Automatic Preimage Attacks on Xoodyak Based on Linear Structures

Zhaoyue Huang[1,2], Meicheng Liu[1,2], and Dongdai Lin[1,2]

[1] State Key Laboratory of Cyberspace Security Defense, Institute of Information Engineering, Chinese Academy of Sciences
[2] School of Cyber Security, University of Chinese Academy of Sciences
{huangzhaoyue,liumeicheng,ddlin}@iie.ac.cn

Abstract. Xoodyak was one of the ten finalists in the NIST Lightweight Cryptography standardization process (2018-2023). In this paper, we investigate preimage attacks on Xoodyak-Xof using the linear structures with the allocation model. A key challenge in applying this approach is to identify suitable linear structures that lead to efficient preimage attacks. To address this challenge, we propose an automatic framework for searching suitable linear structures. The framework incorporates three techniques: a mixed exact and linear approximation of the χ function in the final round, a mixed integer linear programming model for describing the propagation of linear variables through the χ and XOR functions, and weighted condition constraints with a weight update algorithm to guide the structure search. Based on the linear structures identified by the proposed framework, we improve the preimage attacks on 2-round and 3-round Xoodyak-Xof with a 128-bit output, reducing their complexities to $2^{93.934}$ and $2^{115.715}$, respectively, which are the best known results to date. In addition, we present the first theoretical preimage attack on 4-round Xoodyak-Xof, with a complexity of $2^{126.245}$.

Keywords: Xoodyak-Xof; Preimage attack; Linear structure; MILP.

1 Introduction

The rapid proliferation of emerging networked systems, including the Internet of Things (IoT), wireless sensor networks, and vehicular networks, has created an increasing demand for cryptographic schemes that can operate efficiently on resource-constrained devices. Unlike conventional computing platforms, such devices typically have limited computational capability, memory, energy, and communication bandwidth, making the direct deployment of traditional symmetric cryptographic algorithms costly or impractical. To address these challenges, the National Institute of Standards and Technology (NIST) launched the Lightweight Cryptography (LWC) Standardization Competition in 2018, aiming to identify authenticated encryption with associated data (AEAD) and hashing schemes suitable for constrained environments. In February 2019, NIST received 57 algorithm submissions, of which 56 were accepted as first-round candidates in April 2019. After further security and performance evaluation, 32 candidates advanced to the second round, and on March 29, 2021, NIST announced ten finalists. Xoodyak[3] was among these finalists and attracted considerable attention because of its lightweight and versatile design.

Xoodyak is an instance of the Cyclist mode of operation and internally employs the Xoodoo permutation[2], a 384-bit permutation whose design is inspired by Keccak-p[1]. Xoodyak supports both keyed and hash modes. While the keyed mode has been extensively investigated, the hash mode has received comparatively less attention. Therefore, this paper focuses on the hash mode, which is based on the sponge construction and provides two main functionalities: the fixed-length hash function Xoodyak-Hash and the eXtendable-Output Function Xoodyak-Xof.

For a sponge-based hash function H, a preimage attack aims to find a message M corresponding to a given target hash value h, such that $H(M) = h$. The attack is considered successful if the required computational complexity is lower than $2^{\min\{|h|,\max\{c/2,|h|-r\}\}}$[8]. Among the existing approaches, linear structure techniques are currently regarded as one of the most effective methods for mounting preimage attacks. Guo et al.[6] proposed the technique of linear structures for improving preimage attacks on round-reduced Keccak at ASIACRYPT 2016. Building upon the 1.5-round linear structures of Keccak-224/256, they presented the first practical preimage attack on 2-round Keccak-224/256. Subsequently, at EUROCRYPT 2019, Li et al.[10] proposed the allocation approach and achieved a practical preimage attack on 3-round Keccak-224 and improved preimage attacks on 3-round Keccak-256 and 4-round Keccak-224/256. Up to now, several preimage attacks on round-reduced Keccak have been successively proposed by leveraging linear structures with the allocation model[11,12,7]. In 2023, Li et al.[9] applied the same approach to round-reduced Ascon-Xof, combined with the automated tool SAT, yielding the best-known preimage attack results based on algebraic attacks.

To date, there have been few preimage attacks against Xoodyak-Xof. At EUROCRYPT 2023, Qin et al.[13] proposed a 3-round preimage attack on Xoodyak-Xof using the meet-in-the-middle (MITM) approach. In 2024, Fu et al.[5] applied a linearization method with the approximate function of the χ operation, improving Qin et al.'s attack results. Subsequently, Dong et al.[4] developed a generic MITM attack framework for sponge-based hash functions at CRYPTO 2024, improving the attack complexity to 2^{121}. Later, at TOSC 2026, Qin et al.[14] further reduced the memory complexity by employing a linear cancellation technique. However, the algebraic approach based on linear structures with the allocation model has not yet been applied to Xoodyak-Xof. To fill this gap, we establish an automated search model for Xoodyak-Xof to identify linear structures suitable for preimage attacks. Based on the proposed model, we improve the existing 2-round and 3-round preimage attacks and present the first 4-round preimage attack against Xoodyak-Xof. The main contributions of this paper are summarized as follows:

Our Contributions. We apply the linear structures with the allocation model to Xoodyak-Xof, imposing constraints on the initial state outside the linear variables to reduce the diffusion of the linear variables, which ultimately yields linear equations in terms of the linear variables at the hash value. To search for the linear structures used in the preimage attack, we propose three techniques for Xoodyak-Xof: a mixed approximation function for the χ function in the last round, a MILP-based systematic model of both the χ function and the XOR function, and a weighted condition constraint. By combining these techniques, we establish an automated search model for linear

structures and apply it to the preimage attack on Xoodyak-Xof. The comparison of attack complexities is presented in Table 1, and the main contributions are summarized in the following three points:

1. We apply a mixed linear and exact approximation method to the preimage attack on Xoodyak-Xof. By exploiting the algebraic structure of the χ function, we approximate $y = x_0 \oplus x_1x_2$ with the linear expression $y = x_0$, which can bring a gain of $2^{0.585}$, while the direct use of the original output offers a gain of 2^1. We give the mixed linear and exact approximation strategy in the last round. Assume that l output bits are treated exactly and t output bits are treated by linear approximation. Then we can obtain a total of $l + t$ linear equations relating the input variables to the output bits and gain $2^{l+0.585t}$.

2. We establish a constraint model that enables automated search of linear structures applicable to Xoodyak-Xof. The state of each variable is represented by two Boolean variables. We introduce three functions and combine them to model the propagation rules of both the χ function and the linear components. To ensure the validity of the linear structures and facilitate solving for message variables, we linearize the nonlinear conditions by transforming them into additional linear conditions. A weighted approach is adopted for condition counting during model search, and an algorithm for updating the weights is provided. Combining the above algorithms, an automated structure search model is established.

3. We improve the preimage attacks on 2-round and 3-round Xoodyak-Xof, and present the first preimage attack on 4-round Xoodyak-Xof. Based on the search model, we apply the linear structures to the round-reduced Xoodyak-Xof, reducing the preimage attack complexities for 2-round and 3-round Xoodyak-Xof to $2^{93.934}$ and $2^{115.715}$, respectively. The first preimage attack on 4-round Xoodyak-Xof is presented, with a complexity of $2^{126.245}$.

Table 1: Summary of preimage attacks on Xoodyak-Xof

Round	Method	Guesses	Final Time	Memory	Ref.
2/12	Algebraic	$2^{94.655}$	$2^{98.735}$	-	[5]
	Algebraic	$2^{90.545}$	$2^{93.934}$	-	Section 4.1
3/12	MitM	-	2^{125}	2^{97}	[13]
	Algebraic	$2^{123.905}$	$2^{123.905}$	-	[5]
	MitM	-	2^{121}	2^{118}	[4]
	MitM	-	2^{121}	2^{59}	[14]
	Algebraic	$2^{115.715}$	$2^{115.715}$	-	Section 4.2
4/12	Algebraic	$2^{126.245}$	$2^{126.245}$	-	Section 4.3

Paper Organization. Section 2 introduces Xoodyak and the linear structures with the allocating model. Section 3 presents our method and the MILP model. Section 4 presents improved preimage attacks on round-reduced Xoodyak-Xof. Section 5 outlines our conclusion. The source code is available at https://github.com/hzyzy510/Automatic_Preimage_Attacks_on_Xoodyak-XOF.

2 Preliminaries

2.1 Description of Xoodyak

Xoodyak, proposed by Daemen et al.[3], is a versatile cryptographic primitive that supports a range of symmetric-key functions, including hashing, authentication, encryption, and authenticated encryption. It operates in two primary modes: the hash mode and the keyed mode (for AEAD). In this paper, we concentrate specifically on its hash-mode operation.

The permutation underlying Xoodyak is Xoodoo, first proposed by Daemen et al.[2] at TOSC 2018. It operates on a 3-dimensional array of size 384 bits. The 384-bit internal state is represented as a $3 \times 4 \times 32$ array of bits, denoted by $X_{y,x}[z]$, where $0 \le x < 4, 0 \le y < 3, 0 \le z < 32$. Additionally, $X_y, 0 \le y < 3$, is a plane of 128 bits. Thus, $X = X_2 \| X_1 \| X_0$. Xoodoo iterates the round function R 12 times, where $R = \rho_{\text{east}} \circ \chi \circ \iota \circ \rho_{\text{west}} \circ \theta$ and χ are the only nonlinear maps. The details of R are given in (1).

$$
\begin{aligned}
\theta &: B_{y,x}[z] \leftarrow X_{y,x}[z] \oplus \sum_{y'=0}^{2} \left(X_{y',x+1}[z+5] \oplus X_{y',x+1}[z+14] \right) \\
\rho_{\text{west}} &: Y_{0,x}[z] \leftarrow B_{0,x}[z], Y_{1,x}[z] \leftarrow B_{1,x+1}[z], Y_{2,x}[z] \leftarrow B_{2,x}[z+11] \\
\iota &: Y_{0,0}[z] \leftarrow Y_{0,0}[z] \oplus RC_r, RC_r \text{ is round constant} \\
\chi &: Z_{y,x}[z] \leftarrow Y_{y,x}[z] \oplus \left(Y_{y+1,x}[z] \oplus 1 \right) \cdot Y_{y+2,x}[z] \\
\rho_{\text{east}} &: X_{0,x}[z] \leftarrow Z_{0,x}[z], X_{1,x}[z] \leftarrow Z_{1,x}[z+1], X_{2,x}[z] \leftarrow Z_{2,x+2}[z+8]
\end{aligned}
\tag{1}
$$

For convenience, we denote $\lambda_1 = \iota \circ \rho_{\text{west}} \circ \theta$ and $\lambda_2 = \rho_{\text{east}}$. To distinguish operations in different rounds, we add a superscript r to denote the round index—e.g., χ^1 represents the χ function in the first round. Similarly, intermediate state values after round r are also indexed by a superscript r. Thus

$$
X^r \xrightarrow{\lambda_1^r} Y^r \xrightarrow{\chi^r} Z^r \xrightarrow{\lambda_2^r} X^{r+1}.
$$

Xoodyak Hash Scheme. The Xoodyak hash scheme, including Xoodyak-Hash and Xoodyak-Xof, is built upon the Xoodoo permutation using the sponge construction to map arbitrarily long inputs into fixed-length outputs and extendable outputs, respectively.

The sponge construction is a mode of operation for building hash functions from permutations, as illustrated in Fig.1. It comprises three components: an $r + c$-bit permutation denoted as f, parameters r, c referred to as the rate and capacity, respectively, and a padding rule. Xoodyak has a rate of 128 bits, so the capacity is 256 bits. Xoodyak-Hash and Xoodyak-Xof employing the sponge construction take in a message M as input and output a digest. The process has two phases. First, the message is padded and split into blocks of r bits. In the absorbing phase, the XOR operation is implemented between each block and the first r bits of the state, which is followed by the permutation f. When all message blocks are processed, the squeezing phase follows. In the squeezing phase, each iteration returns the first r bits of the state and then applies f. It proceeds until a sufficient number of output bits are obtained.

In this paper, we mainly focus on Xoodyak-Xof with a 128-bit hash value and a 128-bit security claim against preimage attacks.

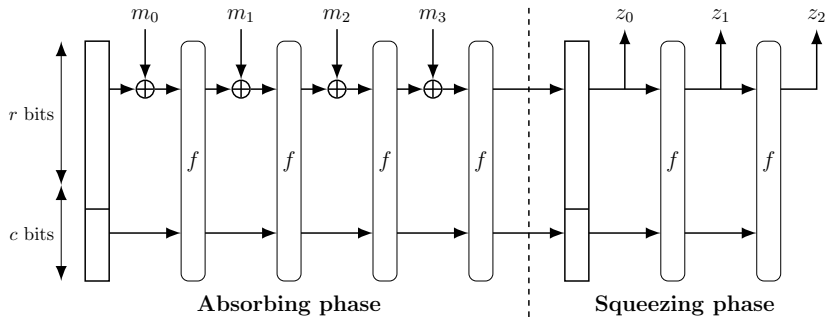

Fig. 1: Sponge construction

2.2 Linear structures with the Allocating Approach

The linear structure technique was first formalized and developed by Guo et al.[6] in 2016. Based on this technique, a series of preimage attacks on Keccak have been proposed. The main idea is to consume the redundant degrees of freedom to constrain the state space of the rate part, which can be used to linearize the permutation. Consequently, the output bits can be expressed as linear functions of the message variables, allowing the preimage problem to be transformed into the solution of a system of linear equations.

In the sponge construction, the absorbed message and public parameters fix part of the initial state, leaving the cryptanalyst with limited freedom to choose it arbitrarily. This restriction becomes a major obstacle when applying linear-structure techniques to single-block preimage attacks. To address this issue, Li and Sun proposed the allocating approach at EUROCRYPT 2019[10], which was used to construct multi-block preimage attacks on round-reduced Keccak-224/256. The main idea is to relax the constraints on the inner state so that more degrees of freedom can be reserved for the attack.

Assume that a linear structure suitable for the attack has been obtained. Let d_1 denote the number of constraints imposed on the inner part of the initial state, and let d_2 denote the number of constraints imposed on the rate part. Furthermore, let d_e be the number of linear equations that can be derived from the target hash value, and let d be the hash length. Then finding an inner part satisfying the structure costs 2^{d_1}, while the random space of one structure has size $2^{d_r} = 2^{r-d_2-d_e}$, where r is the rate size, and the random space also represents the number of algebraic systems that can be generated.

The Final Complexity. For a fixed structure, at most 2^{d_r} algebraic systems can be generated. Solving the d_e linear equations gives a gain of 2^{d_e}, so the expected number of algebraic systems required to obtain a preimage is 2^{d-d_e}. If $2^{d_r} < 2^{d-d_e}$, then $2^{d-d_e-d_r}$ different structures are needed. Hence, the overall search complexity is

$$C_s = \max\left\{2^{d_1} \times 2^{d-d_e-d_r}, 2^{d-d_e}\right\}$$

It should be emphasized that the allocation model requires $2^{d_r} \geq 2^{d_1}$. Failing to meet this condition inflates the complexity to $2^{d_1} \times 2^{d-d_e-d_r}$, which is larger than the ideal 2^{d-d_e}.

The average complexity of obtaining a single solution, denoted by T_s, is measured as the ratio of the number of bit operations required for one Gaussian Elimination turn to the number of bit operations in round-reduced Xoodyak-Xof. In addition, each candidate solution must be verified against the target hash value, denoted by T_v, which is always 2^0 (once target primitive call).

Thus, the final complexity of preimage attack is summarized in Equation (2). When T_s is less than 2^0, the final complexity is 2^{d-d_e}.

$$C_s \times (T_s + T_v) \tag{2}$$

Structure search. Existing linear structures for Keccak are primarily constructed manually, where column-sum conditions are introduced to linearize part of the χ function. Although an automated modeling approach can be applied to search for structures in the lightweight primitive Xoodyak-Xof, there remains a lack of systematic modeling for the propagation rules of different states under the χ function and for how conditions should be imposed.

3 Automatic Preimage Attacks Framework

In this section, we develop an automatic preimage attack framework for round-reduced Xoodyak-Xof based on the linear-structure technique with the allocation approach. The framework combines three techniques. First, in the final round, we use a mixed linear and exact approximation to exploit the algebraic properties of the χ function, thereby transforming the relation between the input and the output into relations involving the input and selected bits of the final round. Second, based on the propagation rules of linear bits, we construct a mixed integer linear programming (MILP) model with constraints describing the XOR and χ operations, enabling an automated search for suitable structures. Finally, we introduce weighted condition constraints to regulate the number of conditions imposed in each round and ensure that the resulting structures can be efficiently linearized. The objective of the model is to maximize the gain provided by the resulting linear equations while minimizing the number of guessed variables.

3.1 Mixed Linear and Exact Approximation in the Last Round

In a preimage attack, our goal is to derive linear relations between the input bits and the output bits. The main obstacle is the nonlinear operation χ. For Xoodoo, the ANF of χ can be written as

$$Z^r_{y,x}[z] = Y^r_{y,x}[z] \oplus \left(Y^r_{y+1,x}[z] \oplus 1\right) Y^r_{y+2,x}[z], \tag{3}$$

where $Z^r_{y,x}[z]$ denotes the output bit after the χ operation in round r, and the row index y is taken modulo 3.

Since the nonlinear term in (3) has the form $B \cdot C$, we first recall the linear approximation property of Boolean polynomials of the form

$$y = A \oplus B \cdot C.$$

If B and C are independent and uniformly distributed Boolean variables, then the approximation $y = A$ holds with probability

$$\Pr[y = A] = \Pr[B \cdot C = 0] = \frac{3}{4} = 2^{-0.415}.$$

Although in multi-round attacks B and C are usually high-degree polynomials rather than independent linear terms, some experimental results suggest that this approximation remains useful in practice. More precisely, Fu et al.[5] made the following observation.
Observation 1. Given a set of Boolean polynomials

$$y_i = A_i \oplus B_i \cdot C_i, \quad 0 \le i < t,$$

when both the number of variables and the number of polynomials are sufficiently large, the simultaneous approximation $\{y_i = A_i, \quad 0 \le i < t\}$, holds with probability p close to 0.75^t, even when B_i and C_i are not linear or independent. Equivalently, if $d = \log_2(p/0.75^t)$, then d is observed to lie in $[-1.15, 1, 15]$ with very high probability.

Therefore, when t nonlinear terms are approximated by removing the product terms, we estimate the success probability by 0.75^t.

In the last round of Xoodyak-Xof, according to the hash value X^{r+1}, we can invert ρ_{east} to obtain the value of Z_2^r. If a linear equation involving X_2^1 can be derived from Z_2^r, then one bit of information can be obtained. We use a mixed linear and exact treatment of χ^r. For some output bits, the nonlinear term in (3) is handled exactly. This can be done by guessing one factor of the product term. For example, if $Y^r_{y+1,x}[z]$ is guessed and both $Y^r_{y,x}[z]$ and $Y^r_{y+2,x}[z]$ are linearized, then $Z^r_{y,x}[z]$ becomes linear. Similarly, if $Y^r_{y+2,x}[z]$ is guessed and both $Y^r_{y,x}[z]$ and $Y^r_{y+1,x}[z]$ are linearized, then the output bit is also linearized exactly. For some other output bits, instead of guessing additional high-degree terms, we apply the linear approximation $Z^r_{y,x}[z] \approx Y^r_{y,x}[z]$, According to Observation 1, if this approximation is used for t output bits, its success probability is estimated as 0.75^t.

This gives the following mixed linear and exact strategy in the last round. Assume that l output bits are treated exactly and t output bits are treated by linear approximation. Then we can obtain in total $l + t$ linear equations relating the input variables to the output bits.

For the l exactly treated output bits, the corresponding equations always hold once the required guessed variables are fixed. Therefore, they do not introduce any probabilistic loss. For the remaining t output bits, we use the approximation $Z^r_{y,x}[z] \approx Y^r_{y,x}[z]$. According to Observation 1, the expected complexity contributed by the approximation part is $2^{0.415t}$. Consequently, the mixed treatment produces $l + t$ linear equations, while its expected complexity is estimated as $2^{0.415t}$.

3.2 Automatic Structure Search Model

Manually selecting structures is not efficient. When the number of attacked rounds increases, the algebraic expressions of the intermediate state become increasingly complicated, making it difficult to determine suitable linear variables and imposed conditions

by hand. Therefore, we transform the search for useful linear structures into an optimization problem and formulate it as a mixed integer linear programming (MILP) model.

The main difficulty in building the MILP model is to describe the propagation of algebraic degree through the linear layer and the nonlinear χ layer, especially after additional conditions are imposed on some intermediate-state bits.

Modeling the Variables. For each state bit $X_{y,x}[z]$, we introduce two binary variables

$$P_{X_{y,x}[z]}, \quad L_{X_{y,x}[z]}.$$

They are used to describe whether the corresponding Boolean polynomial is constant, linear, or nonlinear:

$$(P_{X_{y,x}[z]}, L_{X_{y,x}[z]}) = \begin{cases} (0,0) & \text{if } \deg(X_{y,x}[z]) > 1, \\ (1,1) & \text{if } \deg(X_{y,x}[z]) = 1, \\ (0,1) & \text{if } \deg(X_{y,x}[z]) = 0. \end{cases} \tag{4}$$

Thus, $L_{X_{y,x}[z]} = 1$ means that $X_{y,x}[z]$ has algebraic degree at most one, while $P_{X_{y,x}[z]} = 1$ means that it contains a nonzero linear part. These variables satisfy

$$P_{X_{y,x}[z]} \leq L_{X_{y,x}[z]}. \tag{5}$$

In addition, we introduce two condition variables for each bit $Y_{y,x}[z]$ before the χ layer:

$$con0_{Y_{y,x}[z]}, \quad con1_{Y_{y,x}[z]}.$$

They indicate whether the bit is forced to be a known constant:

$$(con0_{Y_{y,x}[z]}, con1_{Y_{y,x}[z]}) = \begin{cases} (1,0), & Y_{y,x}[z] = 0, \\ (0,1), & Y_{y,x}[z] = 1. \\ (0,0), & \text{else.} \end{cases} \tag{6}$$

The two conditions are mutually exclusive:

$$con0_{Y_{y,x}[z]} + con1_{Y_{y,x}[z]} \leq 1. \tag{7}$$

Since a condition can only be imposed on a constant bit, we also require

$$coni_{Y_{y,x}[z]} \leq L_{Y_{y,x}[z]}, \quad coni_{Y_{y,x}[z]} \leq 1 - P_{Y_{y,x}[z]}, \quad i \in \{0,1\}. \tag{8}$$

Modeling Constraints. We propose three functions and combine them to construct the propagation rules of XOR and χ:

$$y_1 = f_{\text{and}}(x_0, x_1, \ldots, x_{n-1}),$$

$$y_2 = f_{\text{or}}(x_0, x_1, \ldots, x_{n-1}),$$

and

$$y_3 = f_{\text{or_linear}}(x_0, x_1, \ldots, x_{n-1}, y_l).$$

The function f_{and} outputs 1 if and only if all input variables are 1:

$$\begin{cases} y_1 \le x_i, \quad 0 \le i < n, \\ y_1 \ge \sum_{i=0}^{n-1} x_i - (n-1). \end{cases} \tag{9}$$

The function f_{or} outputs 1 if and only if at least one input variable is 1:

$$\begin{cases} y_2 \ge x_i, \quad 0 \le i < n, \\ y_2 \le \sum_{i=0}^{n-1} x_i. \end{cases} \tag{10}$$

The function $f_{\text{or_linear}}$ behaves as an OR function only when $y_l = 1$; otherwise, its output is forced to 0:

$$\begin{cases} y_3 \le y_l, \\ y_3 \ge x_i + y_l - 1, \quad 0 \le i < n, \\ y_3 \le \sum_{i=0}^{n-1} x_i. \end{cases} \tag{11}$$

Modeling XOR. Consider an XOR operation

$$U = V[0] \oplus V[1] \oplus \cdots \oplus V[t-1].$$

The output U is linearized only when all input bits are of algebraic degree at most one. Therefore, we impose

$$L_U = f_{\text{and}}(L_{V[0]}, L_{V[1]}, \ldots, L_{V[t-1]}).$$

If $L_U = 0$, then $P_U = 0$. If $L_U = 1$, the output contains a linear part whenever at least one input contains a linear part. This is modeled by

$$P_U = f_{\text{or_linear}}(P_{V[0]}, P_{V[1]}, \ldots, P_{V[t-1]}, L_U).$$

Modeling χ. Let the input of one χ operation be $Q[0]$, $Q[1]$, and $Q[2]$, and let the output be

$$W = Q[0] \oplus (Q[1] \oplus 1) \cdot Q[2].$$

We first model the product term

$$T = (Q[1] \oplus 1) \cdot Q[2].$$

The product term is forced to zero if either $Q[1] = 1$ or $Q[2] = 0$. Therefore, we introduce

$$K_Q = f_{\text{or}}(con1_{Q[1]}, con0_{Q[2]}),$$

where $K_Q = 1$ means that the product term is removed by an imposed condition.

If no such condition is imposed, the product term is linear only when both multiplicands have degree at most one and at least one of them is constant. This is captured by

$$C_Q = f_{\text{or}}(1 - P_{Q[1]}, 1 - P_{Q[2]}),$$

and

$$L_Q = f_{\text{and}}(L_{Q[1]}, L_{Q[2]}, C_Q).$$

Thus, the product term is linearized if it is either removed by a condition or naturally has degree at most one:

$$L_T = f_{\text{or}}(K_Q, L_Q).$$

Consequently, the output of χ is linearized when both $Q[0]$ and T are linearized:

$$L_W = f_{\text{and}}(L_{Q[0]}, L_T).$$

Next, we describe whether the linear parts of $Q[1]$ and $Q[2]$ propagate to the output. If $Q[1]$ is constant and is not forced to be 1, then the linear part of $Q[2]$ may propagate through the product term. Similarly, if $Q[2]$ is constant and is not forced to be 0, then the linear part of $Q[1]$ may propagate. We introduce

$$y_{Q[2]} = f_{\text{and}}(P_{Q[2]}, 1 - P_{Q[1]}, 1 - con1_{Q[1]}),$$

and

$$y_{Q[1]} = f_{\text{and}}(P_{Q[1]}, 1 - P_{Q[2]}, 1 - con0_{Q[2]}).$$

Finally, if W is linearized, then W contains a linear part whenever one of $Q[0]$, $Q[1]$, or $Q[2]$ contributes a linear term:

$$P_W = f_{\text{or_linear}}(P_{Q[0]}, y_{Q[1]}, y_{Q[2]}, L_W).$$

Modeling the Matching Hash Bit. At model finalization, we introduce two sets of 128 Boolean variables, denoted as $E_{2,x}[z]$ and $L_{2,x}[z]$, where $0 \le x \le 3$ and $0 \le z \le 31$. Here, $E_{2,x}[z] = 1$ indicates that a linear equation is derived from the exact χ function for the corresponding output bit, while $L_{2,x}[z] = 1$ indicates that a linear equation is obtained via the approximation of the χ function. These two variables are mutually exclusive, i.e., they cannot be 1 simultaneously. This constraint is expressed as follows:

$$E_{2,x}[z] \le L_{Z^R_{2,x}[z]}, L_{2,x}[z] \le L_{Y^R_{2,x}[z]}, E_{2,x}[z] + L_{2,x}[z] \le 1,$$

where R is the index of the last round.

Objective Function Setting. The MILP model aims to find a useful $(n-1)$-round structure that maximizes the gain obtained from the resulting linear equations while keeping the number of imposed conditions small.

Let $N_{\text{con}} = \sum(con0_{Y^r_{y,x}[z]} + con1_{Y^r_{y,x}[z]})$ be the number of imposed conditions, and let $N_{\text{var}} = \sum P_{X^1_{x,y}[z]}$ be the number of remaining linear variables that must be assigned in the initial state. To guarantee that the imposed conditions are easily solvable, we require

$$N_{\text{con}} + N_{\text{var}} \le 128. \tag{12}$$

This constraint ensures that the number of conditions does not exceed the available random space.

For the last round, the mixed linear and exact strategy produces two types of equations. Let $l = \sum E_{2,x}[z]$ be the number of output bits treated exactly, and let $t = \sum L_{2,x}[z]$ be the number of output bits treated by linear approximation. Then the total number of obtained linear equations is $N_{\text{eq}} = l + t$. The exact equations do not introduce a probabilistic loss, while the t approximated equations hold with probability $0.75^t = 2^{-0.415t}$. We calculate the final gain in preimage search, denoted by $2^{l+0.585*t}$. Therefore, the final search complexity is calculated as $2^{128-l-0.585*t}$.

In the MILP search stage, we use the complexity of the final search as the main objective and penalize the number of imposed conditions by a small weight. Hence, the objective function is set as

$$\min\left(128 - l - 0.585 * t + \omega N_{\text{con}}\right),$$

where ω is a small positive constant. In our experiments, we set $\omega = 0.001$.

Validity of the Structures Verification Stage. In our setting, imposed conditions may arise in two ways. Some conditions are directly imposed on message state bits. Other conditions are internal state conditions and correspond to linear dependencies involving message variables, which can be satisfied by using multiple message blocks. Therefore, if d_1 denotes the number of internal state conditions and d_2 denotes the number of message-related conditions, the following constraint must be satisfied under the allocation model:

$$d_1 \leq 128 - d_2 - N_{var}. \tag{13}$$

If all conditions are linear, the above relation is equivalent to (12).

The Final Complexity. This objective prioritizes structures that obtain more gain from linear equations, while using the penalty term to avoid unnecessary imposed conditions. The resulting candidate structures are then checked in the validation stage, and the final search complexity is computed using $2^{128-l-0.585*t}$.

3.3 Weighted Constraint for Ensuring Structure Validity

We apply the linear structures with allocation model to round-reduced Xoodyak-Xof and use an automated search model to identify suitable attack structures. However, searching for quadratic or higher-degree structures may be detrimental to the attack. In such structures, the free variables must satisfy nonlinear constraints, and determining their values may require solving a nonlinear system with prohibitively high complexity. To avoid this issue, we linearize the conditions associated with the structure and transform the problem into the solution of a system of linear equations. Accordingly, the relevant measure is not the original number of conditions, but rather the number of linear equations that need to be solved after linearization.

During the search process, however, the exact number of linear equations required for the linearization of the conditions cannot be determined in advance. We therefore employ a weighted-condition strategy based on the following observations. The weight assigned to each round is selected jointly with the structure during the search.

Observation 2. The linearization of the conditions for the 3-round and 4-round Xoodyak-Xof requires guessing a relatively large number of variables. According to the corresponding formulas, linearizing one variable in the 3-round condition requires guessing approximately 105 variables of Y_1, whereas the 4-round condition requires approximately 127 such variables. For comparison, the 2-round case requires guessing only 7 variables.

Observation 3. The search results indicate that the number of conditions imposed in the second round is smaller than that in the first round. Similarly, the number of conditions imposed in the third round is smaller than that in the second round, and this trend continues for subsequent rounds. Thus, the conditions imposed in later rounds should be assigned larger weights than those imposed in earlier rounds.

Combining the above observations, we propose the following Algorithm 1 that simultaneously performs search structure exploration and weight updating, and finally outputs the optimal feasible solution currently obtainable for the attack.

Algorithm 1 Weight-Adaptive Structure Search

Require: Time limit T, Round r, Weight $\boldsymbol{w} = (1, 1, 32, 128)$
Ensure: A feasible solution satisfying validity conditions
Initialize: $flag_valid \leftarrow$ False, $\boldsymbol{w} \leftarrow$ first r bits of w
repeat
  AutoSearchModel($\boldsymbol{w}, r, T$) ▷ Search for the optimal solution under current weights within time T
  Retrieve structural parameters (Y^1, Y^2, Y^3) and condition counts c_1, c_2, c_3 from the search result
  $(v_2, v_3) \leftarrow$ Linearization(Y^1, Y^2, Y^3) ▷ v_i: number of additional Y^1 variables that need to be guessed for linearizing Y^i
  if $c_1 + c_2 + c_3 + v_2 + v_3 + N_{var} \leq 128$ **then** ▷ linearization conditions satisfy validity
    $flag_valid \leftarrow$ True
  else ▷ Update weights
    if $c_2 > 0$ **then**
      $w_1 \leftarrow 1, w_2 \leftarrow 0.1 \cdot \left\lceil 10 \cdot \frac{128 - N_{var} - c_1 - c_3 - v_3}{c_2} \right\rceil, w_3 \leftarrow w_3, w_4 \leftarrow 128$
    else if $c_2 = 0$ and $c_3 > 0$ **then**
      $w_1 \leftarrow 1, w_2 \leftarrow w_2, w_3 \leftarrow 0.1 \cdot \left\lceil 10 \cdot \frac{128 - N_{var} - c_1}{c_3} \right\rceil, w_4 \leftarrow 128$
    end if
    Output: A better result is obtained under current weights, but linearization conditions fail validity
  end if
until $flag_valid$ = True
return current solution

The weight variables are denoted by w, and the corresponding number of conditions in the model is defined as $N_{con} = \sum w_r(con0_{Y^r_{y,x}[z]} + con1_{Y^r_{y,x}[z]}) = \sum w_r c_r$. The initial weight is set to $(1, 1, 32, 128)$, which implies that no conditions are imposed on the fourth round while a limited number of conditions are allowed on the third round. However, subsequent experiments on Xoodyak-Xof reveal that the third round also

Table 2: Linear variables and selected outputs for each lane.

Lane	Linear variables	Exact Selected outputs	Linear Selected outputs
(2,0)	3, 8, 12, 13, 17, 22, 26, 31	5, 14, 24	3, 4, 6, 8, 13, 17, 20, 23, 26, 27, 29, 31
(2,1)	0, 4, 5, 8, 9, 13, 14, 18, 22, 23, 27, 31	-	4, 13, 20, 29
(2,2)	0, 4, 5, 8, 9, 13, 14, 18, 19, 22, 23, 27, 28, 31	0, 1, 5, 10, 14, 15, 19, 23, 24, 28	9
(2,3)	0, 1, 4, 8, 9, 13, 17, 18, 22, 23, 24, 27, 31	0, 1, 5, 6, 10, 14, 15, 19, 23, 24, 28	4, 9, 13, 18, 27,29

contains nearly no conditions. Consequently, weight updates are predominantly applied to the second round. A search for solutions is conducted within a prescribed time limit T, which may be adjusted according to the experimental outcomes. Once a solution is found, we check its validity by verifying whether the number of conditions required for the linearization step meets the validity requirement. If the validity condition is not satisfied, the weights are updated; otherwise, a valid solution is obtained.

In the weight update procedure, when $c_2 > 0$ or $c_2 = 0, c_3 > 0$, the current structure fails the validity test. Upon updating the weights, the number of conditions computed by the function exceeds $128 - N_{var}$, thereby discarding the invalid structure and initiating a new structure search.

Ultimately, within the given time frame, the optimal result under the current weight setting is given. Whenever a superior result is found but fails the linearization validity check, it demonstrates that the current result may be a global optimum.

4 Preimage Attacks on Round-Reduced Xoodyak-XOF

In this section, we apply the above attack framework to preimage attacks on round-reduced Xoodyak-Xof. We apply the attack framework to the 2-round, 3-round, and 4-round Xoodyak-Xof, leading to the best-known preimage attacks for these variants. The MILP models are solved using Gurobi Optimizer version 13.0.2 with 12 threads, on a personal laptop equipped with a 12th Gen Intel(R) Core(TM) i7-12700 CPU.

4.1 Preimage Attacks on 2-Round Xoodyak-XOF

Weight Update Procedure. Algorithm 1 is employed to conduct the structure search for 2-round Xoodyak-Xof. The initial weights are set to $w = (1, 1)$, and the search is performed within a time bound of $T = 2^{18}$ seconds. This yields Structure 1 presented in Appendix A, which contains 76 and 13 conditions in Y^1 and Y^2, respectively. Linearizing Y^2 requires guessing at least 19 additional variables from Y^1. Y^1 is linear with respect to the message variables. Consequently, the linear structures must satisfy $76+13+19 = 108$ linear conditions. There are 38 equations in terms of 38 linear variables, and all outputs are generated using the exact output function, resulting in a final search complexity of

$2^{128-38} = 2^{90}$. However, since $108 + 38 > 128$, the structure fails to satisfy the validity condition under the linearization framework. The weights are subsequently updated to $w = (1, 1.1)$, as $\frac{128-38-76}{13} \approx 1.08 < 1.1$. Under the updated weights, the search yields a new Structure 2 presented in Appendix A containing 60 and 11 conditions in Y^1 and Y^2, respectively. Linearizing Y^2 now requires guessing at least 7 additional variables from Y^1, bringing the total number of linear conditions to be satisfied to 78 and the total number of linear variables to 47. Among these, the number of output bits produced by the exact and approximation functions is 24 and 23, respectively. The resulting condition count satisfies the validity requirement. The final search complexity is $2^{128-24-23*0.585} = 2^{90.545}$.

Quadratic Structure. The final searched structure is summarized in Table 2. The condition positions are listed in Table 3. Here, the linear variables are selected from X_2^1, and the outputs are selected from Y_2^2 or Z_2^2.

Table 3: Bit positions satisfying the corresponding conditions.

Condition	Bit positions z	Condition	Bit positions z
$Y^1_{0,0}[z] = 0$	2, 6, 7, 11, 12, 15, 16, 20, 21, 25, 29, 30	$Y^1_{2,0}[z] = 0$	5, 14
$Y^1_{0,1}[z] = 0$	7, 11, 20, 29, 30	$Y^1_{2,1}[z] = 0$	10, 19, 28
$Y^1_{0,3}[z] = 0$	1, 2, 6, 11, 15, 16, 20, 24, 25, 29	$Y^1_{2,3}[z] = 0$	0, 23
$Y^1_{1,2}[z] = 0$	4, 9, 13, 18, 19, 22, 27, 28	$Y^1_{0,3}[z] = 1$	4, 18, 27
$Y^2_{1,2}[z] = 0$	0, 1, 5, 14, 15, 19, 23, 24, 28	$Y^1_{1,1}[z] = 1$	6, 29
$Y^1_{1,0}[z] = 1$	1, 6, 11, 15, 20, 24, 29	$Y^2_{0,0}[z] = 1$	5, 14
$Y^1_{1,3}[z] = 1$	1, 6, 10, 15, 19, 28		

Linearization. The 11 conditions in Y^2 are expressed as equations involving the variables of Y^1. Upon substitution, 7 distinct quadratic terms arise. It suffices to guess the following 7 additional variables in Y^1 to fully linearize all quadratic conditions.

Table 4: Quadratic terms and guessed variables in the Y^2 conditions

Quadratic terms	$Y^1_{0,2}[14]Y^1_{1,2}[14]$, $Y^1_{0,2}[23]Y^1_{1,2}[23]$, $Y^1_{1,0}[10]Y^1_{2,0}[10]$, $Y^1_{1,0}[19]Y^1_{2,0}[19]$, $Y^1_{1,0}[28]Y^1_{2,0}[28]$, $Y^1_{0,2}[5]Y^1_{1,2}[5]$, $Y^1_{0,2}[0]Y^1_{1,2}[0]$
Guessed variables	$Y^1_{1,2}[0], Y^1_{1,2}[5], Y^1_{1,2}[14], Y^1_{1,2}[23], Y^1_{2,0}[10], Y^1_{2,0}[19], Y^1_{2,0}[28]$

Calculating the Random Space. Using the linearization technique, we obtain 78 linear equations, which consist of 60 linear conditions, 7 additional linear conditions, and 11 quadratic conditions that have already been linearized. Among them, 10 conditions are linearly dependent on the variables in X_2^1; i.e., the internal part of the initial state

must satisfy 10 linear conditions, so $d_1 = 10$. The random space size is unaffected by the linearization technique and is $2^{d_r} = 2^{128-47-(60-10)-7} = 2^{24}$, satisfying $2^{d_r} > 2^{d_1}$.

Table 5: Linear variables and selected outputs for each lane.

Lane	Linear variables	Exact Selected outputs	Linear Selected outputs
(2,0)	0, 5, 14, 23, 28	-	1, 10, 24
(2,1)	-	-	6, 7, 12, 16, 21, 25, 29, 30
(2,2)	1, 2, 6, 10, 11, 15, 19, 20, 24, 28, 29	-	8, 9, 15, 17, 26
(2,3)	7, 12, 16, 21, 30	-	5, 11, 14, 23, 28

The Complexity of Solving Time. Based on the bit operation count in Gaussian Elimination, with n denoting the total number of linear variables, the computation yields $\frac{1}{3}(n^3 - n)$ bit operations. We need to solve remain 68 linear equations with 81 variables. Moreover, we can reduce the number of bit operations by precalculating 57 rows and updating only 11 rows in each turn of guessed variables from Table 4. The precomputation complexity is

$$[81 + (81 + 80) + (81 + 80 + 79) + \cdots + (81 + 80 + \cdots + 26))] = 100016$$

After precomputation, the calculation after each turn becomes:

$$[(81 + 80 \cdots + 25) + (81 + 80 + \cdots + 24) + \cdots + (81 + 80 + \cdots + 15)] = 34386$$

The total number of bit operations for a 2-round Xoodyak-Xof is given by $2 \times (1024 + 768 + 32) = 2 \times 1824 = 3648$. At each turn, we perform Gaussian Elimination on the linear system, expecting to obtain 2^{13} solutions. For each solution, the quadratic structure allows us to generate 47 linear equations from the hash value. The solving complexity is measured as the ratio of the bit operations needed for Gaussian Elimination to the total number of bit operations of 2-round Xoodyak-Xof, which is $\frac{\frac{100016}{2^{11}}+34386}{2^{13}}/3648 \ll 1$. The final complexity of solving the 47 linear equations is $34529/3648 = 2^{3.245}$.

Final Complexity. In summary, by using a single quartic structure and applying a linearization strategy, we improve the preimage attack on 2-round Xoodyak-Xof with the final complexity of $2^{90.545} \cdot (2^{3.245} + 2^0) = 2^{93.934}$.

4.2 Preimage Attacks on 3-Round Xoodyak-XOF

Weight Update Procedure. Algorithm 1 is employed to conduct the structure search for 3-round Xoodyak-Xof. The initial weights are set to $w = (1, 1, 32)$, and the search is performed within a time bound of $T = 2^{16}$ seconds. This yields Structure 1 presented in Appendix B, which contains 76 and 37 conditions in Y^1 and Y^2, respectively. Linearizing Y^2 requires guessing at least 70 additional variables from Y^1. Consequently, the total number of linear conditions amounts to $76 + 37 + 70 = 183$, with a total

of 14 linear variables. Among these, the number of output bits produced by the exact and approximation functions is 11 and 3, respectively. The final search complexity of $2^{128-11-3*0.585} = 2^{115.245}$. However, since 183 + 14 > 128, the structure fails to satisfy the validity condition under the linearization framework. The weights are subsequently updated to $w = (1, 1.1, 32)$, as $\frac{128-14-76}{37} \approx 1.03 < 1.1$. The subsequent weight update procedure is concisely presented in the following Table 6. The final weights are set to $(1, 2.5, 32)$. Under the updated weights, the search yields a new Structure 5 presented in Appendix B containing 90 and 3 conditions in Y^1 and Y^2, respectively. Linearizing Y^2 now requires guessing at least 9 additional variables from Y^1, bringing the total number of linear conditions to be satisfied to 102 and the total number of variables to 21. All outputs are generated using the linear output function. The resulting condition count satisfies the validity requirement. The final search complexity is $2^{128-21*0.585} = 2^{115.715}$.

Table 6: Weight Update Procedure

Attack Structure	Structure1	Structure2	Structure3	Structure4
Variables Count	14	20	13	21
Search Complexity	$2^{115.245}$	$2^{115.47}$	$2^{115.83}$	$2^{115.715}$
Conditions Count	113	103	104	100
Add-Conditions Count	70	35	59	12
Validity	N	N	N	N
Weight Update	$1 \rightarrow 1.1$	$1.1 \rightarrow 1.3$	$1.3 \rightarrow 1.4$	$1.4 \rightarrow 2.5$

Quadratic Structure. The final searched structure is summarized in Table 5. The condition positions are listed in Table 7. Here, the linear variables are selected from X_2^1, and the outputs are selected from Y_2^3.

Table 7: Bit positions satisfying the corresponding conditions.

Condition	Bit positions z	Condition	Bit positions z
$Y^1_{0,1}[z] = 0$	8, 9, 13, 17, 22, 31	$Y^1_{0,0}[z] = 0$	3
$Y^1_{0,2}[z] = 0$	4, 13, 17, 18, 19, 22, 27	$Y^1_{1,0}[z] = 0$	0, 9, 18, 27
$Y^1_{1,1}[z] = 0$	1, 5, 6, 10, 14, 15, 19, 20, 23, 24, 28, 29	$Y^2_{1,1}[z] = 0$	1, 10, 24
$Y^1_{1,2}[z] = 0$	2, 7, 11, 16, 25	$Y^1_{0,1}[z] = 1$	11
$Y^1_{0,0}[z] = 1$	1, 5, 6, 10, 15, 19, 20, 24, 28, 29	$Y^1_{0,3}[z] = 1$	27
$Y^1_{1,0}[z] = 1$	3, 7, 12, 16, 30	$Y^1_{2,0}[z] = 1$	0, 9, 18, 27
$Y^1_{1,1}[z] = 1$	4, 8, 9, 12, 13, 18, 21, 22, 27, 31	$Y^1_{2,2}[z] = 1$	2, 11, 16, 25
$Y^1_{1,2}[z] = 1$	0, 5, 9, 14, 17, 18, 19, 23, 28		
$Y^1_{2,1}[z] = 1$	1, 5, 10, 14, 19, 23, 28, 29		

Linearization. The 3 conditions in Y^2 are expressed as equations involving the variables of Y^1. Upon substitution, 9 distinct quadratic terms arise. It suffices to guess the following 9 additional variables in Y^1 to fully linearize all quadratic conditions.

Table 8: Quadratic terms and guessed variables in the Y^2 conditions

Quadratic terms	$Y^1_{0,3}[7]Y^1_{2,3}[7]$, $Y^1_{0,3}[16]Y^1_{2,3}[16]$, $Y^1_{1,3}[6]Y^1_{2,3}[6]$, $Y^1_{0,1}[0]Y^1_{1,1}[0]$, $Y^1_{0,3}[25]Y^1_{2,3}[25]$, $Y^1_{1,3}[15]Y^1_{2,3}[15]$, $Y^1_{1,3}[24]Y^1_{2,3}[24]$, $Y^1_{0,3}[30]Y^1_{2,3}[30]$, $Y^1_{1,3}[29]Y^1_{2,3}[29]$
Guessed variables	$Y^1_{1,1}[0], Y^1_{2,3}[6], Y^1_{2,3}[7], Y^1_{2,3}[15], Y^1_{2,3}[16], Y^1_{2,3}[24], Y^1_{2,3}[25], Y^1_{2,3}[29], Y^1_{2,3}[30]$

Calculating the Random Space. Using the linearization technique, we obtain 102 linear equations, which consist of 90 linear conditions, 9 additional linear conditions, and 3 quadratic conditions that have already been linearized. Among them, 9 conditions are linearly dependent on the variables in X^1_2; i.e., the internal part of the initial state must satisfy 9 linear conditions, so $d_1 = 9$. The random space size is unaffected by the linearization technique and is $2^{d_r} = 2^{128-21-(90-9)-3} = 2^{23}$, satisfying $2^{d_r} > 2^{d_1}$.

Table 9: Linear variables and selected outputs for each lane.

Lane	Linear variables	Exact Selected outputs	Linear Selected outputs
(2,3)	10, 19, 28	-	8, 17, 31

The Complexity of Solving Time. Now, we need to solve 93 linear equations with 107 variables. We reduce the number of bit operations by precalculating 90 rows and updating only 3 rows in each turn of guessed variables from Table 8. The precomputation complexity is

$$[107 + (107 + 106) + \cdots + (107 + 106 + \cdots + 19))] = 311055$$

After precomputation, the calculation of each turn becomes:

$$[(107 + 106 \cdots + 18) + (107 + 106 + \cdots + 17) + (107 + 106 + \cdots + 16)] = 16925$$

The total number of bit operations for a 3-round Xoodyak-Xof is given by $3 \times (1024 + 768 + 32) = 5472$. At each turn, we perform Gaussian Elimination on the linear system, expecting to obtain 2^{14} solutions. For each solution, the quadratic structure allows us to generate 21 linear equations from the hash value. The solving complexity is measured as the ratio of the bit operations needed for Gaussian Elimination to the total number of bit operations of 3-round Xoodyak-Xof, which is $\frac{\frac{311055}{2^3}+16925}{2^{14}}/5472 \ll 1$. The final complexity of solving the 21 linear equations is $3080/5472 = 2^{-0.829}$.

Final Complexity. In summary, by using a single quartic structure and applying a linearization strategy, we improve the preimage attack on 3-round Xoodyak-Xof with the final complexity of $2^{115.715}$.

4.3 Preimage Attacks on 4-Round Xoodyak-XOF

Weight Update Procedure. Algorithm 1 is employed to conduct the structure search for 4-round Xoodyak-Xof. The initial weights are set to $w = (1, 1, 32, 128)$, and the search is performed within a time bound of $T = 2^{16}$ seconds. This yields Structure 1 presented in Appendix C, which contains 36 and 34 conditions in Y^1 and Y^2, respectively. Linearizing Y^2 requires guessing at least 88 additional variables from Y^1. Consequently, the total number of linear conditions amounts to 36+34+88=158, with a total of 4 linear variables. All outputs are generated using the linear output function, resulting in a final search complexity of $2^{128-4*0.585} = 2^{125.66}$. However, since $158 + 4 > 128$, the structure fails to satisfy the validity condition under the linearization framework. The weights are subsequently updated to $w = (1, 2.6, 32, 128)$, as $\frac{128-4-36}{34} \approx 2.59 < 2.6$. Under the updated weights, the search yields a new Structure 2 presented in Appendix C containing 29 and 15 conditions in Y^1 and Y^2, respectively. Linearizing Y^2 now requires guessing at least 49 additional variables from Y^1, bringing the total number of linear conditions to be satisfied to 93 and the total number of variables to 3. All outputs are generated using the linear output function. The resulting condition count satisfies the validity requirement. The final search complexity is $2^{128-3*0.585} = 2^{126.245}$.

Quadratic Structure. The final searched structure is summarized in Table 9. The condition positions are listed in Table 10. Here, the linear variables are selected from X_2^1, and the outputs are selected from Y_2^4.

Table 10: Bit positions satisfying the corresponding conditions.

Condition	Bit positions z	Condition	Bit positions z
$Y^1_{0,2}[z] = 0$	3, 12, 17, 26	$Y^1_{0,1}[z] = 1$	5, 14, 23, 28
$Y^1_{0,3}[z] = 0$	8, 17, 31	$Y^1_{1,2}[z] = 1$	3, 12, 17, 26
$Y^1_{1,2}[z] = 0$	5, 14, 23, 28	$Y^1_{1,3}[z] = 1$	8, 17, 31
$Y^1_{2,1}[z] = 0$	5, 14, 28	$Y^1_{2,2}[z] = 1$	5, 14, 23, 28
$Y^2_{0,0}[z] = 0$	6, 29	$Y^2_{0,0}[z] = 1$	0, 14, 23
$Y^2_{1,0}[z] = 0$	8, 17	$Y^2_{0,2}[z] = 1$	4, 13
$Y^2_{1,1}[z] = 1$	7, 21, 30	$Y^2_{1,0}[z] = 1$	2, 16, 25

Linearization. The 15 conditions in Y^2 are expressed as equations involving the variables of Y^1. Upon substitution, 49 distinct quadratic terms arise. It suffices to guess the following 49 additional variables in Y^1 to fully linearize all quadratic conditions.

Calculating the Random Space. Using the linearization technique, we obtain 93 linear equations, which consist of 29 linear conditions, 49 additional linear conditions, and 15 quadratic conditions that have already been linearized. Among them, 1 condition

Table 11: Quadratic terms and guessed variables in the Y^2 conditions

Quadratic terms	$Y^1_{0,0}[6]Y^1_{1,0}[6]$, $Y^1_{0,0}[7]Y^1_{1,0}[7]$,$Y^1_{0,0}[15]Y^1_{1,0}[15]$, $Y^1_{0,0}[21]Y^1_{1,0}[21]$, $Y^1_{0,0}[24]Y^1_{1,0}[24]$, $Y^1_{0,0}[29]Y^1_{1,0}[29]$, $Y^1_{0,0}[30]Y^1_{1,0}[30]$, $Y^1_{0,1}[2]Y^1_{1,1}[2]$, $Y^1_{0,1}[3]Y^1_{1,1}[3]$, $Y^1_{0,1}[3]Y^1_{2,1}[3]$, $Y^1_{0,1}[6]Y^1_{2,1}[6]$, $Y^1_{0,1}[9]Y^1_{2,1}[9]$, $Y^1_{0,1}[11]Y^1_{1,1}[11]$, $Y^1_{0,1}[12]Y^1_{2,1}[12]$, $Y^1_{0,1}[15]Y^1_{2,1}[15]$, $Y^1_{0,1}[17]Y^1_{1,1}[17]$, $Y^1_{0,1}[17]Y^1_{2,1}[17]$, $Y^1_{0,1}[18]Y^1_{2,1}[18]$, $Y^1_{0,1}[20]Y^1_{1,1}[20]$, $Y^1_{0,1}[20]Y^1_{2,1}[20]$, $Y^1_{0,1}[21]Y^1_{2,1}[21]$, $Y^1_{0,1}[26]Y^1_{1,1}[26]$, $Y^1_{0,1}[26]Y^1_{2,1}[26]$, $Y^1_{0,1}[29]Y^1_{1,1}[29]$, $Y^1_{0,1}[29]Y^1_{2,1}[29]$, $Y^1_{0,2}[0]Y^1_{2,2}[0]$, $Y^1_{0,2}[8]Y^1_{2,2}[8]$, $Y^1_{0,2}[22]Y^1_{2,2}[22]$, $Y^1_{0,2}[31]Y^1_{2,2}[31]$, $Y^1_{0,3}[4]Y^1_{1,3}[4]$, $Y^1_{0,3}[4]Y^1_{2,3}[4]$, $Y^1_{0,3}[10]Y^1_{1,3}[10]$, $Y^1_{0,3}[10]Y^1_{2,3}[10]$, $Y^1_{0,3}[13]Y^1_{1,3}[13]$, $Y^1_{0,3}[13]Y^1_{2,3}[13]$, $Y^1_{0,3}[19]Y^1_{1,3}[19]$, $Y^1_{0,3}[19]Y^1_{2,3}[19]$, $Y^1_{0,3}[22]Y^1_{1,3}[22]$, $Y^1_{0,3}[22]Y^1_{2,3}[22]$, $Y^1_{0,3}[27]Y^1_{1,3}[27]$, $Y^1_{0,3}[27]Y^1_{2,3}[27]$, $Y^1_{0,3}[28]Y^1_{1,3}[28]$, $Y^1_{0,3}[28]Y^1_{2,3}[28]$, $Y^1_{1,0}[0]Y^1_{2,0}[0]$, $Y^1_{1,0}[6]Y^1_{2,0}[6]$, $Y^1_{1,0}[14]Y^1_{2,0}[14]$, $Y^1_{1,0}[23]Y^1_{2,0}[23]$, $Y^1_{1,0}[29]Y^1_{2,0}[29]$, $Y^1_{1,1}[02]Y^1_{2,1}[02]$, $Y^1_{1,1}[11]Y^1_{2,1}[11]$, $Y^1_{1,1}[19]Y^1_{2,1}[19]$, $Y^1_{1,1}[20]Y^1_{2,1}[20]$, $Y^1_{1,2}[4]Y^1_{2,2}[4]$, $Y^1_{1,2}[7]Y^1_{2,2}[7]$, $Y^1_{1,2}[13]Y^1_{2,2}[13]$, $Y^1_{1,2}[16]Y^1_{2,2}[16]$, $Y^1_{1,2}[21]Y^1_{2,2}[21]$, $Y^1_{1,2}[22]Y^1_{2,2}[22]$, $Y^1_{1,2}[30]Y^1_{2,2}[30]$, $Y^1_{1,2}[31]Y^1_{2,2}[31]$, $Y^1_{1,3}[3]Y^1_{2,3}[3]$, $Y^1_{1,3}[9]Y^1_{2,3}[9]$, $Y^1_{1,3}[12]Y^1_{2,3}[12]$, $Y^1_{1,3}[18]Y^1_{2,3}[18]$, $Y^1_{1,3}[21]Y^1_{2,3}[21]$, $Y^1_{1,3}[26]Y^1_{2,3}[26]$, $Y^1_{1,3}[27]Y^1_{2,3}[27]$
Guessed variables	$Y^1_{0,1}[3]$, $Y^1_{0,1}[17]$, $Y^1_{0,1}[26]$, $Y^1_{0,1}[29]$, $Y^1_{0,3}[04]$, $Y^1_{0,3}[10]$, $Y^1_{0,3}[13]$, $Y^1_{0,3}[19]$, $Y^1_{0,3}[22]$, $Y^1_{0,3}[28]$, $Y^1_{1,0}[6]$, $Y^1_{1,0}[07]$, $Y^1_{1,0}[15]$, $Y^1_{1,0}[21]$, $Y^1_{1,0}[24]$, $Y^1_{1,0}[29]$, $Y^1_{1,0}[30]$, $Y^1_{1,1}[2]$, $Y^1_{1,1}[11]$, $Y^1_{1,1}[20]$, $Y^1_{1,3}[27]$, $Y^1_{2,0}[0]$, $Y^1_{2,0}[14]$, $Y^1_{2,0}[23]$, $Y^1_{2,1}[6]$, $Y^1_{2,1}[9]$, $Y^1_{2,1}[12]$, $Y^1_{2,1}[15]$, $Y^1_{2,1}[18]$, $Y^1_{2,1}[19]$, $Y^1_{2,1}[20]$, $Y^1_{2,1}[21]$, $Y^1_{2,2}[0]$, $Y^1_{2,2}[4]$, $Y^1_{2,2}[7]$, $Y^1_{2,2}[8]$, $Y^1_{2,2}[13]$, $Y^1_{2,2}[16]$, $Y^1_{2,2}[21]$, $Y^1_{2,2}[22]$, $Y^1_{2,2}[30]$, $Y^1_{2,2}[31]$, $Y^1_{2,3}[3]$, $Y^1_{2,3}[9]$, $Y^1_{2,3}[12]$, $Y^1_{2,3}[18]$, $Y^1_{2,3}[21]$, $Y^1_{2,3}[26]$, $Y^1_{2,3}[27]$

is linearly dependent on the variables in X^1_2; i.e., the internal part of the initial state needs to satisfy 1 linear condition, so $d_1 = 1$. The random space size is unaffected by the linearization technique and is $2^{d_r} = 2^{128-3-(29-1)-15} = 2^{82}$, satisfying $2^{d_r} > 2^{d_1}$.

The Complexity of Solving Time. Now, we need to solve 92 linear equations with 125 variables. We reduce the number of bit operations by precalculating 77 rows and updating only 15 rows in each turn of guessed variables from Table 11. The precomputation complexity is

$$[125 + (125 + 124) + \cdots + (125 + 124 + \cdots + 50))] = 292600$$

After precomputation, the calculation of each turn becomes:

$$[(125 + 124 \cdots + 49) + \cdots + (125 + 124 + \cdots + 35)] = 105070$$

The total number of bit operations for a 4-round Xoodyak-Xof is given by $4 \times (1024 + 768 + 32) = 7296$. At each turn, we perform Gaussian Elimination on the linear system,

expecting to obtain 2^{33} solutions. For each solution, the quadratic structure allows us to generate 3 linear equations from the hash value. The solving complexity is measured as the ratio of the bit operations needed for Gaussian Elimination to the total number of bit operations of 4-round Xoodyak-Xof, which is $\frac{\frac{292600}{2^{15}}+105070}{2^{33}}/7296 \ll 1$. As for the complexity of solving 3 linear equations, this part can be neglected since the number of bit operations is only 8.

Final Complexity. In summary, by using a single quartic structure and applying a linearization strategy, we improve the preimage attack on 4-round Xoodyak-Xof with the final complexity of $2^{126.245}$.

5 Conclusion

In this paper, we propose a preimage attack framework for Xoodyak-Xof. We establish canonical constraints for the propagation rules of linear variables in both the χ function and the linear functions, and develop an automatic structure search model for Xoodyak-Xof based on MILP. Combined with the weight update algorithm, we search for linear structures that improve preimage attacks on 2-round and 3-round Xoodyak-Xof, and we present the first preimage attack result on 4-round Xoodyak-Xof. Although a canonical modeling of the χ function is provided in this paper, the affine function of χ, eg, the nonlinear function in Ascon, has not yet been canonically modeled. Moreover, applying nonlinear structures to preimage attacks remains an open problem worth considering.

References

1. Bertoni, G., Peeters, M., Van Assche, G., et al.: The keccak reference. http://keccak.noekeon.org (2011)
2. Daemen, J., Hoffert, S., Assche, G.V., Keer, R.V.: The design of xoodoo and xoofff. IACR Trans. Symmetric Cryptol. **2018**(4), 1–38 (2018), https://doi.org/10.13154/tosc.v2018.i4.1-38
3. Daemen, J., Hoffert, S., Peeters, M., Assche, G.V., Keer, R.V.: Xoodyak, a lightweight cryptographic scheme. IACR Trans. Symmetric Cryptol. **2020**(S1), 60–87 (2020), https://doi.org/10.13154/tosc.v2020.iS1.60-87
4. Dong, X., Zhao, B., Qin, L., Hou, Q., Zhang, S., Wang, X.: Generic mitm attack frameworks on sponge constructions. In: Reyzin, L., Stebila, D. (eds.) Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part IV. Lecture Notes in Computer Science, vol. 14923, pp. 3–37. Springer (2024), https://doi.org/10.1007/978-3-031-68385-5_1
5. Fu, Q., Lv, Y., Liu, Z., Li, Y., Song, L., Weng, J.: Preimage attacks on xoodyak and gaston based on algebraic strategies. IEEE Internet Things J. **11**(19), 31368–31384 (2024), https://doi.org/10.1109/JIOT.2024.3420177
6. Guo, J., Liu, M., Song, L.: Linear structures: Applications to cryptanalysis of round-reduced keccak. In: Cheon, J.H., Takagi, T. (eds.) Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10031, pp. 249–274 (2016), https://doi.org/10.1007/978-3-662-53887-6_9

7. He, L., Lin, X., Yu, H.: Improved preimage attacks on round-reduced keccak-384/512. Des. Codes Cryptogr. **94**(3), 55 (2026), https://doi.org/10.1007/s10623-025-01780-z
8. Lefevre, C., Mennink, B.: Tight preimage resistance of the sponge construction. In: Dodis, Y., Shrimpton, T. (eds.) Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part IV. Lecture Notes in Computer Science, vol. 13510, pp. 185–204. Springer (2022), https://doi.org/10.1007/978-3-031-15985-5_7
9. Li, H., He, L., Chen, S., Guo, J., Qiu, W.: Automatic preimage attack framework on ascon using a linearize-and-guess approach. IACR Trans. Symmetric Cryptol. **2023**(3), 74–100 (2023), https://doi.org/10.46586/tosc.v2023.i3.74-100
10. Li, T., Sun, Y.: Preimage attacks on round-reduced keccak-224/256 via an allocating approach. In: Ishai, Y., Rijmen, V. (eds.) Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part III. Lecture Notes in Computer Science, vol. 11478, pp. 556–584. Springer (2019), https://doi.org/10.1007/978-3-030-17659-4_19
11. Lin, X., He, L., Yu, H.: Improved preimage attacks on 3-round keccak-224/256. IACR Trans. Symmetric Cryptol. **2021**(3), 84–101 (2021), https://doi.org/10.46586/tosc.v2021.i3.84-101
12. Liu, F., Isobe, T., Meier, W., Yang, Z.: Algebraic attacks on round-reduced keccak. In: Baek, J., Ruj, S. (eds.) Information Security and Privacy - 26th Australasian Conference, ACISP 2021, Virtual Event, December 1-3, 2021, Proceedings. Lecture Notes in Computer Science, vol. 13083, pp. 91–110. Springer (2021), https://doi.org/10.1007/978-3-030-90567-5_5
13. Qin, L., Hua, J., Dong, X., Yan, H., Wang, X.: Meet-in-the-middle preimage attacks on sponge-based hashing. In: Hazay, C., Stam, M. (eds.) Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part IV. Lecture Notes in Computer Science, vol. 14007, pp. 158–188. Springer (2023), https://doi.org/10.1007/978-3-031-30634-1_6
14. Qin, L., Zhao, B., Hou, Q., Jia, K., Wang, G., Dong, X.: Linear cancellations in the mitm attacks on sponge functions. IACR Trans. Symmetric Cryptol. **2025**(3), 633–692 (2025), https://doi.org/10.46586/tosc.v2025.i3.633-692

Appendixes

Appendix A

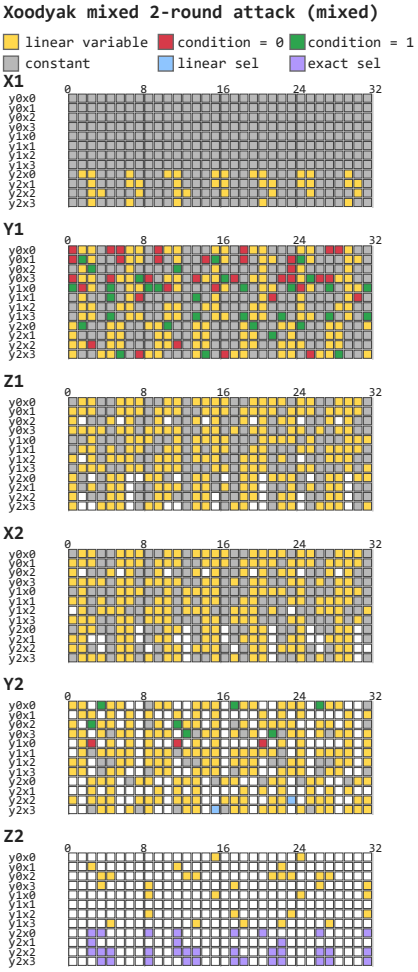

Fig. 2: structure 1

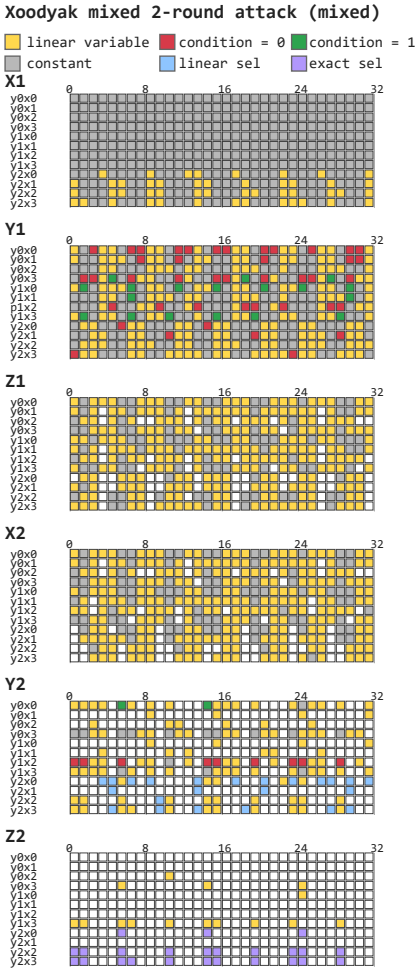

Fig. 3: structure 2

Appendix B

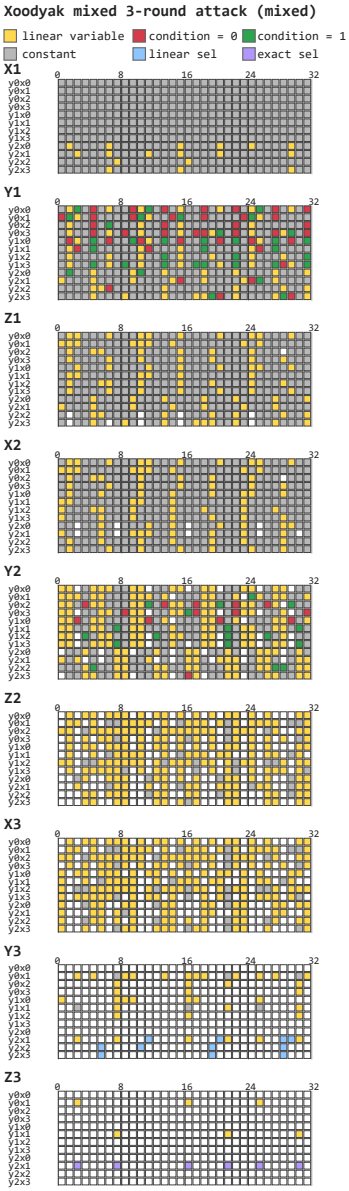

Fig. 4: structure 1

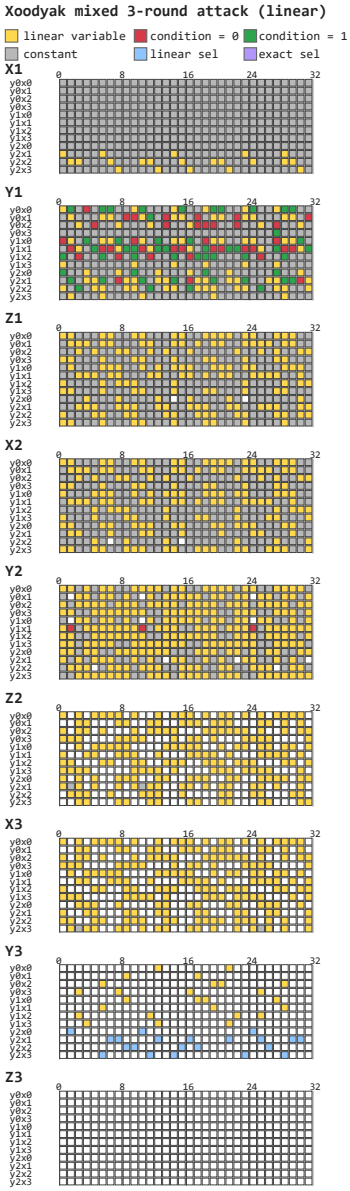

Fig. 5: structure 5

Appendix C

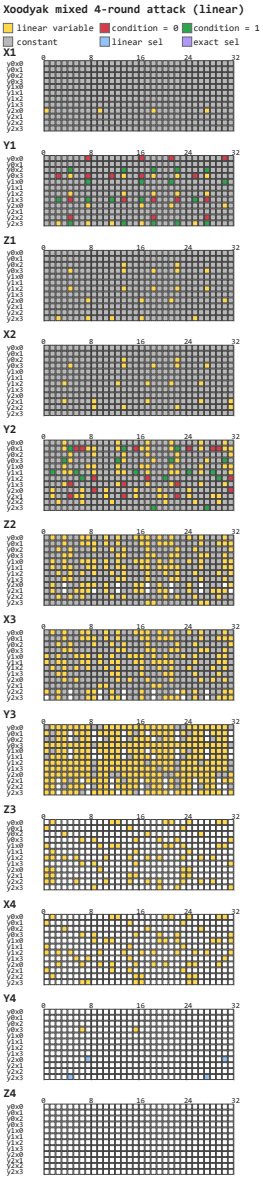

Fig. 6: structure 1

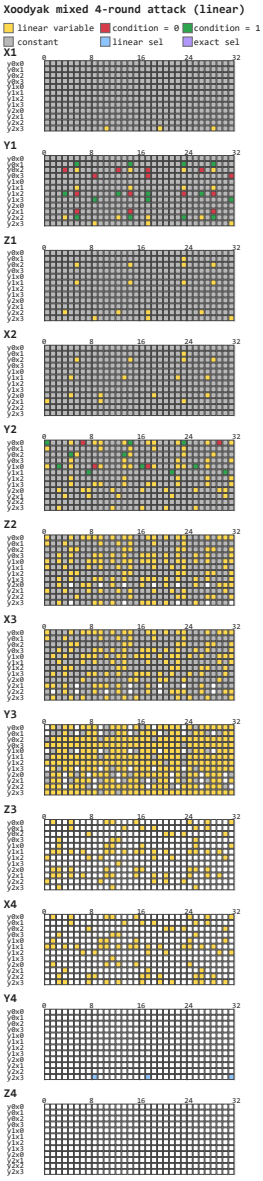

Fig. 7: structure 2