ and $W = \Delta(L(\Delta_\mathcal{R}, \Delta_\mathcal{C}))$ ▷ W is a variable vector $(w_0, \ldots, w_{b/2-1})$, $w_i = w_i(\Delta_\mathcal{R}, \Delta_\mathcal{C})$ is a linear function about $\Delta_\mathcal{R} = (\delta_0, \ldots, \delta_{r/2-p-1})$, $\Delta_\mathcal{C} = (\delta_{r/2-p}, \ldots, \delta_{b/2-1})$.
for $j = 0$ to 160 **do**
    Get the output difference δ_{out} of the j-th Sbox from α_1^*
    if $\delta_{out} = 0$ **then**
        $E_\Delta = E_\Delta \cup \{w_{5j}(\Delta_\mathcal{R}, \Delta_\mathcal{C}) = 0, \ldots, w_{5j+4}(\Delta_\mathcal{R}, \Delta_\mathcal{C}) = 0\}$
    end if
end for
Reduce $E_\Delta(\Delta_\mathcal{R}, \Delta_\mathcal{C}) = E_\mathcal{R}(\Delta_\mathcal{R}, \Delta_\mathcal{C}) \cup E_\mathcal{C}(\Delta_\mathcal{C})$, $E_\Delta = E_\mathcal{R}$, $E_\mathcal{C}^* = E_\mathcal{C}$
for $j = 0$ to 160 **do**
    Get the output difference δ_{out} of the j-th Sbox from α_1^*
    if $\delta_{out} \neq 0$ **then**
        repeat
            $r_1 = \mathrm{rank}(E_\Delta)$
            Select and delete a subspace $\mathrm{Ker}(\sum_{j=0}^{4} l_j \cdot 2^j, q)$ from $\mathrm{MDST}[\delta_{out}]$
            $E'_\Delta = E_\Delta \cup \{l_0 \cdot w_{5j}(\Delta_\mathcal{R}, \Delta_\mathcal{C}) + \cdots + l_4 \cdot w_{5j+4}(\Delta_\mathcal{R}, \Delta_\mathcal{C}) = q\}$
            Reduce $E'_\Delta(\Delta_\mathcal{R}, \Delta_\mathcal{C}) = E'_\mathcal{R}(\Delta_\mathcal{R}, \Delta_\mathcal{C}) \cup E'_\mathcal{C}(\Delta_\mathcal{C})$, $E'_\Delta = E'_\mathcal{R}$
            $r_2 = \mathrm{rank}(E'_\Delta)$
            if $r_2 > r_1$ **then**
                $E_\Delta = E'_\Delta$
            else if $\mathrm{MDST}[\delta_{out}] = \emptyset$ **then**
                $E_\mathcal{C}^* = E_\mathcal{C}^* \cup \{l_0 \cdot w_{5j}(\Delta_\mathcal{R}, \Delta_\mathcal{C}) + \cdots + l_4 \cdot w_{5j+4}(\Delta_\mathcal{R}, \Delta_\mathcal{C}) = q\}$
            end if
        until $r_2 \neq r_1$ or $\mathrm{MDST}[\delta_{out}] = \emptyset$
    end if
end for
$E_\Delta = E_\Delta \cup E_\mathcal{C}^*$
return (E_Δ, p_1^*)

The procedure of Greedy-PIDS (Algorithm 4) can be summarized as follows:

1. Process the non-active S-boxes. For each non-active S-box, whose input difference must be zero, five linear equations are added to E_Δ to constrain the five bits of the input difference[8] (line 2–7).
2. Reduce the current E_Δ and split it into two subsystems $E_\mathcal{R}$ and $E_\mathcal{C}$. The latter contains equations of E_Δ that only respect variables in the inner part, while the former contains the remaining equations that have variables in both the inner part and the outer part (line 8).
3. Process the active S-boxes. For each active S-box, choose an affine subspace from the candidate MDST and add the linear constraints of this affine subspace to E_Δ, then reduce E_Δ to obtain $E_\mathcal{R}$ and $E_\mathcal{C}$. If the number of equations in $E_\mathcal{C}$ does not increase or all affine subspaces have been tried, proceed to the next active S-box; otherwise, restore E_Δ and try another affine subspace from the MDST (line 9–25).
4. After all active S-boxes have been processed, return E_Δ and corresponding p_1^*, which is calculated by multiplying the difference density of each selected affine subspace for active S-boxes.

Remark 6. Greedy-PIDS treats non-active and active S-boxes in two separate loops in Algorithm 4 because the fixed constraints for non-active S-boxes serve as the baseline linear system. Active S-box constraints are then added greedily without introducing any new equations into the reduced capacity subsystem $E_\mathcal{C}$.

[8] The added equations essentially replace the variables about the input differences (β_0) with variables about the initial differences (α_0) through an inverse linear layer. Subsequent equations follow the same way.