Xoodyak - Analysis of the Xoodoo permutation

R Prathamesh

Indian Institute of Technology, Bhilai, India, rprathamesh@iitbhilai.ac.in

Abstract. In this paper I analyse the Xoodoo permutation, which is used by Xoodyak internally. More specifically, I analyse the non-linear operation of the permutation and perform a zero-sum attack upto 4 rounds of the permutation. I also implement the permutation in Verilog and analyse the area taken. I also provide the constraints required to model the permutation as an MILP problem. Finally, I explain a known attack on the permutation.

Keywords: Xoodyak · Xoodoo · Zero-Sum Distinguisher · MILP · Linearization

1 Introduction

Xoodyak[1] is a versatile cryptographic primitive that can be used for hashing, encryption, MAC computation and authenticated encryption. Internally, it uses the Xoodoo[2] permutation. The Xoodoo permutation is an iterated permutation with a 384-bit state whose design approach is inspired by the Keccak-p permutation while it is dimensioned like Gimli for efficiency on low-end processors. The structure of the Xoodoo state consists of 3 horizontal planes with each plane containing 4 parallel 32-bit lanes.
The Xoodoo permutation is described in Section 2 followed by cryptanalysis in Section 3.

2 Specifications

In my analysis, I am mainly interested in the Xoodoo permutation. In Section 2.1, I specify the notations used and in Section 2.2, I give a detailed description of the Xoodoo permutation as given in [2].

2.1 Notation

The following notations are used extensively to describe the permutation.

- $A_y \longleftarrow$ Plane y of state A
- $A_y + A_{y'} \longleftarrow$ Biwise sum (XOR) of planes A_y and $A_{y'}$
- $A_y.A_{y'} \longleftarrow$ Bitwise product (AND) of planes A_y and $A_{y'}$
- $\overline{A_y} \longleftarrow$ Bitwise complement of plane A_y
- $A_y \lll (t, v) \longleftarrow$ Cyclic shift of plane A_y : (x, z) goes to $(x + t, z + v)$

Each round of the permutation has 5 operations that are denoted by the following symbols

- $\theta \longleftarrow$ Mixing layer - theta
- $\rho_{west} \longleftarrow$ Plane shifting layer - rho-west

- $\iota \longleftarrow$ Round constant addition layer - iota
- $\chi \longleftarrow$ Non linear layer - chi
- $\rho_{east} \longleftarrow$ Plane shifting layer - rho-east

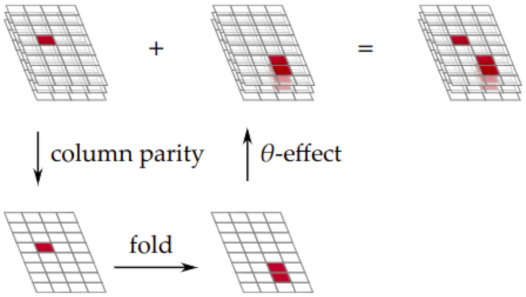
2.2 Xoodoo[n_r]

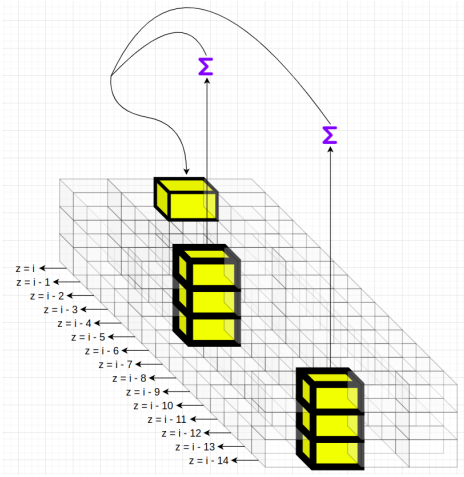
The rounds of Xoodoo are numbered from $1 - n_r$ to 0 where n_r is the number of rounds for which Xoodoo will be run. Xoodyak uses 12 rounds of the Xoodoo permutation, so the rounds are numbered from -11 to 0. Each round of the Xoodoo permutation has 5 operations as described below.

2.2.1 Mixing Layer θ

$$P \longleftarrow A_0 + A_1 + A_2$$
$$E \longleftarrow P \lll (1,5) + P \lll (1,14)$$
$$A_y \longleftarrow A_y + E \text{ for } y \in \{0,1,2\}$$

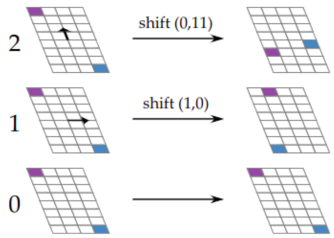
The following images describe the action of the θ operation on the state





2.2.2 Plane shifting ρ_{west}

$$A_1 \longleftarrow A_1 \lll (1,0)$$
$$A_2 \longleftarrow A_2 \lll (0,11)$$



Action of ρ_{west} on the state

2.2.3 Round constant addition ι

$$A_0 \longleftarrow A_0 + C_i$$

The round constants for 12 rounds is given in the table below

i	c_i	i	c_i	i	c_i	i	c_i
-11	$0x00000058$	-8	$0x000000D0$	-5	$0x00000060$	-2	$0x000000F0$
-10	$0x00000038$	-7	$0x00000120$	-4	$0x0000002C$	-1	$0x000001A0$
-9	$0x000003C0$	-6	$0x00000014$	-3	$0x00000380$	0	$0x00000012$

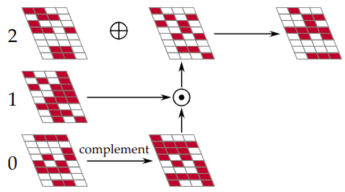
2.2.4 Non linear layer χ

$$B_0 \longleftarrow \overline{A_1}.A_2$$

$$B_1 \longleftarrow \overline{A_2}.A_0$$

$$B_2 \longleftarrow \overline{A_0}.A_1$$

$$A_y \longleftarrow A_y + B_y \text{ for } y \in \{0, 1, 2\}$$



Action of χ on the state

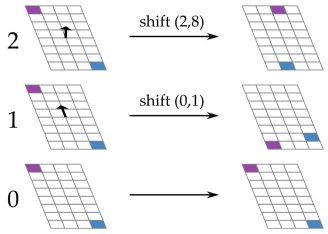
It is easy to notice that χ acts on each column seperately. Thus, we can capture the action of χ on each column in the form of the following S-Box

x	0	1	2	3	4	5	6	7
$S(x)$	0	3	6	1	5	4	2	7

2.2.5 Plane shifting ρ_{east}

$$A_1 \longleftarrow A_1 \lll (0, 1)$$

$$A_2 \longleftarrow A_2 \lll (2, 8)$$



Action of ρ_{east} on the state

3 Cryptanalysis

As part of the cryptanalysis of the permutation, I first analyse the non linear operation of each round of the permutation in Section 3.1 followed by the verification of the zero - sum property in Section 3.2

3.1 Non linear layer Analysis

As seen in Section 2.2.4, the non linear layer χ acts on each of the 4 x 32 = 128 columns independently, so the 3-bit S-Box that captures the action of χ is given below

x	0	1	2	3	4	5	6	7
$S(x)$	0	3	6	1	5	4	2	7

It is easy to see that the S-Box is involutive

3.1.1 DDT

The Difference Distribution Table (DDT) of the S-Box which is used extensively in differential cryptanalysis is given below

	0	1	2	3	4	5	6	7
0	8	0	0	0	0	0	0	0
1	0	2	0	2	0	2	0	2
2	0	0	2	2	0	0	2	2
3	0	2	2	0	0	2	2	0
4	0	0	0	0	2	2	2	2
5	0	2	0	2	2	0	2	0
6	0	0	2	2	2	2	0	0
7	0	2	2	0	2	0	0	2

3.1.2 LAT

The Linear Approximation Table (LAT) of the S-Box which is used extensively in linear cryptanalysis is given below

	0	1	2	3	4	5	6	7
0	4	0	0	0	0	0	0	0
1	0	2	0	-2	0	2	0	2
2	0	0	2	2	0	0	-2	2
3	0	-2	2	0	0	2	2	0
4	0	0	0	0	2	-2	2	2
5	0	2	0	2	-2	0	2	0
6	0	0	-2	2	2	2	0	0
7	0	2	2	0	2	0	0	-2

3.1.3 1-1 DDT

The 1-1 Difference Distribution Table is the Difference Distribution Table restricted to input and output differences of Hamming Weight 1 only. This table is used to track the propagation of high probability differential trails across different rounds.

	bit 0	bit 1	bit 2
bit 0	2	0	0
bit 1	0	2	0
bit 2	0	0	2

3.1.4 1-1 LAT

The 1-1 Linear Approximation Table is the Linear Approximation Table restricted to input and output masks of Hamming Weight 1 only. This table is used to track the propagation of high probability linear trails across different rounds.

	bit 0	bit 1	bit 2
bit 0	2	0	0
bit 1	0	2	0
bit 2	0	0	2

3.1.5 BOGI

The concept of BOGI[3] or Bad Output must go to Good Input describes the action of the linear layer. However, the availability of such a linear layer also depends on the S-Box, which is why I analyse it here. When we try to prevent the possibility of a high probability trail in a construction, we try to disallow single bit to single bit transitions across the S-Box. By disallowing such a transition, we usually spread the trail into multiple branches in the following rounds.
Thus, the term 'Good Input' refers to a single bit input difference to the S-Box which does not allow any single bit output difference. Similarly, a 'Bad Output' refers to a single bit output difference that has the possibility of arriving from a single bit input difference.
It can be easily noticed from the 1-1 DDT that none of the single bit input differences are 'Good Inputs' and none of the output differences are 'Good Outputs'. Thus, the concept of BOGI cannot be applied here.

3.1.6 BCT

The Boomerang Connectivity Table (BCT) which is used to look at how compatible two trails are to be part of a Boomerang attack is shown below

	0	1	2	3	4	5	6	7
0	8	8	8	8	8	8	8	8
1	8	2	0	2	0	2	0	2
2	8	0	2	2	0	0	2	2
3	8	2	2	0	0	2	2	0
4	8	0	0	0	2	2	2	2
5	8	2	0	2	2	0	2	0
6	8	0	2	2	2	2	0	0
7	8	2	2	0	2	0	0	2

3.1.7 BDT

The Boomerang Difference Table is an improvement over the Boomerang Connectivity Table. One of the deficiencies of the BCT was that it did not account for the propagation of the difference itself. There are cases where the BCT entry is non zero but the DDT entry is zero, thereby preventing the differences from being compatible. Thus, the BDT accounts for both of these and is a mapping with 3 inputs. Thus, to represent it we need to use $2^3 = 8$ tables each having dimensions 8 x 8. The BDT along with some observations are in Appendix A

3.2 Zero - Sum Property

The degree of the algebraic form of the state may increase only due to the non linear layers and the only non linear layer here is χ whose degree is 2. It is known that the zero - sum

property is theoretically guaranteed if we compute the $d + 1^{th}$ derivative of a function whose maximum degree is d. Thus, the 3^{rd} order derivative should give guaranteed zero - sum for 1 round of Xoodoo. Similarly, the 5^{th} order derivative should also give guaranteed zero - sum for 2 rounds of Xoodoo. For 3 and 4 rounds of Xoodoo, 9^{th} and 17^{th} order derivatives should give guaranteed zero - sum.
I have verified the above property and the results are summarised in the table below. The $-$ in the table signifies that zero - sum has been reached theoretically for a lower degree derivative and hence not required to compute.

Degree\Rounds	1	2	3	4
17	-	-	-	100*
16	-	-	-	100*
15	-	-	-	100*
14	-	-	-	100*
13	-	-	-	60*
12	-	-	-	30*
11	-	-	-	24
10	-	-	-	20
9	-	-	100	18
8	-	-	100	15
7	-	-	97	8
6	-	-	78.87	8
5	-	100	28.5	4
4	-	99	3.31	2
3	100	73.12	1.78	2
2	86.03	0.59	0.46	1
1	0	0	0	0

* : Results may be unreliable as less than 100 experiments were run due to time constraints

4 Automated Tools

In this section, I describe the constraints needed to model the Xoodoo permutation using MILP.
The challenge in modelling each round of the Xoodoo permutation is mainly in the θ operation and the non linear layer χ. This is because 2 out of the other 3 operations are a simple shifting of the planes which can be easily captured in MILP by simply equating variables. The 3^{rd} operation is the addition of a round constant, which is irrelevant for a MILP model, because MILP modelling is mainly for finding differential trails and a round constant addition does not affect the differential trail similar to the key annihilation property.
As described in Section 2.2.1, the θ operation does the following

$$P \longleftarrow A_0 + A_1 + A_2$$

$$E \longleftarrow P \lll (1,5) + P \lll (1,14)$$

$$A_y \longleftarrow A_y + E \text{ for } y \in \{0,1,2\}$$

P and E are computed column-wise and there are a total of 128 columns in the state. Thus, to compute each bit of P and E, we need to repeat the constraints 128 times. For

each bit of P, we need 2 XOR operations. If we store the value of P, then we can compute each bit of E with just 1 more XOR operation. Thus, we can compute each bit of E with 3 XOR operations. That is a total of 3 x 128 = 384 XOR operations. If we now store the value of E, then each bit of the state A can be computed with one more XOR operation, thus giving a total of $384 + 384 = 768$ XOR operations in total.
Each XOR operation can be modelled using the following 5 inequalities. So if we have an XOR operation $a \oplus b = c$, then the constraints needed are

$$a + b + c \geq 2d$$
$$d \geq a$$
$$d \geq b$$
$$d \geq c$$
$$a + b + c \leq 2$$

Therefore, the total number of inequalities needed for the θ operation is 768 x 5 = 3840

As described in Section 2.2.2, the ρ_{west} operation does the following

$$A_1 \longleftarrow A_1 \lll (1, 0)$$
$$A_2 \longleftarrow A_2 \lll (0, 11)$$

Since this operation is just a shifting of bits, we can capture this in MILP by simply equating the variable of the next round with the appropriate variables of the current round. Thus, the number of constraints required will be 384 equalities. The next operation ι is not discussed here due to the key annihilation-like property.

As described in Section 2.2.4, the χ operation does the following

$$B_0 \longleftarrow \overline{A_1}.A_2$$
$$B_1 \longleftarrow \overline{A_2}.A_0$$
$$B_2 \longleftarrow \overline{A_0}.A_1$$
$$A_y \longleftarrow A_y + B_y \text{ for } y \in \{0, 1, 2\}$$

Rather than modelling the AND operation here, I decided to consider it as a 3-bit S-Box and use the logical condition modelling [4] technique to capture the action of the non linear layer. I identified the following truncated transitions for the S-Box and the corresponding constraint to be added is given along with it

$001 \longrightarrow **1$	$x_2 + x_1 - x_0 + y_0 \geq 0$
$010 \longrightarrow *1*$	$x_2 - x_1 + x_0 + y_1 \geq 0$
$100 \longrightarrow 1**$	$-x_2 + x_1 + x_0 + y_0 \geq 0$
$**1 \longrightarrow 001$	$x_0 + y_2 + y_1 - y_0 \geq 0$
$*1* \longrightarrow 010$	$x_1 + y_2 - y_1 + y_0 \geq 0$
$1** \longrightarrow 100$	$x_2 - y_2 + y_1 + y_0 \geq 0$

Each S-Box acts on a column and there are 128 columns so these 6 constraints will apply to each of the columns. So the total number of constraints needed will be 128 x 6 = 768

As described in Section 2.2.5, the ρ_{east} operation does the following

$$A_1 \longleftarrow A_1 \lll (0, 1)$$
$$A_2 \longleftarrow A_2 \lll (2, 8)$$

Again, 384 equalities will be enough.

Thus, the total number of constraints that we need to model one round of the Xoodoo permutation is $3840 + 384 + 768 + 384 = 5376$.

5 Known Attacks on Xoodoo

In this section, I describe a known attack on Xoodoo which is a practical zero sum distinguisher for 12 rounds of Xoodoo. This attack was shown in [5]. However, to appreciate this attack, we need to look into the generic attacks on 12 rounds of Xoodoo.

The algebraic degree of 1 round of Xoodoo is 2. Hence, for 12 rounds, the maximum degree will be $2^{12} = 4096$. However, since there are only 384 bits in the state, the maximum degree can only be 384. Therefore, for 12 rounds, the 385^{th} order derivative will give guaranteed zero - sum. This is clearly the most naive attack possible.

An improvement that can be done here is to start from the middle of the 12 rounds. The improvement here comes from the fact that there are only 6 rounds to go forward and backward. The non linear layer χ is involutive, hence the algebraic degree of the backward direction is also 2 degrees per round. Thus, the algebraic degree at the input of the state and the output of the state are both equal to $2^6 = 64$. So, the 65^{th} order derivative will give guaranteed zero - sum. This is a much better attack, however, it is still not that practical.

The attack that is shown in [5] implements the above attack while also using the concept of linearization. Linearization is a technique in which we choose the variables for the derivative smartly in such a way that the algebraic degree remains 1. This is possible when one of the bits in an AND operation is a constant in the derivative. In the paper [5], they use a MILP model to return the best possible bits to ensure linearization in both directions.

Thus, the maximum algebraic degree of 6 rounds of Xoodoo will now be 32 and the same will be for its inverse. Thus, if we take the 33^{rd} order derivative, we will get guaranteed zerosum at the outout of the state. Similarly the same can be said for the inverse, i.e. the maximum algebraic degree of the input state will be 32 as well. So we can compute the 33^{rd} order derivative on both sides to get a practical attack with a time complexity of $O(2^{33})$ which is quite practical.

A Boomerang Difference Table

	0	1	2	3	4	5	6	7
0	8	8	8	8	8	8	8	8
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0

For $(\Delta_0, \Delta_1 = 0, \nabla_0)$

	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	2	2	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	2	0	0	2	0	0	0	0
4	0	0	0	0	0	0	0	0
5	2	0	0	0	0	2	0	0
6	0	0	0	0	0	0	0	0
7	2	0	0	0	0	0	0	2

For $(\Delta_0, \Delta_1 = 1, \nabla_0)$

	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
2	2	0	2	0	0	0	0	0
3	2	0	0	2	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	2	0	0	0	0	0	2	0
7	2	0	0	0	0	0	0	2

For $(\Delta_0, \Delta_1 = 2, \nabla_0)$

	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	2	2	0	0	0	0	0	0
2	2	0	2	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	2	0	0	0	0	2	0	0
6	2	0	0	0	0	0	2	0
7	0	0	0	0	0	0	0	0

For $(\Delta_0, \Delta_1 = 3, \nabla_0)$

	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	2	0	0	0	2	0	0	0
5	2	0	0	0	0	2	0	0
6	2	0	0	0	0	0	2	0
7	2	0	0	0	0	0	0	2

For $(\Delta_0, \Delta_1 = 4, \nabla_0)$

	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	2	2	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	2	0	0	2	0	0	0	0
4	2	0	0	0	2	0	0	0
5	0	0	0	0	0	0	0	0
6	2	0	0	0	0	0	2	0
7	0	0	0	0	0	0	0	0

For $(\Delta_0, \Delta_1 = 5, \nabla_0)$

	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
2	2	0	2	0	0	0	0	0
3	2	0	0	2	0	0	0	0
4	2	0	0	0	2	0	0	0
5	2	0	0	0	0	2	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0

For $(\Delta_0, \Delta_1 = 6, \nabla_0)$

	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	2	2	0	0	0	0	0	0
2	2	0	2	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	2	0	0	0	2	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	2	0	0	0	0	0	0	2

For $(\Delta_0, \Delta_1 = 7, \nabla_0)$

The largest value in the BDT is the trivial maximum value, i.e. 8 which occurs when $\Delta_0 = \Delta_1 = 0$. The next largest value is the same as any other non-zero value which is 2 and it occurs 8 times for each value of Δ_i where $i = \{1, 2, \ldots, 7\}$. Out of the 8 times that 2 occurs 4 of them occur for $\nabla_0 = 0$ while the other 4 occur for some specific cases where $\Delta_0 = \nabla_0$.

References

[1] J. Daemen, S. Hoffert, M. Peeters, G. Van Assche, and R. Van Keer, "Xoodyak, a lightweight cryptographic scheme," *IACR Transactions on Symmetric Cryptology*, vol. 2020, no. S1, pp. 60–87, 2020.

[2] J. Daemen, S. Hoffert, G. V. Assche, and R. V. Keer, "The design of Xoodoo and Xoofff," *IACR Transactions on Symmetric Cryptology*, vol. 2018, no. 4, pp. 1–38, 2018.

[3] S. Banik, S. K. Pandey, T. Peyrin, Y. Sasaki, S. M. Sim, and Y. Todo, "GIFT: A small present - towards reaching the limit of lightweight encryption," in *Cryptographic Hardware and Embedded Systems – CHES 2017* (W. Fischer and N. Homma, eds.), vol. 10529 of *Lecture Notes in Computer Science*, (Taipei, Taiwan), pp. 321–345, Springer, Heidelberg, Germany, Sept. 25–28, 2017.

[4] S. Sun, L. Hu, P. Wang, K. Qiao, X. Ma, and L. Song, "Automatic security evaluation and (related-key) differential characteristic search: Application to SIMON, PRESENT, LBlock, DES(L) and other bit-oriented block ciphers," in *Advances in Cryptology – ASIACRYPT 2014, Part I* (P. Sarkar and T. Iwata, eds.), vol. 8873 of *Lecture Notes in Computer Science*, (Kaoshiung, Taiwan, R.O.C.), pp. 158–178, Springer, Heidelberg, Germany, Dec. 7–11, 2014.

[5] F. Liu, T. Isobe, W. Meier, and Z. Yang, "Algebraic attacks on round-reduced keccak/xoodoo." Cryptology ePrint Archive, Report 2020/346, 2020. https://eprint.iacr.org/2020/346.