# Khazad-Block-Cipher

Swapnil Narad
Devansh Chaudhary
Aditya Susawat

[1] IIT Bhilai, Raipur, India, swapniln@iitbhilai.ac.in
[2] IIT Bhilai, Raipur, India, devanshc@iitbhilai.ac.in
[3] IIT Bhilai, Raipur, India, adityaks@iitbhilai.ac.in

**Abstract.** This paper contains a detailed report on the KHAZAD cipher. This paper gives a brief history of the KHAZAD cipher followed by some features of the cipher like block size, key size, etc. The implementation of the KHAZAD cipher is also mentioned in this paper and it contains a detailed description of the key expansion algorithm, general round structure of the cipher, the encryption algorithm, the decryption algorithm, and the code in python for the cipher implementation. This paper also has details about the s-box of KHAZAD and various attacks on the cipher like differential attack, integral attack, etc. on a round reduced variant of the KHAZAD cipher. Finally, it also contains the details about the security and key features of the cipher and the brownie point for our work done.

**Keywords:** encryption, decryption, integral, differential

## 1 Introduction

KHAZAD is a symmetrical block cipher developed by two cryptographers: Belgian Vincent Reimen (author of the cipher Rijndael) and Brazilian Paulo Barreto. [Wik]
KHAZAD was presented at the European competition of cryptographic primitives NESSIE in 2000.
**Key Size**: 128 bits
**Block Size**: 64 bit (8 bytes)

The predecessor of the KHAZAD algorithm is considered to be the SHARK code developed in 1995 by Vincent Rayman and Joan Dimen. The authors of KHAZAD claim that the algorithm is based on the strategy of developing cryptographically stable encryption algorithms (Wide-Trail strategy), proposed by Joan Dimen.

The KHAZAD algorithm has conservative parameters and is designed to replace existing ciphers with a 64-bit block, such as IDEA and DES, providing a higher level of security at high execution speeds.

Involution transformations are widely used in ciphers, which minimizes the difference between encryption and decryption algorithms.
[Bar]

| *Name* | **KHAZAD** |
|---|---|
| *Number of rounds* | **8** |
| *Schedule (extension) of the key* | **The Feistel scheme** |
| *Unreduced polynomial of the field* $GF(2^8)$ | $x^8 + x^4 + x^3 + x^2 + 1$ |
| *Implementation of the S-box* | **Recursive P - and Q-miniblocks** |
| *Implementation of the mixing matrix* | **Involutional MDS code** |

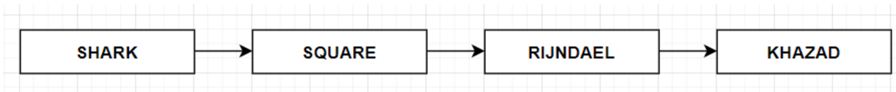
# 2 Main Result

## 2.1 Brief Description

It's an block cipher that has a single variant block size of 64 bits with a key of single variant of size 128 bits. The input data block is served as a string of 8 bytes.

The S-box and the mixing matrix are chosen in a way which ensures that the process of encryption and decryption are the same, except the round connections.

KHAZAD, like the AES algorithm (Rijndael), is from a family of block ciphers which are formed from the SHARK cipher.




## 2.2 The Implementation Algorithm :

- Apply key extension algorithm to the key results in a set of round keys.
- The algorithm have 8 rounds, each of which consists of 3 stages:
  1. *Nonlinear Transformation* $\gamma$
  2. *Linear Transformation* $\theta$
  3. *Adding a round key* $\sigma$
- Set of round keys $k_0$ ... $k_8$ obtained by applying to the encryption key K key extension procedures.
- Before the first round $\sigma(k_0)$ is performed.
- Operation $\theta$ is not performed in the last round.

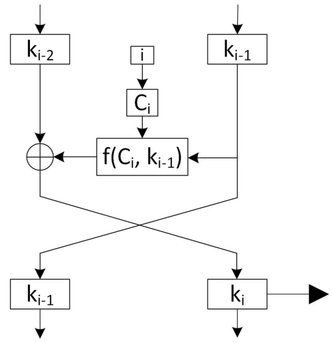
## 2.3 Key Expansion

- 128-bit (16-byte) key K is divided into 2 equal parts:
- $k_{-1}$ - older 8 bytes (from the 15th to the 8th)
- $k_{-2}$ - lower 8 bytes (from the 7th to the 0th)
- Keys $k_0 \dots k_8$ calculated according to the Feistel scheme :
- $k_i = f(C_i, k_{i-1}) \oplus k_{i-2}$

Here:

$f(x, y)$ - round function of the algorithm with the input block x and the key y.

$C_i$ - 64-bit constant, j which byte is $C_i^j = S(8i + j)$




## 2.4 General Round Structure

### 2.4.1 Nonlinear transformation

Denoted as $\gamma$. In each round, the input block is divided into smaller blocks of 8 bytes, which are independently subjected to nonlinear transformation (change), i.e. pass in parallel through the same S-blocks (each S-block - 8x8 bits, i.e. 8 bits at the input and 8 bits at the output).

Replacement blocks in the source and modified (tweaked) ciphers are different. The substitution unit is selected so that the nonlinear transformation is involutionary, i.e. $\gamma = \gamma^{-1}$ *or* $\gamma(\gamma(x)) = x$.

### 2.4.2 Linear transformation

Denoted by $\theta$. An 8-byte row of data is multiplied byte by byte to a fixed matrix H size 8 x 8, and byte multiplication is performed in the Galois field $GF(2^8)$ with a polynomial that is not given $x^8 + x^4 + x^3 + x^2 + 1$ $(0x11D)$.[BR00]

$$\theta(x) = x \times H \quad \text{where}$$

$$H = \begin{bmatrix} 01_x & 03_x & 04_x & 05_x & 06_x & 08_x & 0B_x & 07_x \\ 03_x & 01_x & 05_x & 04_x & 08_x & 06_x & 07_x & 0B_x \\ 04_x & 05_x & 01_x & 03_x & 0B_x & 07_x & 06_x & 08_x \\ 05_x & 04_x & 03_x & 01_x & 07_x & 0B_x & 08_x & 06_x \\ 06_x & 08_x & 0B_x & 07_x & 01_x & 03_x & 04_x & 05_x \\ 08_x & 06_x & 07_x & 0B_x & 03_x & 01_x & 05_x & 04_x \\ 0B_x & 07_x & 06_x & 08_x & 04_x & 05_x & 01_x & 03_x \\ 07_x & 0B_x & 08_x & 06_x & 05_x & 04_x & 03_x & 01_x \end{bmatrix},$$

### 2.4.3 Adding a round key

A 64-bit XOR operation is performed on the 64-bit data block & the 64-bit round key . A 64-bit data block is been xored with a round key of 64 bits calculated using key expansion algorithm based on Fiestal scheme.

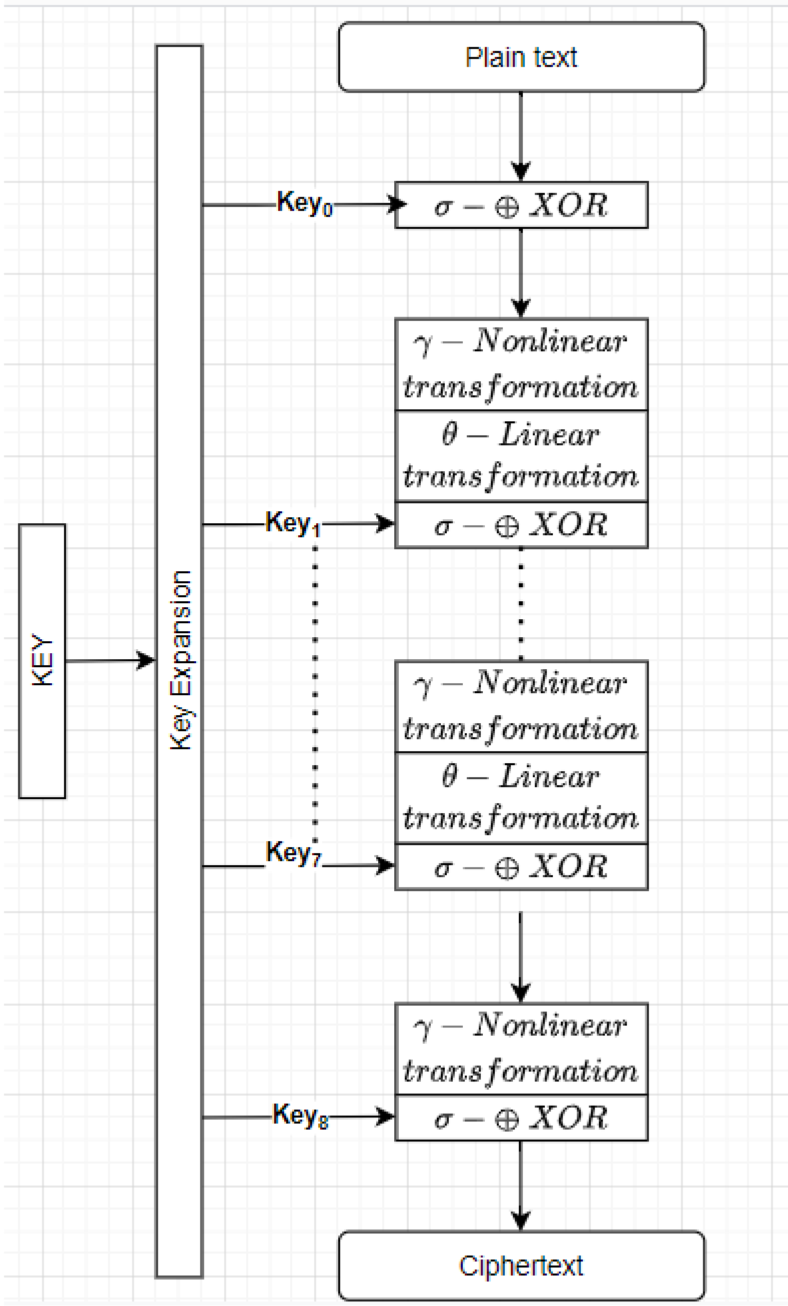
$$\text{For } i^{th} \ round \ : \ \sigma(x_i) = x_{i-1} \oplus k_{i-1}$$

## 2.5 The Encryption and Decryption Algorithm

### 2.5.1 Encryption Algorithm

The encryption algorithm is explained below with the help of figure, the figure give good overview of how different layers have been working in course of 8 rounds.

$$\alpha_8[K^0,\ \ldots\ K^8] = \sigma[K^8].\gamma.\left(Round_1^7\right).\sigma[K^0])$$

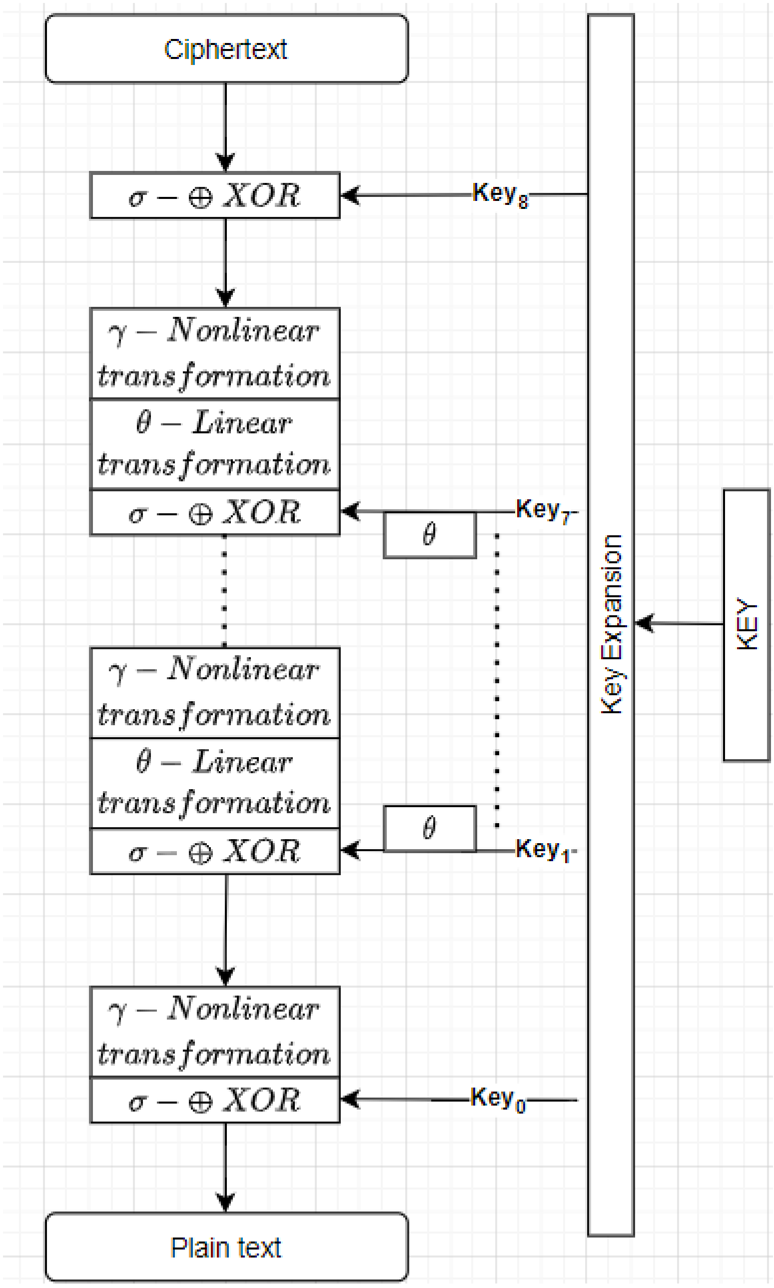



**The Encryption Algorithm**

### 2.5.2 Decryption Algorithm

The decryption algorithm is explained below with the help of figure, the figure give good overview of how different layers have been working in course of 8 rounds.

$$\alpha_8[K^8, \ \ldots \ K^0] = \sigma[K^0].\gamma. \left(Round_7^1\right).\sigma[K^8])$$



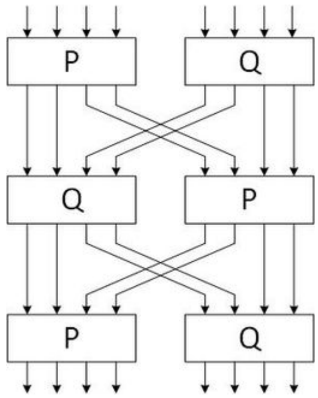

**The Decryption Algorithm**

## 2.6 The Creation of S-box

In the original version of the cipher (KHAZAD-0) tabular replacement was represented by a classic S-block.
The bool type definition $GF(2)^n \longrightarrow$ GF(2) is called Zhegalkin polynomial. The nonlinear order function f is called the order of Zhegalkin's polynomial, i.e. the maximum of the orders of its members.
The KHAZAD-0 cipher uses a pseudo random generated S-block that meets the following requirements :

- must be an involution
- $\delta$-value might not be over $8 \times 2^{-8}$
- $\lambda$-value might not be over $16 \times 2^{-16}$
- nonlinear order $\nu$ should be a maximum, namely equal to 7

In the modified version of the cipher, the S-block 8x8 is modified and represented by a recursive structure consisting of mini-blocks P and Q, each of which is a small replacement block with 4 bits at the input and output (4x4).
Recursive structure of the replacement unit in the modified KHAZAD cipher:[BR00]




This structure of P - and Q-miniblocks is equivalent to the S-block with the following substitution table:

**Correspondence of output values to inputs for mini-block P [Bar]**

| u | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| P (u) | 3 | F | E | 0 | 5 | 4 | B | C | D | A | 9 | 6 | 7 | 8 | 2 | 1 |

**Correspondence of output values to inputs for mini-block Q [Bar]**

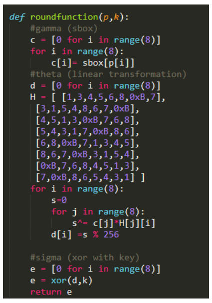
| u | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Q (u) | 9 | E | 5 | 6 | A | 2 | 3 | C | F | 0 | 4 | D | 7 | B | 1 | 8 |

**The final KHAZAD S-box** [BR00]

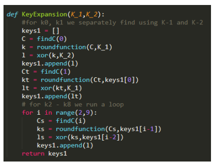
| | 00$_x$ | 01$_x$ | 02$_x$ | 03$_x$ | 04$_x$ | 05$_x$ | 06$_x$ | 07$_x$ | 08$_x$ | 09$_x$ | 0A$_x$ | 0B$_x$ | 0C$_x$ | 0D$_x$ | 0E$_x$ | 0F$_x$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00$_x$ | BA$_x$ | 54$_x$ | 2F$_x$ | 74$_x$ | 53$_x$ | D3$_x$ | D2$_x$ | 4D$_x$ | 50$_x$ | AC$_x$ | 8D$_x$ | BF$_x$ | 70$_x$ | 52$_x$ | 9A$_x$ | 4C$_x$ |
| 10$_x$ | EA$_x$ | D5$_x$ | 97$_x$ | D1$_x$ | 33$_x$ | 51$_x$ | 5B$_x$ | A6$_x$ | DE$_x$ | 48$_x$ | A8$_x$ | 99$_x$ | DB$_x$ | 32$_x$ | B7$_x$ | FC$_x$ |
| 20$_x$ | E3$_x$ | 9E$_x$ | 91$_x$ | 9B$_x$ | E2$_x$ | BB$_x$ | 41$_x$ | 6E$_x$ | A5$_x$ | CB$_x$ | 6B$_x$ | 95$_x$ | A1$_x$ | F3$_x$ | B1$_x$ | 02$_x$ |
| 30$_x$ | CC$_x$ | C4$_x$ | 1D$_x$ | 14$_x$ | C3$_x$ | 63$_x$ | DA$_x$ | 5D$_x$ | 5F$_x$ | DC$_x$ | 7D$_x$ | CD$_x$ | 7F$_x$ | 5A$_x$ | 6C$_x$ | 5C$_x$ |
| 40$_x$ | F7$_x$ | 26$_x$ | FF$_x$ | ED$_x$ | E8$_x$ | 9D$_x$ | 6F$_x$ | 8E$_x$ | 19$_x$ | A0$_x$ | F0$_x$ | 89$_x$ | 0F$_x$ | 07$_x$ | AF$_x$ | FB$_x$ |
| 50$_x$ | 08$_x$ | 15$_x$ | 0D$_x$ | 04$_x$ | 01$_x$ | 64$_x$ | DF$_x$ | 76$_x$ | 79$_x$ | DD$_x$ | 3D$_x$ | 16$_x$ | 3F$_x$ | 37$_x$ | 6D$_x$ | 38$_x$ |
| 60$_x$ | B9$_x$ | 73$_x$ | E9$_x$ | 35$_x$ | 55$_x$ | 71$_x$ | 7B$_x$ | 8C$_x$ | 72$_x$ | 88$_x$ | F6$_x$ | 2A$_x$ | 3E$_x$ | 5E$_x$ | 27$_x$ | 46$_x$ |
| 70$_x$ | 0C$_x$ | 65$_x$ | 68$_x$ | 61$_x$ | 03$_x$ | C1$_x$ | 57$_x$ | D6$_x$ | D9$_x$ | 58$_x$ | D8$_x$ | 66$_x$ | D7$_x$ | 3A$_x$ | C8$_x$ | 3C$_x$ |
| 80$_x$ | FA$_x$ | 96$_x$ | A7$_x$ | 98$_x$ | EC$_x$ | B8$_x$ | C7$_x$ | AE$_x$ | 69$_x$ | 4B$_x$ | AB$_x$ | A9$_x$ | 67$_x$ | 0A$_x$ | 47$_x$ | F2$_x$ |
| 90$_x$ | B5$_x$ | 22$_x$ | E5$_x$ | EE$_x$ | BE$_x$ | 2B$_x$ | 81$_x$ | 12$_x$ | 83$_x$ | 1B$_x$ | 0E$_x$ | 23$_x$ | F5$_x$ | 45$_x$ | 21$_x$ | CE$_x$ |
| A0$_x$ | 49$_x$ | 2C$_x$ | F9$_x$ | E6$_x$ | B6$_x$ | 28$_x$ | 17$_x$ | 82$_x$ | 1A$_x$ | 8B$_x$ | FE$_x$ | 8A$_x$ | 09$_x$ | C9$_x$ | 87$_x$ | 4E$_x$ |
| B0$_x$ | E1$_x$ | 2E$_x$ | E4$_x$ | E0$_x$ | EB$_x$ | 90$_x$ | A4$_x$ | 1E$_x$ | 85$_x$ | 60$_x$ | 00$_x$ | 25$_x$ | F4$_x$ | F1$_x$ | 94$_x$ | 0B$_x$ |
| C0$_x$ | E7$_x$ | 75$_x$ | EF$_x$ | 34$_x$ | 31$_x$ | D4$_x$ | D0$_x$ | 86$_x$ | 7E$_x$ | AD$_x$ | FD$_x$ | 29$_x$ | 30$_x$ | 3B$_x$ | 9F$_x$ | F8$_x$ |
| D0$_x$ | C6$_x$ | 13$_x$ | 06$_x$ | 05$_x$ | C5$_x$ | 11$_x$ | 77$_x$ | 7C$_x$ | 7A$_x$ | 78$_x$ | 36$_x$ | 1C$_x$ | 39$_x$ | 59$_x$ | 18$_x$ | 56$_x$ |
| E0$_x$ | B3$_x$ | B0$_x$ | 24$_x$ | 20$_x$ | B2$_x$ | 92$_x$ | A3$_x$ | C0$_x$ | 44$_x$ | 62$_x$ | 10$_x$ | B4$_x$ | 84$_x$ | 43$_x$ | 93$_x$ | C2$_x$ |
| F0$_x$ | 4A$_x$ | BD$_x$ | 8F$_x$ | 2D$_x$ | BC$_x$ | 9C$_x$ | 6A$_x$ | 40$_x$ | CF$_x$ | A2$_x$ | 80$_x$ | 4F$_x$ | 1F$_x$ | CA$_x$ | AA$_x$ | 42$_x$ |

## 2.7 Code Implementation

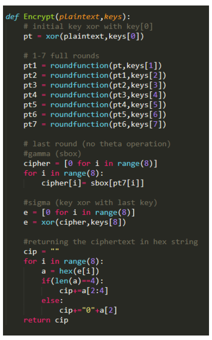
The code file for implementation is provided in the same *github* repository, the test vectors are also provided for the same.



(a) The code snippet for round structure



(b) The code snippet for key expansion algorithm



(c) The code snippet for encryption algorithm

Figure 1: Implementation code snippets

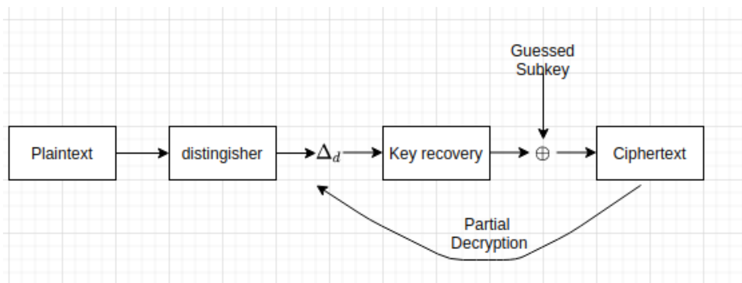
## 2.8 Attacks

The group of ciphers which include Shark, Square, Rijndael, Anubis and Khazad are made in such a way that as far as differential attack and attacks like linear attacks are concerned, it is very unusual to be successful for these ciphers on their full versions.

### 2.8.1 Differential attack

This section will be on a differential attack for a 3 rounds version of Khazad cipher whose time complexity is very large as compared to 3 round integral attack which is discussed below. The effect of each round on the message block due to different layers is shown below:


The attack would follow this strategy:

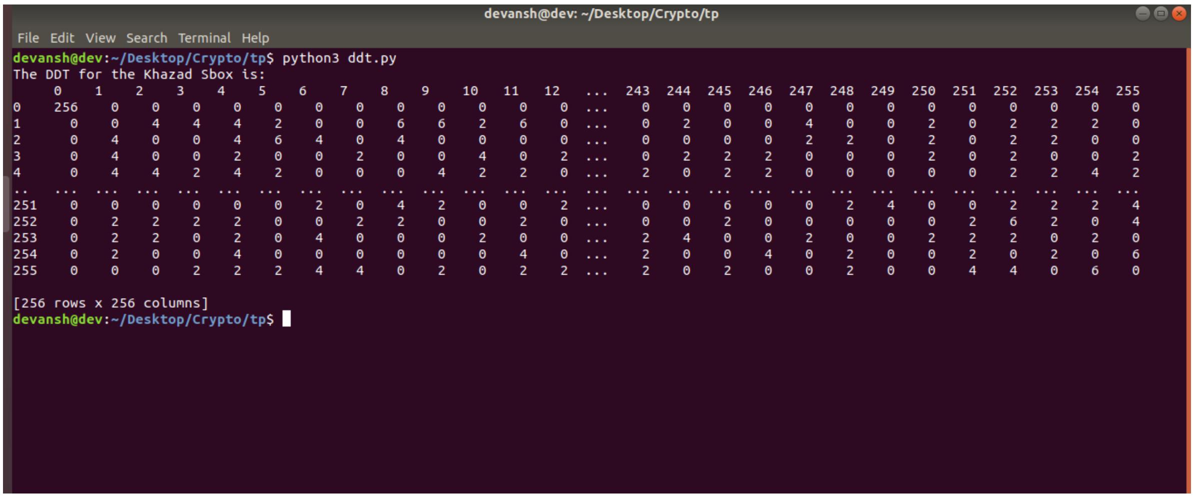



So after guessing 8 bytes of key or guessing subkey there would be at max $2^{64}$ possible guesses for 8 bytes of subkey, therefore the time complexity achieved would be $2^{64}$

| Attack Type | Rounds | Time |
|---|---|---|
| differential attack | 3 | $2^{64}$ |

### 2.8.2 DDT

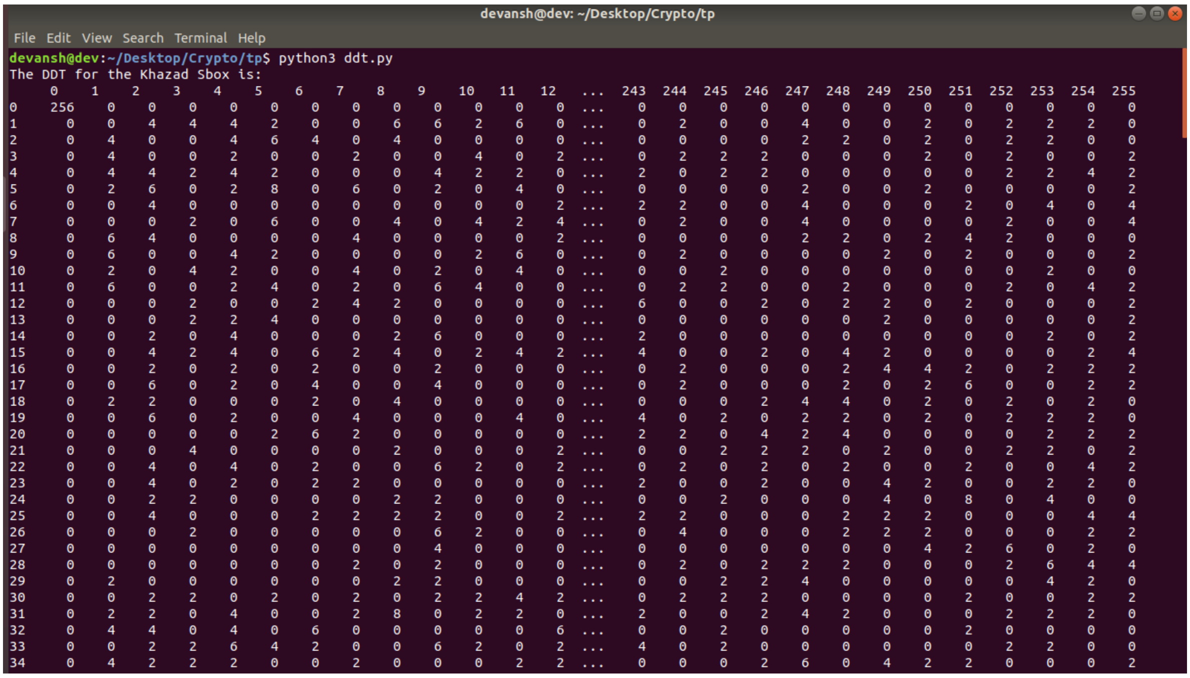
The DDT for s-box of KHAZAD can be created similar to how it was created for other block ciphers.



```
devansh@dev: ~/Desktop/Crypto/tp
File Edit View Search Terminal Help
devansh@dev:~/Desktop/Crypto/tp$ python3 ddt.py
The DDT for the Khazad Sbox is:
      0   1   2   3   4   5   6   7   8   9  10  11  12  ...  243  244  245  246  247  248  249  250  251  252  253  254  255
0   256   0   0   0   0   0   0   0   0   0   0   0   0  ...    0    0    0    0    0    0    0    0    0    0    0    0    0
1     0   0   4   4   4   2   0   0   6   6   2   6   0  ...    0    2    0    0    4    0    0    2    0    2    2    2    0
2     0   4   0   0   4   6   4   0   4   0   0   0   0  ...    0    0    0    0    2    2    0    2    0    2    2    0    0
3     0   4   0   0   2   0   0   2   0   0   4   0   2  ...    0    2    2    2    0    0    0    2    0    2    0    0    2
4     0   4   4   2   4   2   0   0   0   4   2   2   0  ...    2    0    2    2    0    0    0    0    0    2    2    4    2
..  ... ... ... ... ... ... ... ... ... ... ... ... ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...
251   0   0   0   0   0   0   2   0   4   2   0   0   2  ...    0    0    6    0    0    2    4    0    0    2    2    2    4
252   0   2   2   2   2   0   0   2   2   0   0   2   0  ...    0    0    2    0    0    0    0    0    2    6    2    0    4
253   0   2   2   0   2   0   4   0   0   0   2   0   0  ...    2    4    0    0    2    0    0    2    2    2    0    2    0
254   0   2   0   0   4   0   0   0   0   0   0   4   0  ...    2    0    0    4    0    2    0    0    2    0    2    0    6
255   0   0   0   2   2   2   4   4   0   2   0   2   2  ...    2    0    2    0    0    2    0    0    4    4    0    6    0

[256 rows x 256 columns]
devansh@dev:~/Desktop/Crypto/tp$
```

The horizontal view



```
devansh@dev: ~/Desktop/Crypto/tp
File Edit View Search Terminal Help
devansh@dev:~/Desktop/Crypto/tp$ python3 ddt.py
The DDT for the Khazad Sbox is:
      0   1   2   3   4   5   6   7   8   9  10  11  12  ...  243  244  245  246  247  248  249  250  251  252  253  254  255
0   256   0   0   0   0   0   0   0   0   0   0   0   0  ...    0    0    0    0    0    0    0    0    0    0    0    0    0
1     0   0   4   4   4   2   0   0   6   6   2   6   0  ...    0    2    0    0    4    0    0    2    0    2    2    2    0
2     0   4   0   0   4   6   4   0   4   0   0   0   0  ...    0    0    0    0    2    2    0    2    0    2    2    0    0
3     0   4   0   0   2   0   0   2   0   0   4   0   2  ...    0    2    2    2    0    0    0    2    0    2    0    0    2
4     0   4   4   2   4   2   0   0   0   4   2   2   0  ...    2    0    2    2    0    0    0    0    0    2    2    4    2
5     0   2   6   0   2   8   0   6   0   2   0   4   0  ...    0    0    0    0    2    0    0    2    0    0    0    0    2
6     0   0   4   0   0   0   0   0   0   0   0   0   2  ...    2    2    0    0    4    0    0    0    2    0    4    0    4
7     0   0   0   2   0   6   0   0   4   0   4   2   4  ...    0    2    0    0    4    0    0    0    0    2    0    0    4
8     0   6   4   0   0   0   0   4   0   0   0   0   2  ...    0    0    0    0    2    2    0    2    4    2    0    0    0
9     0   6   0   0   4   2   0   0   0   0   2   6   0  ...    0    2    0    0    0    0    2    0    2    0    0    0    2
10    0   2   0   4   2   0   0   4   0   2   0   4   0  ...    0    0    2    0    0    0    0    0    0    0    2    0    0
11    0   6   0   0   2   4   0   2   0   6   4   0   0  ...    0    2    2    0    0    2    0    0    0    2    0    4    2
12    0   0   0   2   0   0   2   4   2   0   0   0   0  ...    6    0    0    2    0    2    2    0    2    0    0    0    2
13    0   0   0   2   2   4   0   0   0   0   0   0   0  ...    0    0    0    0    0    0    2    0    0    0    0    0    2
14    0   0   2   0   4   0   0   0   2   6   0   0   0  ...    2    0    0    0    0    0    0    0    0    0    2    0    2
15    0   0   4   2   4   0   6   2   4   0   2   4   2  ...    4    0    0    2    0    4    2    0    0    0    0    2    4
16    0   0   2   0   2   0   2   0   0   2   0   0   0  ...    0    2    0    0    0    2    4    4    2    0    2    2    2
17    0   0   6   0   2   0   4   0   0   4   0   0   0  ...    0    2    0    0    0    2    0    2    6    0    0    2    2
18    0   2   2   0   0   0   2   0   4   0   0   0   0  ...    0    0    0    2    4    4    0    2    0    2    0    2    0
19    0   0   6   0   2   0   0   4   0   0   0   4   0  ...    4    0    2    0    2    2    0    2    0    2    2    2    0
20    0   0   0   0   0   2   6   2   0   0   0   0   0  ...    2    2    0    4    2    4    0    0    0    0    2    2    2
21    0   0   0   4   0   0   0   0   2   0   0   0   2  ...    0    0    2    2    2    0    2    0    0    2    2    0    2
22    0   0   4   0   4   0   2   0   0   6   2   0   2  ...    0    2    0    2    0    2    0    0    2    0    0    4    2
23    0   0   4   0   2   0   2   2   0   0   0   0   0  ...    2    0    0    2    0    0    4    2    0    0    2    2    0
24    0   0   2   2   0   0   0   0   2   2   0   0   0  ...    0    0    2    0    0    0    4    0    8    0    4    0    0
25    0   0   4   0   0   0   2   2   2   2   0   0   2  ...    2    2    0    0    0    2    2    2    0    0    0    4    4
26    0   0   0   2   0   0   0   0   0   6   2   0   0  ...    0    4    0    0    0    2    2    2    0    0    0    2    2
27    0   0   0   0   0   0   0   0   0   4   0   0   0  ...    0    0    0    0    0    0    0    4    2    6    0    2    0
28    0   0   0   0   0   0   0   2   0   2   0   0   0  ...    0    2    0    2    2    2    0    0    0    2    6    4    4
29    0   2   0   0   0   0   0   0   2   2   0   0   0  ...    0    0    2    2    4    0    0    0    0    0    4    2    0
30    0   0   2   2   0   2   0   2   0   2   2   4   2  ...    0    2    2    2    0    0    0    0    2    0    0    2    2
31    0   2   2   0   4   0   0   2   8   0   2   2   0  ...    2    0    0    2    4    2    0    0    0    2    2    2    0
32    0   4   4   0   4   0   6   0   0   0   0   0   6  ...    0    0    2    0    0    0    0    0    2    0    0    0    0
33    0   0   2   2   6   4   2   0   0   6   2   0   2  ...    4    0    2    0    0    0    0    0    0    2    2    0    0
34    0   4   2   2   2   0   0   2   0   0   0   2   2  ...    0    0    0    2    6    0    4    2    2    0    0    0    2
```

The center view

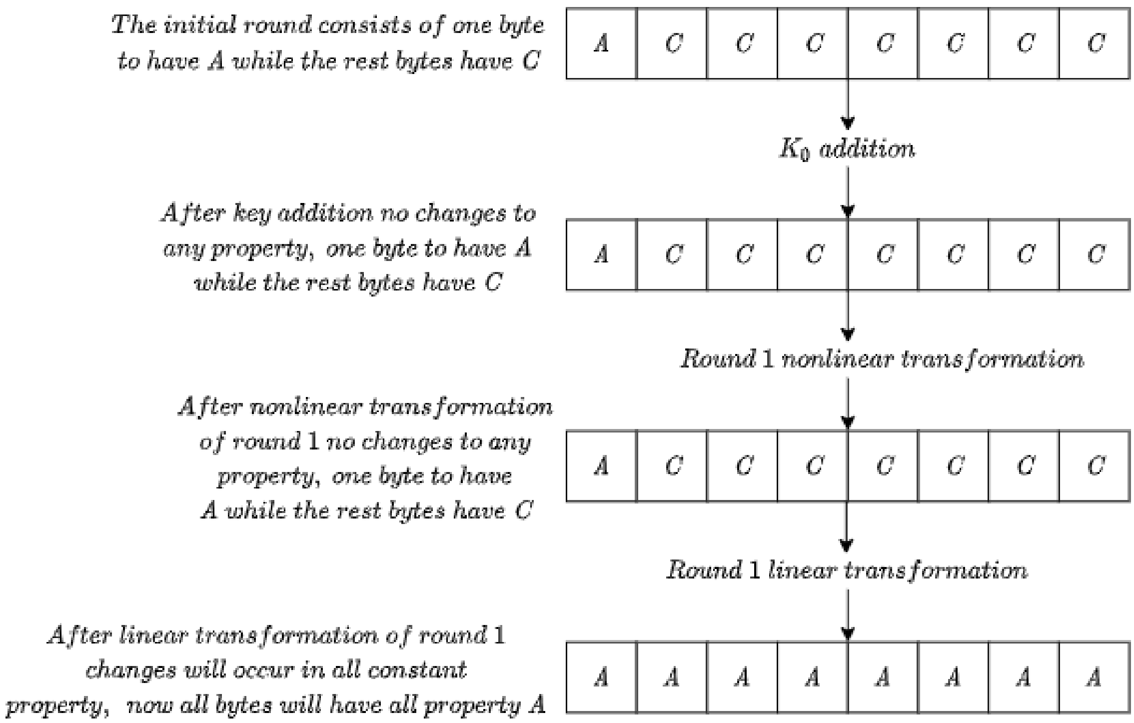
There were around 100 s-box transitions like 5 -> 5 , 4 -> 2E, 7 -> 86 having the best probability equal to $\frac{8}{16} = 0.5$. Any of the byte can be taken accordingly for differential attack.

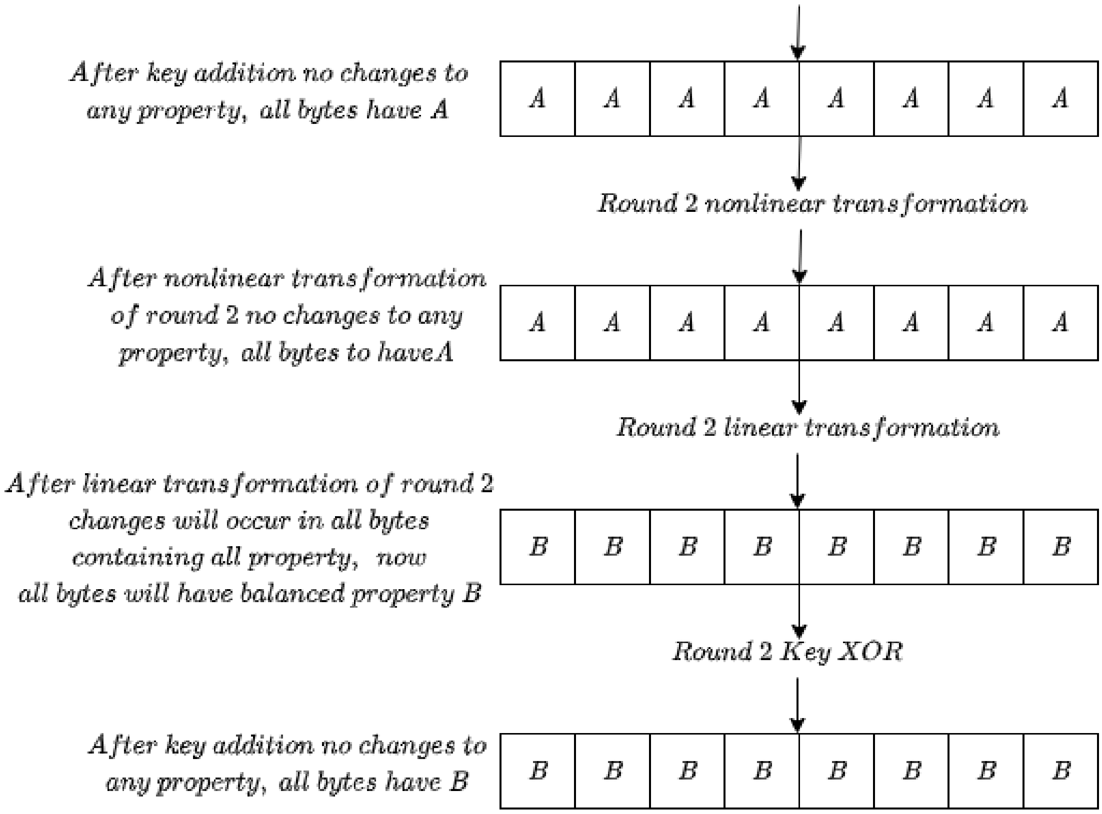
### 2.8.3 The Integral attack

Also known as the **The Square Attack**. The integral attack consists of the following properties. The attack is set on a 256 plaintexts, in such a way that first byte will take values ranging to all possible 256/FF values i.e. have all property while other bytes have a single value i.e. a constant values i.e. have constant property.

- **The All property :** The All property is the byte in which all values come once among the texts in the set. It is denoted by ***A***.
- **The Constant property :** The constant property refers to the byte in which all texts in the set have the same value. It is denoted by ***C***.
- **The Balanced property :** Also called the 0-sum property, the balanced property refers to the byte in which sum of all the texts in the set is ZERO. It is denoted by ***B***.

The integral attack on Khazad cipher is explained in the below figure :[Mul03]






So we can say that all bytes in our plaintexts will have balanced property after 2 rounds. Considering the fact that there will be no H-box or linear transformation in the last round, the subkey bytes can be separately guessed so to do a complete khzazad 3-rounds version attack here.
The complexity of this attack will be nearly $2^{16}$ sbox looks and $2^9$ plaintexts selections. Also we can increase this attack to one more round making ot 4 rounds by guessing the other subkey and this will increase time complexity by a large factor of $2^{64}$.
So far we have seen two variants of integral attacks on Khazad in which one is 3 round integral attack where complexity is :

| Attack Type | Rounds | Time | Space |
|---|---|---|---|
| integral attack-1 | 3 | $2^{16}$ | $2^9$ |

The second one is 4 round integral attack where we guess the other subkey.

| Attack Type | Rounds | Time | Space |
|---|---|---|---|
| integral attack-2 | 4 | $2^{80}$ | $2^9$ |

## 2.9 Other Attacks

### 2.9.1 Improved Integral attack

[BR00] The improved integral attack is basically an extension of integral attack up to 5 rounds and is having following time complexity.[Mul03]

| Attack Type | Rounds | Time | Space |
|---|---|---|---|
| integral attack-3(improved) | 5 | $2^{91}$ | $2^{64}$ |

With the use of key-scheduling algorithm and algebraic properties of round structure, this attack was made possible for 5 round KHAZAD in a very impressive time and space complexity.

### 2.9.2 Weak Keys Attack

[BR00] There are various new attacks on Khazad one of which is weak keys attack. The one more cryptanalytic result on Khazad version of 5-rounds is the of $2^{64}$ weak keys identified where it can be broken with around $2^{43}$ steps of lookups and analysis using $2^{38}$ encryption blocks.

| Attack Type | Rounds | Time | Space |
|---|---|---|---|
| Weak Keys | 5 | $2^{43}$ | $2^{38}$ |

### 2.9.3 Interpolation attacks

[BR00] This attack generally give benefit using the cipher components having simple algebraic structures like the structure of s-box etc.,that can be used as combined to give polynomial in feasible complexity.The way s-box of KHAZAD is created,and enforcing the effect of the diffusion layer, makes an attack of this type infeasible.

### 2.9.4 The boomerang attack

[BR00] The boomerang attack require ciphers whose encryption and decryption efficiency is different; this is not the case for KHAZAD, as it have involution structure.

## 2.10 Security

KHAZAD's crypto-resistance is equal to a block cipher with the given block and key lengths.
The motive of KHAZAD cipher was to achieve a K-secure and Hermetic secrecy.

Major security highlights of KHAZAD are :

- The most effective attack to find the KHAZAD cipher key is a full search.
- In KHAZAD, retrieving information about some Plain-Cipher text pairs from any given Plain-Cipher text pair is as efficient as using complete key search to determine the key.
- The approximate complexity of the key search by the full search method is directly dependent on the bit length of the key and is equal to $2^{127}$ applications of KHAZAD.

### 2.11 Key-Features

- It is observed that KHAZAD is much better than most of the modern available ciphers as far as compatibility is concerned.
- It is a very fast cipher and it avoids using excessive storage space for all of its code and tables.
- It does not have uncommon and expensive instruction built for a processor. This is the reason it is good for most platforms.
- The maths included in the creation algorithm is not complex and easy to understand.
- Since the key schedule is similar to the round function, we don't require any extra storage.

### 2.12 Brownie Point

- We implemented the key expansion algorithm and the code implementation of the cipher in python language which was not found anywhere and was solely done by us.
- We created several figures using `draw.io` which would be helpful for the people who wants to understand KHAZAD implementation algorithm and basic attacks on it. These figures were not available online and was solely done by us.

## References

[Bar] Paulo S. L. M. Barreto. The khazad block cipher. https://web.archive.org/web/20171011071731/http://www.larc.usp.br/~pbarreto/KhazadPage.html.

[BR00] Paulo Barreto and Vincent Rijmen. The khazad legacy-level block cipher. 01 2000.

[Mul03] Frédéric Muller. A new attack against khazad. In Chi-Sung Laih, editor, *Advances in Cryptology - ASIACRYPT 2003*, pages 347–358, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[Wik] Wikipedia. Wikipedia-khazad. https://en.wikipedia.org/wiki/KHAZAD.