Pyjamask-128 - Implementation and Analysis

NBS Bharath Kumar Reddy[1,2], Vetsa Tarun[1,3] and GM Imdadulla Khan[1,4]

[1] Indian Institute of Technology, Bhilai, India
[2] nbsbharath@iitbhilai.ac.in
[3] vetsatarun@iitbhilai.ac.in
[4] gmimdadulla@iitbhilai.ac.in

Abstract. In this paper we discuss the algorithm of Pyjamask-128 and Differential and Integral Cryptanalysis on it.

Keywords: Pyjamask-128 · Differential Cryptanalysis · Integral Cryptanalysis

1 Introduction

This document specifies Pyjamask, an authenticated encryption with associated data (AEAD) scheme based on a new block cipher (BC) called Pyjamask and on the AEAD operating mode OCB.

The cipher rely on a Substitution-Permutation Network (SPN) structure, which applies a key-dependent round function multiple times to convert the original plaintext into the ciphertext. An iterated key schedule algorithm is used to extract each round key from the secret key.

It consists of 14 rounds for encryption of each block.

2 Round functions

In this section, we go over how the data is represented inside the cypher.

2.1 Data

The internal states of the ciphers , which are seen as matrices of bits with 4 rows and 32 columns, are initially loaded with the plaintext.

0	1	2	3	4	5	6	....	29	30	31
32	33	34	35	36	37	38	....	61	62	63
64	65	66	67	68	69	70	....	93	94	95
96	97	98	99	100	101	102	....	125	126	127

Figure 1: internal state of 4 rows and 32 columns

2.2 Rounds

The plaintext is converted into the final output of ciphertext over the course of 14 rounds, each of which consists of the following three steps: AddRoundKey, SubBytes, and MixRows of the input plaintext or the output from the previous round.

AddRoundKey

Bitwise addition of the first n bits of the key state (define below) into the internal state

Key Addition

0	1	2	3	4	5	6	....	29	30	31
32	33	34	35	36	37	38	....	61	62	63
64	65	66	67	68	69	70	....	93	94	95
96	97	98	99	100	101	102	....	125	126	127

$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$	....	$\oplus$	$\oplus$	$\oplus$
$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$	....	$\oplus$	$\oplus$	$\oplus$
$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$	....	$\oplus$	$\oplus$	$\oplus$
$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$	....	$\oplus$	$\oplus$	$\oplus$

For better understanding we can represent the state as shown below
0' = 0 $\oplus$ k[0]

0'	1'	2'	3'	4'	5'	6'	....	29'	30'	31'
32'	33'	34'	35'	36'	37'	38'	....	61'	62'	63'
64'	65'	66'	67'	68'	69'	70'	....	93'	94'	95'
96'	97'	98'	99'	100'	101'	102'	....	125'	126'	127'

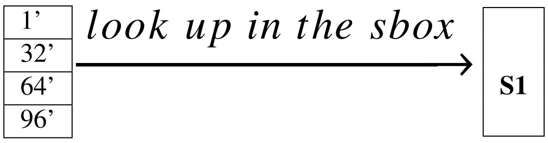
SubBytes

The objective of the substitute byte (S-box) is to confuse the data that needs to be encrypted. The S-box is a 4 by 4 matrix box that is indexed in a row and column pattern and holds a total of 16 bytes of hexadecimal data.

The same Sbox is applied to each of the 32 columns of the internal state. For Pyjamask-128, the Sbox is shown below.

x	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**a**	**b**	**c**	**d**	**e**	**f**
S[x]	**2**	**d**	**3**	**9**	**7**	**b**	**a**	**6**	**e**	**0**	**f**	**4**	**8**	**5**	**1**	**c**

The substitution happens on each column(4 bits each), each column is replaced with new column(S1)



Similarly for the rest of the columns also, now the state would look like the figure below.

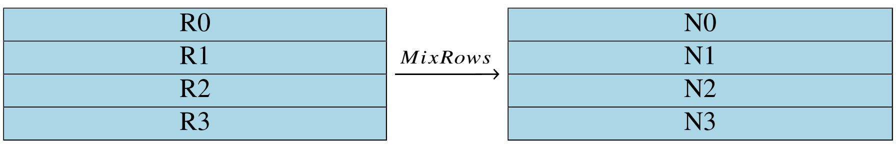
s1	s2	s3	s4	s5	s6	s7	....	s30	s31	s32

MixRows

In MixRows, each row Ri of the internal state, where I = 0, 1, 2, and 3, is viewed as a column vector with 32 elements, and is substituted by Mi.Ri . The matrices Mi are the following 32 X 32 constant circulant binary matrices.

M0 = cir([1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0])
M1 = cir([0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1])
M2 = cir([0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1])
M3 = cir([0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1])

Now in this round function we basically multiply the row bits (R0, 32X1 as a column) will be multiplied with a certain circulant matrix(32 X 32) to give one column matrix(32 X 1) which is replaced by R0.
Lets say R0 X M0 is equal to N0. so after doing this for all the four rows the state would be as shown below.



2.3 Inverse Round Functions

Similar to forward round functions, inverse round functions apply the inverse of the elementary transformations in the opposite order.

- **invMixRows:** Each row R_i of the internal state, and $i \in \{0, 1, 2, 3\}$ is seen as a column vector of 32 elements and is replaced by $M^{-1} \cdot R_i$
- **invSubBytes:** The inverse Sbox S_4^{-1} is applied to all 32 columns of the internal state.
- **invAddRoundKey:** The first n bits of the key state is XORed to the internal state

Before applying the first round, cipher text is xored with last key produced in key scheduling. Then, apply the inverse round function for the remaining 14 rounds.

2.4 Key Schedule

The master key has 128 bits in it. The 128-bit key state is first loaded in the same sequence as the internal state. Then, to produce 14 additional keys, the 128-bit key state goes through three simple changes.

- **MixColumns:** Each 4-bit column C_i of the key state is seen as a vector of four element over GF(2) and is replaced by $M \cdot C_i$, where the matrix M is defined by:

$$M = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

  Column diffusion

M1	M2	M3	M4	M5	M6	M7	....	M30	M31	M32

- **MixAndRotateRows:** The first row vector R0 of the key state is replaced by Mk*R0 and is seen as a 32-element vector over F2.
  where Mk is circular matrix;

  Mk= cir([1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0]).

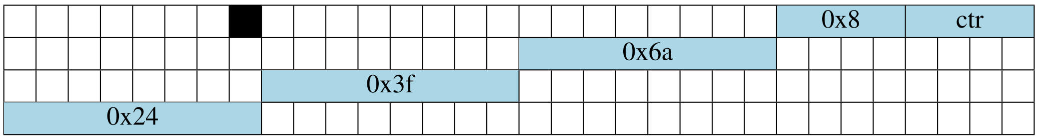
  The second row R1, third row R2, and fourth row R3 are left-rotated by 8, 15,18 positions. Namely they are replaced by $R1 \lll 8$, $R2 \lll 15$, and $R3 \lll 18$ respectively.

M_k
$\lll 8$
$\lll 15$
$\lll 18$

- **AddConstant:** The final step involves defining and dividing a 32-bit round constant into four separate bytes, which are bitwise added to various regions of the rows of the key state. The remaining 28 bits are fixed to a constant represented on Figure 3 by the hexadecimal value 0x243f6a8, and the final four bits of the constant which increases from 0 to 13 for all the 14 rounds.

  CONSTANT = [0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0]

  Then, this constant's most significant byte (MSB) is XORed to the MSB of the fourth row R3, its second MSB is XORed to the MSB of the third row R2, its third MSB is XORed to the MSB of the second row R1, and finally its least significant byte (LSB) is XORed to the LSB of the first row R0.



3 DDT and LAT

3.1 Analysis of the S-box:

The s-box has been selected in order to obtain optimal linear and differential properties.
In Pyjamask-128, simple operations are used to build the 4-bit sbox i.e.,
$(a,b,c,d) \longrightarrow (\,b,c,d \oplus (a \wedge b))$.
By simply iterating these operations we obtain Sboxes with optimal differential and linear properties.
. Comparing the Values of Differential Uniformity and Differential branch number with other groups:

Group name	Cipher Name	SBox size	Differential Uniformity	Differential Branch number
Brain fog	Pyjamask-128	4-bit	4	2
gugu gaga	Midori	4-bit	4	2
SPL Encrypted	GIFT	4-bit	6	2
Hope We 3 SDVV	SERPENT	4-bit	4	3
-../ ./ cipher	Prince	4-bit	4	2
Tech Heist 3.0	Pride	4-bit	4	2
Rook	Ascon	5-bit	8	3
Three Amigos	Klein	4-bit	4	2
Decryptor	PHOTON-beetle	4-bit	4	3
cryptoducks	LED	4-bit	4	3
ping 999+	Elephant	4-bit	4	3
Kryptonian	Wage	8-bit	8	2
cipherbytes	Aria	8-bit	4	2
C14	Primates APE	5-bit	2	2
Bitbees	Skinny	4-bit	2	2
Bash Ciphers	Print	3-bit	2	2
SHA 69	Mysterion	4-bit	4	2
Hex Brains	Rectangle	4-bit	4	2

DDT	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	16	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
1	-	-	-	-	-	-	-	-	-	-	2	2	4	4	2	2
2	-	4	-	-	4	-	-	-	-	4	-	-	-	4	-	-
3	-	4	-	-	4	-	-	-	-	-	2	2	-	-	2	2
4	-	-	-	-	-	4	4	-	2	2	-	-	-	-	2	2
5	-	-	-	4	-	4	-	-	2	2	2	2	-	-	-	-
6	-	2	2	-	2	-	-	2	2	-	-	2	2	-	-	-
7	-	2	2	-	2	-	-	2	2	-	2	-	2	-	2	-
8	-	-	-	-	-	-	-	-	-	-	2	2	4	4	2	2
9	-	-	4	4	-	-	4	4	-	-	-	-	-	-	-	-
a	-	-	2	2	-	-	2	2	-	4	-	-	-	4	-	-
b	-	-	2	2	-	-	2	2	-	-	2	2	-	-	2	2
c	-	-	4	-	-	4	-	-	2	2	2	2	-	-	-	-
d	-	-	-	-	-	4	-	4	2	2	-	-	-	-	2	2
e	-	2	-	2	2	-	2	-	2	-	-	2	2	-	-	2
f	-	2	-	2	2	-	2	-	2	-	2	-	2	-	2	-

LAT	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	8	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
1	-	-	-4	-	2	2	2	-2	-	-4	-	-	2	-2	-2	-2
2	-	-	-	-	-	4	-	4	-	-	-	-	-	4	-	-4
3	-	4	-	-	-2	-2	2	-2	-	-	-4	-	-2	2	-2	-2
4	-	-	-	-	-	-	-	-	-	4	-	4	4	-	-4	-
5	-	-	-4	-	-2	-2	-2	2	-	-	-	-4	2	2	-2	2
6	-	4	-	-4	-	-	-	-	-	-	-	-	-	-	4	-
7	-	-	-	-4	2	-2	-2	-2	-	-	4	-	-2	2	-2	-2
8	-	-2	-4	-2	2	-	-2	-	-	2	-4	2	-2	-	2	-
9	-	2	-	2	-	2	-4	-2	4	-2	-	2	-	2	-	2
a	-	-2	-	2	-2	-	-2	-4	-	2	-	-2	2	-	2	-4
b	-	-2	-	-2	-	2	4	-2	4	2	-	-2	-	2	-	2
c	-	-2	4	-2	2	-	-2	-	-	-2	-4	-2	2	-	-2	-
d	-	2	-	2	4	-2	-	2	4	2	-	-2	-	-2	-	-2
e	-	2	-	-2	-2	4	-2	-	-	2	-	-2	-2	-4	-2	-
f	-	2	-	2	4	2	-	-2	-4	2	-	-2	-	2	-	2

4 Differential Cryptanalysis:

In case of block cyphers, it refers to a collection of methods for identifying differences within the network of transformations, identifying instances in which the cypher displays non-random behaviour, and taking use of these characteristics to find the secret key.
We can get the lower bounds on the number of active Sboxes for up to four rounds efficiently for pyjamask-128.
The bounds obtained provide a strong indication that no high probability characteristic exist for Pyjamask.
The lower bounds are given in the table below:

Cipher	**1**	**2**	**3**	**4**
Pyjamask-96	1	12	19	$\in \{27, \ldots, 30\}$
Pyjamask-128	1	12	$\in \{18, 19\}$	≥ 20

We use the ideal 2-round characteristic to investigate the feasibility of characteristics with a low number of active Sboxes for more rounds and stretch it out in both directions.
When it comes to Pyjamask-128, the 4-bit Sbox S4 does not permit iterative propagation from 1-bit difference to 1-bit difference.
One of the better characteristics for Pyjamsk-128 can be:

Round	**Input to Sbox Layer**	**Input to Linear Layer**	**Active**
0	281a088b200200020000200000080001	08100888280a088b081808092012200a	11
1	1b8983b0175328ad345a10f629c9b369	00000000031b2a090cd88bb03b99b3ff	26
2	0000000000000001180040c9000040c8	000000000000000000000000180040c9	7
3	0000000000000000000000000114a000	0114a000011480000114200000000000	5
4	e6e2431674f49dd216e2eb1900000000	c684f6152430b9cec4b29804b6c6eac3	27
5	041000c80210018006000c8061000c8	061001c800100180061001c804100148	7
6	7d31d40c9f26e70a5b4dcd134fa24e25		

5 Integral Cryptatnalysis:

It is a **Chosen Plaintext Attack** proposed on SQUARE Block Cipher by Daemen.
Similar to Differential but here we use a Set of Plaintexts known as Delta Set Δ instead of just two.

- **All Property (A):** Every value appears as the same number in the multiset.
- **Constant Property (C):** The value is fixed to a constant for all texts in the multiset.
- **Balanced Property (B):** The XOR of all texts in the multiset is 0.
- **Unknown (U):** The multiset is indistinguishable from one of n-bit random values.

The first step is to obtain the longest distinguisher for Pyjamask-128. In AES we had a 2 round distinguisher due to 2 round avalanche effect.

In square attack(integral) of AES our input plaintext space had All(A) in one byte and Constant(C) in the remaing bytes which comes out to be 256 plaintexts, with (A^1,C^{15}) we could get the (B^{16}) porperty in just two rounds. To find out the longest distinguisher for pyjamask-128 lets see the longest distinguisher for pyjamask-96. The following are some possible distinguishers for pyjamask-96.

$$(C^{32},C^{32},A^9C^{23}) \xrightarrow{5R} (B^{32},B^{32},B^{32})$$
$$(C^{32},C^{32},A^{17}C^{15}) \xrightarrow{6R} (B^{32},B^{32},B^{32})$$
$$(C^1A^{31},A^{32},A^{32}) \xrightarrow{11R} (B^{32},{B_c}^{32},B^{32})$$

In the third case we would need 2^{95} plaintexts because there is only one constant bit and 95 all bits.

Similarly for pyjamask-128 for a set of 2^{127} plaintexts of the form $((A^{32},C^1A^{31},A^{15},A^{32})$ the intermediate state after 9 rounds has the form of $(B^{32},U^{32},B^{32},U^{32})$. We know how to extract a key using square attack on AES, similarly we guess some bits of the key and try going in backward direction of encryption till we reach the balanced property. The subkey space keeps reducing which will give us an idea of what could be the key.

6 Automated Cryptanalysis:

MILP Constraints:
To ensure that $a_i = 1$, when any of the X_{ij} in its input is 1.
$X_{0j}, X_{1j}, X_{2j}, X_{3j}$ are the column elements of the state that goes to sbox a_j and if any one of the above four elements is one then sbox is active.
$X_{00} - a_0 \leq 0$
$X_{10} - a_0 \leq 0$
$X_{20} - a_0 \leq 0$
$X_{30} - a_0 \leq 0$

$$\begin{array}{|c|}\hline X00 \\ \hline X10 \\ \hline X20 \\ \hline X30 \\ \hline \end{array} \longrightarrow \boxed{a0}$$

Also, when $a_i = 1$, one of Xij in its input must be 1:
$X_{00} + X_{10} + X_{20} + X_{30} - a - 0 \geq 0$.

$$\begin{array}{|c|}\hline \text{X00} \\ \hline \text{X10} \\ \hline \text{X20} \\ \hline \text{X30} \\ \hline \end{array} \longrightarrow \boxed{\text{a0}}$$

$Y_{0j}, Y_{1j}, Y_{2j}, Y_{3j}$ are the output bits of $X_{0j}, X_{1j}, X_{2j}, X_{3j}$ after applied through Sbox a_j

The input difference must result in output difference and vice versa:
$4Y_{00} + 4Y_{10} + 4Y_{20} + 4Y_{30} - (X_{00} + X_{10} + X_{20} + X_{30}) \geq 0$
$4X_{00} + 4X_{10} + 4X_{20} + 4X_{30} - (Y_{00} + Y_{10} + Y_{20} + Y_{30}) \geq 0$

$$\begin{array}{|c|}\hline \text{X00} \\ \hline \text{X10} \\ \hline \text{X20} \\ \hline \text{X30} \\ \hline \end{array} \longrightarrow \boxed{\text{a0}} \longrightarrow \begin{array}{|c|}\hline \text{Y00} \\ \hline \text{Y10} \\ \hline \text{Y20} \\ \hline \text{Y30} \\ \hline \end{array}$$

7 Conclusion:

We have tried to give an insight of how the Round functions, Key Expansion, Encryption and Decryption work in Pyjamask-128 visually. Also we have performed integral and differential attacks on the cipher. We found that the cipher is immune to differential attacks as it only gives some information up to 6 rounds. In the integral attack, we took a particular case which gives us balanced property after 9 rounds(distinguisher) and now we have to follow the same process as AES square attack and recover key.

8 References:

1. https://eprint.iacr.org/2021/1572.pdf

2. https://doc.sagemath.org/html/en/reference/cryptography/sage/crypto/sbox.html

3. https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/pyjamask-spec-round2.pdf