SERPENT
Term Paper by Hope_we_3 SDVV

Tallapaka Lahari Sreeja[1], Shaleen Malik[2] and Sahithi Ampolu[3]

[1] Indian Institute of Technology, Raipur, India, tallapakasreeja@iitbhilai.ac.in
[2] Indian Institute of Technology, Raipur, India, shaleenmalik@iitbhilai.ac.in
[3] Indian Institute of Technology, Raipur, India, ampolusahithi@iitbhilai.ac.in

Abstract. In this paper, we have given a detailed description of the SERPENT Cipher. We have produced block diagrams for the cipher and also given the precise implementation of the SERPENT cipher with both sliced and non-sliced implementations. We analyzed a few generic attacks which can be implemented on the cipher.

Keywords: Linear Transformation · Bit Slice · Symmetric Key

1 Introduction

Serpent is a symmetric key block cipher that was a finalist in the Advanced Encryption Standard (AES) contest, where it was ranked second to Rijndael. Serpent was designed by Ross Anderson, Eli Biham, and Lars Knudsen. Serpent has a block size of 128 bits and supports a key size of 128, 192 or 256 bits.The cipher is a 32-round substitution–permutation network operating on a block of four 32-bit words.Serpent was designed so that all operations can be executed in parallel, using 32 bit slices.

2 Brief Description of SERPENT cipher

The main variant of the cipher encrypts a 128-bit plaintext P to a 128-bit cipher text C in 32 rounds under the control of 33 128-bit subkeys K_0,..., K_{32}. The input is considered as 4 words of 32 bit each. The cipher is an SP-network and consists of:

- an initial permutation IP
- 32 rounds, each consisting of a key mixing operation, a pass through S-boxes (S box S_j is used in each round i where j = i mod 32) and a linear transformation. In the last round, this linear transformation is replaced by an additional key mixing operation
- a final permutation FP.

Each round is comprised by a key addition, an substitution layer of S-box, a linear transformation layer.
Key Addition (k_i): Each round function has a key addition k_i of round i to the input.
Permutation layers : There are two permutation layers Initial permutation at the beginning of first round and Final permutation after 32 rounds which is the inverse of initial permutation. **Substitution layer** : SERPENT uses 8 different s-boxes S_0 to S_7 with S_{imod32} s-box in ith round.
In the non-bit sliced version 4 adjacent bits enters the sbox while in bit-sliced version the ith bits of word0 , word1 , word2 , word3 are given as input to the ith sbox.

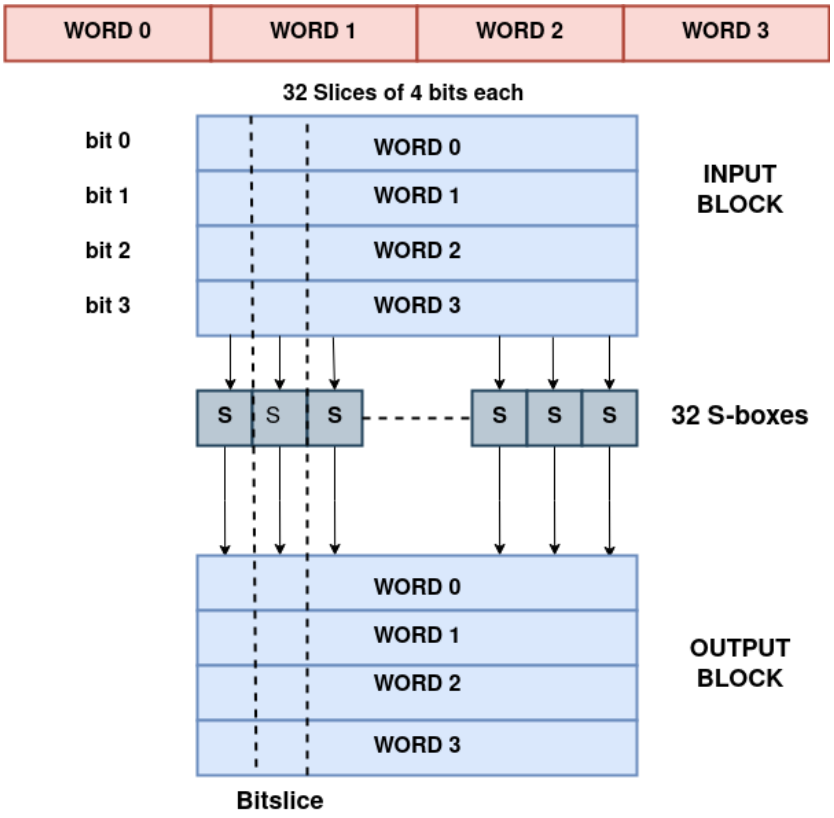


Figure 1: Sbox

Linear Transformation layer: Linearly mixes the 4 words using XOR,bit left shift and bit right shift of words and rotations.(Replaced in last round by additional key mixing).

2.1 Encryption

Let B_i represent Serpent's intermediate state prior to the ith round of encryption. Notice that $B_0 = \text{P}$ and $B_32 = \text{C}$, where P and C are the plaintext and ciphertext, respectively. Let K_i represent the 128-bit ith round subkey and let S_i represent the application of the ith round S-box. Let L be Serpent's linear transformation. Then the Serpent round function is defined as:

$$X_i \leftarrow B_i \oplus K_i$$

$$Y_i \leftarrow S_i(X_i)$$

$$B_{i+1} \leftarrow L(Y_i)\ i = 0, ..., 30$$

$$B_{i+1} \leftarrow Y_i \oplus K_{i+1}\ i = 31$$

Consider the application of an S-box S_i to the 128 bit block X_i. Serpent first separates X_i into four 32-bit words x0, x1, x2, and x3. For each of the 32-bit positions, Serpent constructs a nibble from the corresponding bit in each of the four words, with the bit from x_3 being the most significant bit. Serpent then applies the S-box S_i to the constructed

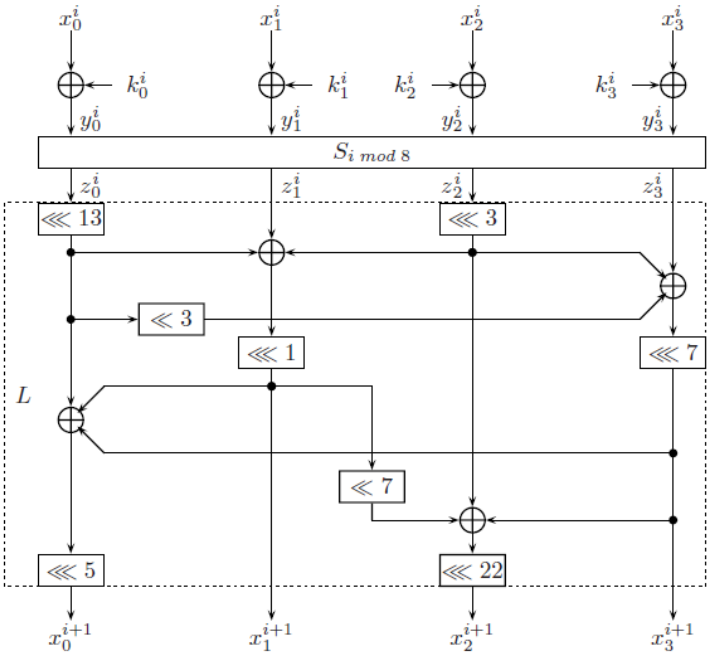


Figure 2: Round function of serpent round i

nibble and stores the result in the respective bits of Y_i = (y0; y1; y2; y3). The linear transform L on Y_i = (y0; y1; y2; y3) is defined as

$$y_0 \leftarrow y_0 <<< 13$$

$$y_2 \leftarrow y_2 <<< 3$$

$$y_1 \leftarrow y_0 \oplus y_1 \oplus y_2$$

$$y_3 \leftarrow y_2 \oplus y_3 \oplus (y_0 << 3)$$

$$y_1 \leftarrow y_1 <<< 1$$

$$y_3 \leftarrow y_3 <<< 7$$

$$y_0 \leftarrow y_0 \oplus y_1 \oplus y_3$$

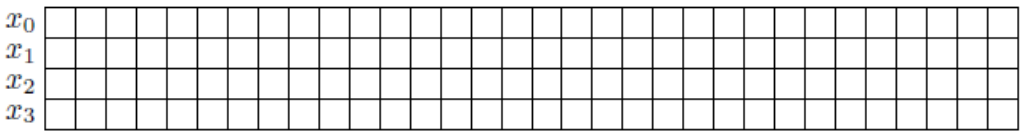
$$y_2 \leftarrow y_2 \oplus y_3 \oplus (y_1 << 7)$$

$$y_0 \leftarrow y_0 <<< 5$$

$$y_2 \leftarrow y_2 <<< 22$$

$$B_{i+1} \leftarrow (y_0, y_1, y_2, y_3)$$

where «< denotes a left rotation and « denotes a left shift. The internal state of SERPENT is represented as :



where Xi is the internal state under inspection and $X_i = (x_0; x_1; x_2; x_3)$.

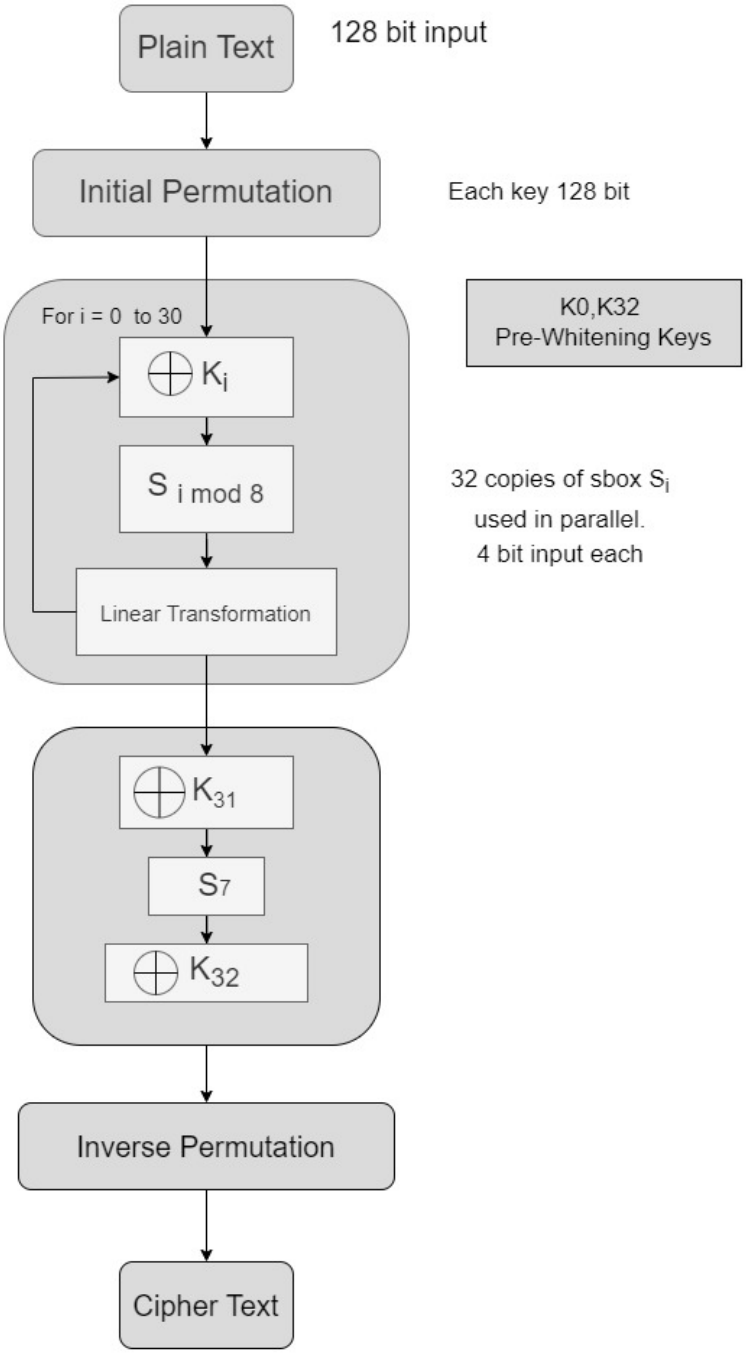


Figure 3: Block Diagram

2.2 The Key Schedule

In a 256-bit key size, Serpent sets the eight 32-bit words $w_8; w_7; ....; w_1$ to the key. If not,the key is converted to a 256-bit key by appending a '1' bit followed by a string of '0's. Serpent computes the prekeys $w_0; w_1; :::; w_131$ using the recurrence

$$w_i \leftarrow (w_{i-8} \oplus w_{i-5} \oplus w_{i-3} \oplus w_{i-1} \oplus \phi) <<< 11$$

where ϕ is 0x9e3779b9.
Serpent then computes the 128-bit subkeys Kj by applying an S-box to the prekeys $w_{4j}; ...; w_{4j+3}$:

$$K_0 \leftarrow S_3(w_0, w_1, w_2, w_3)$$
$$K_1 \leftarrow S_2(w_4, w_5, w_6, w_7)$$
$$K_2 \leftarrow S_1(w_8, w_9, w_10, w_11)$$
$$\vdots$$
$$K_{31} \leftarrow S_4(w_124, w_125, w_126, w_127)$$
$$K_{32} \leftarrow S_0(w_128, w_129, w_130, w_131)$$

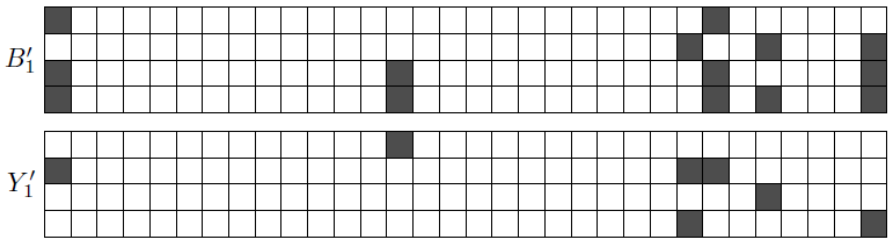
3 Practical Attacks

All publicly known attacks are computationally infeasible, and none of them affect the full 32-round Serpent.

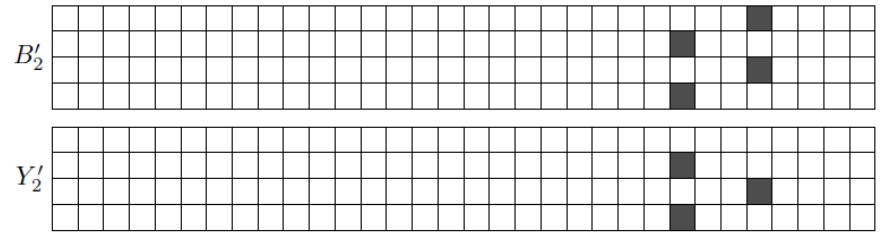
3.1 Basic Six-Round Differential Attack

Although there may exist other high-probability differentials through several rounds of Serpent, we focus on a particular five-round characteristic, $B_1^{'} \rightarrow Y_5^{'}$,with probability $p = 2^{-80}$. This characteristic spans Serpent's second through sixth rounds (rounds i = 1,.....,5).For completeness, this characteristic is illustrated in figure given here.

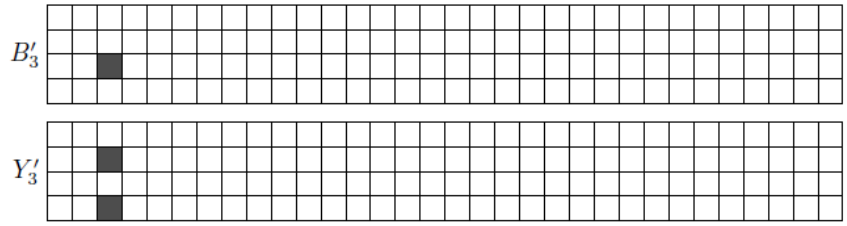
The first round characteristics,$B_1^{'} \rightarrow Y_1^{'}$, has probability 2^{-13}



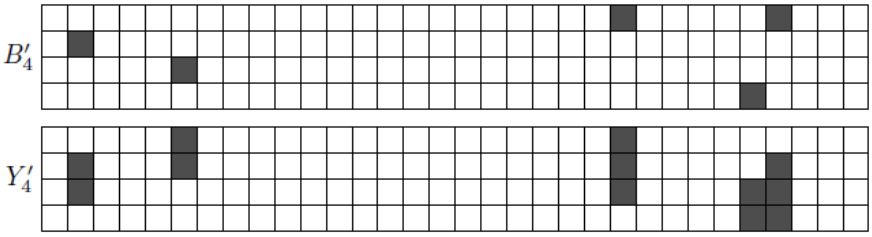
The second round characterstics has probability has 2^{-5}



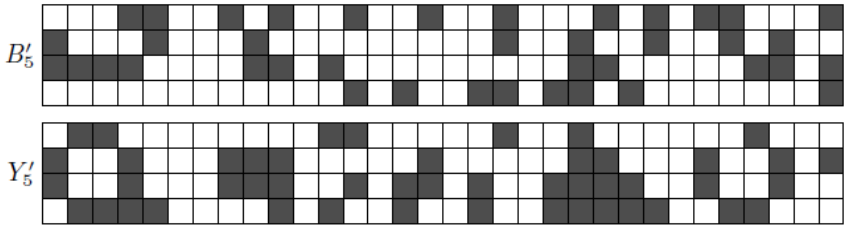
The third round characterstics has probability has 2^{-3}



The fourth round characterstics has probability has 2^{-10}

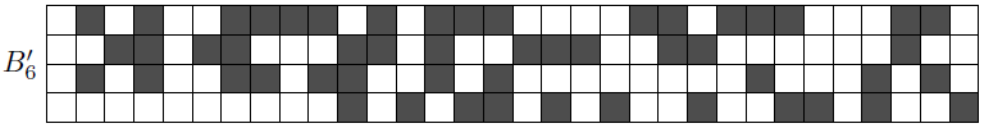


The fifth round characterstics has probability has 2^{-49}.



Notationally, we use $X^{'}$ to represent the xor difference between two values X and X^{*}.
We can use the above-mentioned five-round, probability $p = 2^{-80}$, characteristic to attack rounds one through six of 192- and 256-bit Serpent.For our attack we request 2^{82} plaintext pairs with input difference $B_1^{'}$. We set a count variable to zero for each final round subkey guess. Then, for each unfiltered pair, we peel back to the previous round and look for our projected output difference from the fifth round.If we observe our expected output difference,we increment our counter. If we count three or more right pairs, this subkey might be correct.

If we apply the linear transformation L to the intermediate difference $Y_5^{'}$, we get the following expected input divergence to the sixth round: We can immediately identify all



but approximately 2^{-47} of our ciphertext pairs as wrong pairs because their differences $B_7^{'}$ cannot correspond to our desired difference $B_6^{'}$.
After filtering we are left with approximately 2^{35} ciphertext pairs. Our attack thus requires

approximately 2^{35} x 2^{116} partial decryptions, or work equivalent to approximately 2^{150} six-round Serpent encryptions. If we retain only our unfiltered ciphertext pairs, this attack requires approximately 2^{40} bytes of sequential memory. The signal to noise ratio of this attack is 2^{35}.

3.2 Bit Pattern Based Integral Attack

3.5-round Distinguisher: Serpent reduced to 3.5 rounds can be distinguished from a random permutation by choosing a structure of 2^10 plaintexts. The paper explains how sbox preserve balancedness. Terminology required for bit pattern attack: In this attack the positions of each bit is independently taken into consideration.Each bit position in a plaintext structure holds a specific sequence of bit '0' and/or '1'. The pattern in which the bit sequence is repeated forms the foundation of the bit-based notations. The following describes the notation.

- The pattern c in a position means that all bits in this position within the structure consists of only bit '0' or '1'. This pattern is called a constant bit pattern.
- The pattern a_i in a position means that the first block of 2i consecutive bits in this position are constant, and the next block of 2i consecutive bits all have the opposite value of the first. The alternating values of bits in 2i-blocks is repeated throughout the structure. This pattern is called an active bit pattern.
- The pattern b_i in a position means that blocks of 2i consecutive bits in this position are constant, but the values of the blocks are not necessarily repeated in an alternating manner.
- The pattern d_i in a position means that the bits in this position may hold either a c (constant) or an a_i (active) pattern. This pattern is called a dual bit pattern.

If the XOR sum of all the bits in one pattern equals 0, we say that the pattern is balanced. Furthermore, if the cipher block which is the XOR sum of all the texts in a structure only has 0-bits we say that the structure is balanced.All patterns described above are balanced, except for the b_0-pattern which may or may not be balanced. Balanced pattern b_0 is denoted with b_0 and unbalanced with ${b_0}^*$.

Bit-patterns will be XORed together in the linear operations of the cipher. The following properties are easy to verify.

- c $\oplus$ p = p for any pattern p
- $a_i \oplus a_i$ = c
- $p_j \oplus q_i = b_i$ for j>i and p,q $\in$ a,b.If $i = 0$ the right-hand side will be b_0.
- p $\oplus$b$_0^*$ = ${b_0}^*$ when p $\neq$b$_0$

The plain-texts are chosen such that the ten most significant bits of x0 2 hold all possible 10-bit values. The rest of the plaintext bits hold constant values such as the following:

$$(X_0)^(j) = ({x_0}^{0(j)}, {x_1}^{0(j)}, {x_2}^{0(j)}, {x_3}^{0(j)} = (c_0, c_1, j||c_2, c_3)$$

where $0 \leq$ j $\leq 2^{10} - 1, c_0, c_1, c_3$ are 32-bit constants and c_2 is a 22-bit arbitrary constant.The ten leftmost bits of $(x_2)^0(j)$ therefore hold the pattern a9a8 . . . a0 and the rest of the bits hold a c pattern. In first round, the inputs to the ten affected instances receive an input difference of 4. Each input value is repeated 2^9 times thus sbox outputs a pair of values repeated 2^9 times.The output bits of S-Boxes are denoted as d_i pattern as 8 S-boxes behave differently but at least 2 bits have a_i pattern because of the Sbox property that one-bit

input change results in at least two-bit output change. All other bits remain constant. The linear layer in the first round doesnot affect the balancedness of the structure.
Each instance of the second round S-box may receive a single constant input value, or between two to sixteen different input values. The input values are repeated between 2^6 and 2^10 times. The number of distinct outputs is therefore even and this ensures that the structure is balanced after the S-box in the second round.
The linear layer in the second round ensures that the number of distinct values in every column occurs an even number times, or that all values occur an odd number of times. All bits will hold a $b_0{}^*$ pattern except for a few positions which still retain a b_1-pattern. These bits are then fed to the third round S-box. The number of repetition for each distinct output value matches that of its input and this preserves the balancedness of the structure until after L in the third round.
The application of the fourth round S-boxes is expected to destroy the balancedness of the structure.
Key Recovery:
In last round linear layer is replaced with key addition so for four-round Serpent the ciphertexts and the output of the S-boxes in the fourth round is only separated by a simple XOR of the last round key. This attack requires 2×2^10 chosen plaintexts, and it must be repeated 32 times to recover the whole last round key. The time complexity is therefore $2 \times 2^10 \times 2^4 \times 32 = 2^20$ partial encryptions.

4 Automated Cryptanalysis

The S-boxes can be formulated using the equations:
To ensure S-box to be active when any one of its input is 1.
x_00 - $a_00 \leq 0$
x_01 - $a_00 \leq 0$
x_02 - $a_00 \leq 0$
x_03 - $a_00 \leq 0$
To ensure when S-box is active,one of its must be 1:
$x_00 + x_01 + x_02 + x_03$ - $a_00 \geq 0$.
Few constraints to model inputs and outputs difference:
$x_00 + x_01 + x_02 + x_03$ - $y_00 + y_01 + y_02 + y_03 \geq -3y_00 + y_01 + y_02 + y_03$ - $x_00 + x_01 + x_02 + x_03 \geq -3$

5 S-box Analysis

The implementation of the Serpent cipher requires the use of 8 different 4-bit s-boxes.

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S_0[x]$	3	8	F	1	A	6	5	B	E	D	4	2	7	0	9	C
$S_1[x]$	F	C	2	7	9	0	5	A	1	11	E	8	6	D	3	4
$S_2[x]$	8	6	7	9	3	C	A	F	D	1	E	4	0	B	5	2
$S_3[x]$	0	F	B	8	C	9	6	3	D	1	2	4	A	7	5	E
$S_4[x]$	1	F	8	3	C	0	B	6	2	5	4	A	9	E	7	D
$S_5[x]$	F	5	2	B	4	A	9	C	0	3	E	8	D	6	7	1
$S_6[x]$	7	2	C	5	8	4	6	B	E	9	1	F	D	3	A	0
$S_7[x]$	1	D	F	0	E	8	2	B	7	4	C	A	9	3	5	6

The Serpent s-boxes are 4-bit permutations, and subject to the following properties:

- a 1-bit input difference will never lead to a 1-bit output difference, a differential characteristic has a probability of 1:4 or less.
- each linear characteristic has a probability in the range $1/2 \pm 1/4$, and a linear relation between one single bit in the input and one single bit in the output has a probability in the range $1/2 \pm 1/8$.
- the nonlinear order of the output bits as function of the input bits is 3. However there have been output bits found which in function of the input bits have an order of only 2.

5.1 DDT and LAT for S-Box 0

Table 1: LAT

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	-2	-2	0	-2	0	0	-2	0	-2	2	4	-2	0	-4	2
2	0	2	-2	0	0	2	2	4	0	-2	-2	4	0	-2	2	0
3	0	0	-4	0	2	-2	-2	-2	0	-4	0	0	2	2	2	-2
4	0	0	0	0	0	0	0	0	0	0	0	0	4	-4	-4	-4
5	0	2	-2	4	-2	-4	0	2	0	2	2	0	2	0	0	2
6	0	-2	2	0	0	-2	-2	4	-4	-2	-2	0	0	2	-2	0
7	0	0	0	-4	2	-2	2	2	4	0	0	0	2	2	-2	2
8	0	-2	-2	0	2	0	0	2	0	-2	2	-4	-2	-4	0	2
9	0	0	0	0	0	4	-4	0	0	0	0	0	4	0	0	4
A	0	4	0	0	2	-2	-2	-2	0	0	-4	0	-2	-2	-2	2
B	0	-2	2	0	4	-2	-2	0	0	2	2	4	0	-2	2	0
C	0	-2	2	4	2	0	4	-2	0	-2	-2	0	2	0	0	2
D	0	-4	-4	0	0	0	0	0	0	4	-4	0	0	0	0	0
E	0	0	0	4	2	2	-2	2	4	0	0	0	-2	2	-2	-2
F	0	-2	2	0	-4	-2	-2	0	4	-2	-2	0	0	-2	2	0

Table 2: DDT

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	2	0	2	2	2	0	0	0	2	2	0	4	0
2	0	0	0	0	0	0	0	0	0	2	2	0	4	2	2	4
3	0	2	2	2	0	0	0	2	0	4	0	2	2	0	0	0
4	0	0	0	0	0	0	0	0	0	4	4	0	0	4	4	0
5	0	0	2	0	4	2	0	0	2	0	2	2	0	0	2	0
6	0	2	2	4	0	2	2	2	4	0	0	0	0	0	0	0
7	0	0	2	0	4	2	0	0	2	2	0	2	0	2	0	0
8	0	0	0	2	0	2	2	2	0	0	0	2	2	4	0	0
9	0	2	2	0	0	2	2	0	0	2	2	0	0	2	2	0
A	0	2	2	2	0	0	0	2	0	0	4	2	2	0	0	0
B	0	2	2	0	0	2	2	0	0	0	0	0	4	0	0	4
C	0	2	0	0	4	0	2	0	2	2	0	2	0	2	0	0
D	0	0	0	4	0	0	0	4	4	0	0	0	0	0	0	4
E	0	2	0	0	4	0	2	0	2	0	2	2	0	0	2	0
F	0	2	2	0	0	2	2	0	4	0	0	0	0	0	0	4

- The Differential uniformity of all S-Boxes is 4.
- The Differential branch number for each S-Box is 3.
- The Linearity of each S-Box is 2.

Table 3: A Table comparing Sboxes of Other groups

Group Name	Cipher Name	Sbox SIze	Differential Uniformity	Differential branch number
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
TechHeist 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
SHA69	Mysterion	4-bit	4	2
Hex Brains	Rectangle	4-bit	4	2

6 Software Application

This Software application is used to implement SERPENT Cipher in encrypting and decrypting individual columns in CSV files.It is implemented with the help of Flask,Python,Html,Css.

On running the software application on local machine,the html file opens on the webserver.We first select whether we want to perform Encryption or Decryption.On selecting Data Encryption,the web page is redirected to SERPENT Encrpytion in which we upload a CSV file,a key which is in hex digits along with the Column name.On uploading,the Serpent Cipher implementation will encrypt the selected column and update the CSV file with encrypted column. On pressing the download button the updated CSV file will directly download into the local machine.The same happens with Data Decryption where we enter the key we used to encrypt the file along with the encrypted file and the column to be decrypted. The decrypted file will automatically download onto the local machine.

This Software application can be used in places where only few columns of information has to be hidden so that it is multi-applicable.One such example is Hospital where details of patients can be encrypted and the remaining data regarding diseases,symptoms,etc.. can be shared with others.This helps in protecting the privacy of the patients and also in decrease of database storage as same file can be used for all purposes.

7 Conclusion

We successfully implemented the Serpent cipher in python and have detailed S-box analysis.There is a thorough description of differential attack and bit-wise integral attack on round reduced serpent cipher.

8 References

- Serpent: A Proposal for the Advanced Encryption Standard [Refer Here]
- Bit-Pattern Based Integral Attack [Refer Here]
- Preliminary Cryptanalysis of Reduced-Round Serpent [Refer Here]