Khazad-Block-Cipher

CodeRangers

Department of Computer Science
Indian Institute of Technology Bhilai

November 27, 2020

Outline

Origin

KHAZAD is a block cipher designed by Paulo S. L. M. Barreto and Vincent Rijmen, one of the designers of the Advanced Encryption Standard (Rijndael). KHAZAD is named after Khazad-dûm, the fictional dwarven realm in the writings of J.R.R. Tolkien.

It was presented at the first NESSIE workshop in 2000, and, after some small changes, was selected as a finalist in the project.

Introduction

KHAZAD has an eight-round substitution–permutation network structure similar to that of SHARK. The design is classed as a "legacy-level" algorithm, with a 64-bit block size and a 128-bit key.

KHAZAD makes heavy use of involution as sub-components which minimises the difference between the algorithms for encryption and decryption.

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-mini-blocks**
Implementation of the mixing matrix	**Involution MDS code**

Outline

Brief Overview

Khazad has an SPN structure. During encryption, it iterates 8 times a SP round function. Each of this 8 rounds consists of 3 stages (except the last round):

1. *Nonlinear Transformation* γ
2. *Linear Transformation* θ
3. *Adding a round key* σ

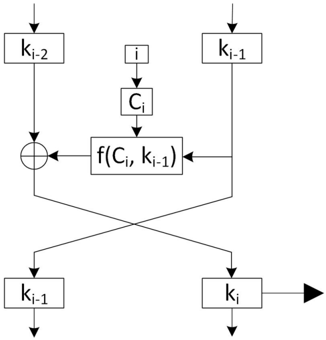
NOTE:-

The last round does not have a linear transformation layer.

Key Expansion

A 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 $k_0 \ldots k_8$ 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 $C_i^j = S(8i + j)$

Key Expansion



Round Function

Now lets see the Round function

General Round Structure

A single round consists of 3 stages:

1. *Nonlinear Transformation* (γ) : An sbox is applied in this layer to each byte of the current state.
2. *Linear Transformation* $(\theta\ )$: The state matrix is multiplied with a square matrix in GF $(2^8\ )$ of size 8
3. *Adding a round key* (σ) : The xor of the round key and the state matrix is taken in this stage.

Nonlinear transformation (γ)

Denoted as γ.

In each round, the input block is divided into smaller blocks of 8 bytes, which are independently subjected to nonlinear transformation (change), i.e. passed 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* $\gamma(\gamma(x)) = x$.

Linear transformation θ

Denoted by θ. 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 $x^8 + x^4 + x^3 + x^2 + 1$ $(0x11D)$.

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

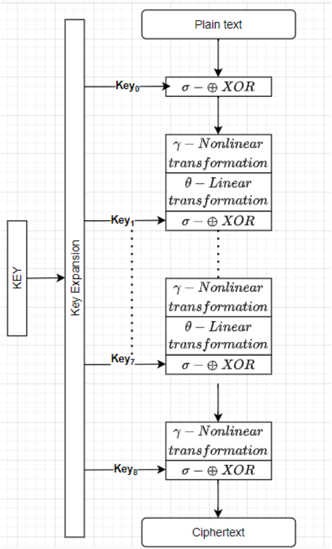
$$H = \begin{bmatrix} 01_x & 03_x & 04_x & 05_x & 06_x & 08_x & 0B_x & 07_x \\ 03_x & 01_x & 05_x & 04_x & 08_x & 06_x & 07_x & 0B_x \\ 04_x & 05_x & 01_x & 03_x & 0B_x & 07_x & 06_x & 08_x \\ 05_x & 04_x & 03_x & 01_x & 07_x & 0B_x & 08_x & 06_x \\ 06_x & 08_x & 0B_x & 07_x & 01_x & 03_x & 04_x & 05_x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

Adding a round key σ

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

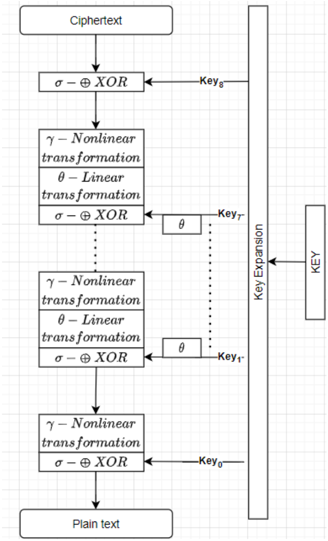
$$\text{For } i^{th} \textit{ round} \;:\; \sigma(x_i) = x_{i-1} \oplus k_{i-1}$$

Encryption Algorithm



The Encryption Algorithm

Decryption Algorithm



The Decryption Algorithm

Outline

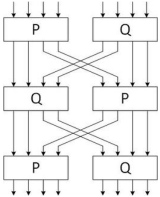
1. Introduction
2. Cipher Implementation
3. Observations
4. Brownie Point Nominations
5. Conclusion

The Substitution box

- In the original version of the cipher (KHAZAD-0) tabular replacement was represented by a classic S-block.
- 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).

The Substitution box

- Recursive structure of the replacement unit in the modified KHAZAD cipher:



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

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

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

S-box

The final KHAZAD S-box

	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$_x$

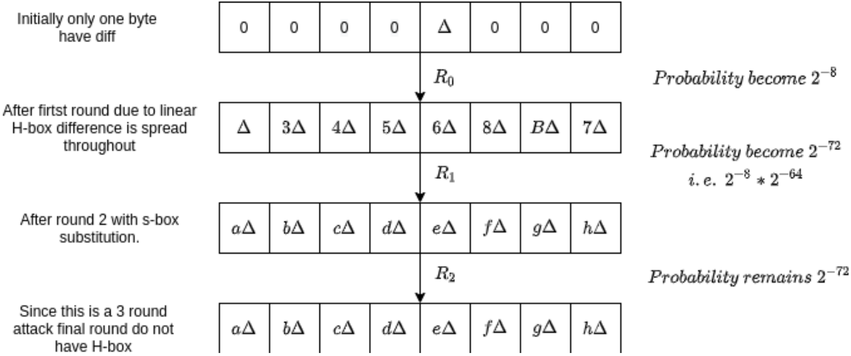
Attacks on Khazad

1. Khazad belong to group of ciphers which consists of Shark, Square, Rijndael, Anubis.
2. These were made in such a way that differential attack and linear attacks are not successful attacks for them.
3. It is very unusual to be successful for these ciphers on their full versions.

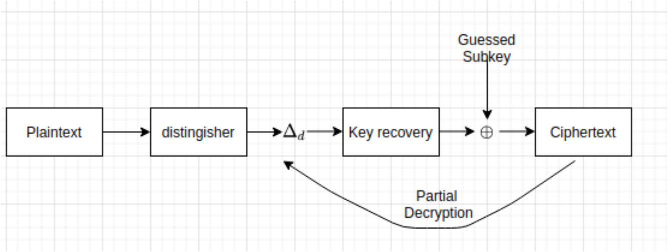
Differential attack

1. A differential attack exists for a 3 rounds Khazad cipher but its time complexity is very large as compared to 3 round integral attack.
2. Lets see the effect of each round on the message block due to different layers.

The 3 round differential attack



The 3 round differential attack

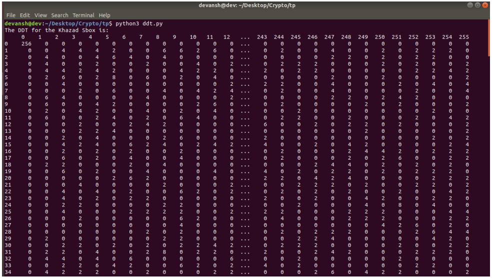


So after guessing 8 bytes of key or guessing subkey there would be at max 2^{64} possible guesses for 8 bytes of subkey, therefore the time complexity achieved would be 2^{64}

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

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$ 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
```

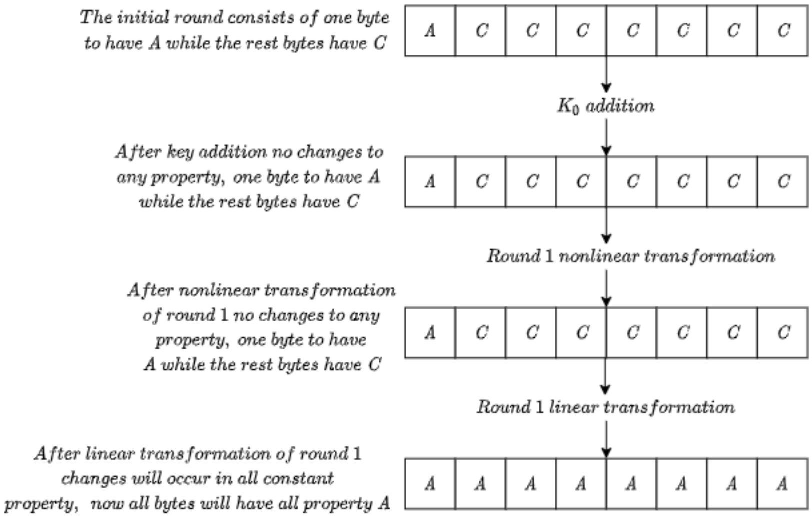
There were around 100 s-box transitions like 5 - 5 , 4 - 2E, 7 - 86 having the best probability equal to $\frac{8}{16} = 0.5$. Any of the byte can be taken accordingly for differential attack.

Integral attack

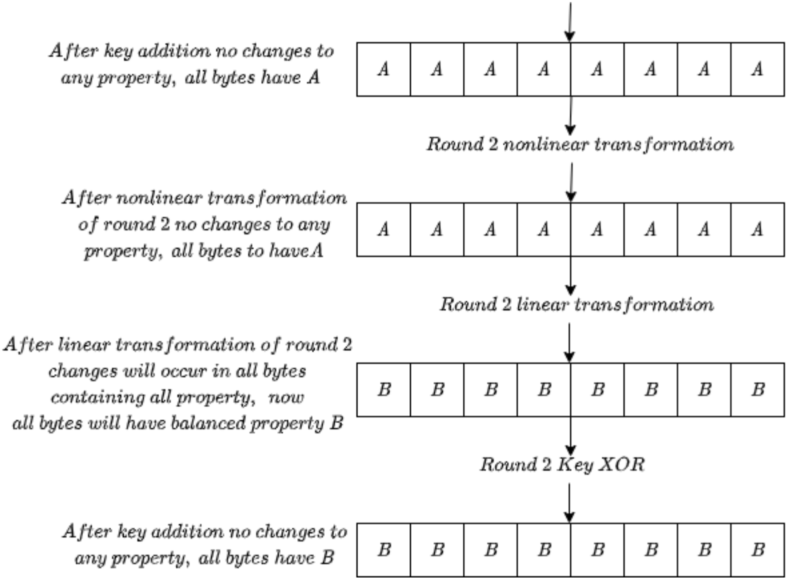
Also known as the **The Square Attack**. The integral attack consists of the following properties. The attack is set on a 256 plaintexts, such that the first byte takes all 256 possible values while other bytes have constant values.

- **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 $\boldsymbol{A}$.
- **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 $\boldsymbol{C}$.
- **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 0.It is denoted by $\boldsymbol{B}$.

The 3 round integral attack



The 3 round integral attack



The 3 round integral attack

1. All bytes in our plaintexts will have balanced property after 2 rounds.
2. No H-box or linear transformation in the last round.
3. sub-key guessed separately to do a complete 3-rounds attack here.
4. The complexity of this attack will be nearly 2^{16} sbox looks and 2^9 plaintexts selections. Also we can increase this attack to 4 rounds by guessing the other subkey and this will increase time complexity by 2^{64}.
5. The 3 round integral attack's complexity is:

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

The 4 round integral attack and beyond

The another variant is 4 round integral attack where we guess the other subkey.

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

- Improved Integral attack for 5 rounds.
- Weak Keys Attack
- Interpolation attack
- The boomerang attack

Outline

Brownie Point

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

Outline

Security

In terms of security:

- The most effective attack to find the KHAZAD cipher key is a full search.
- Retrieving any information about some Plain-Cipher text pairs using 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.

Key Features

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

Thanks

Team Members

- Swapnil Narad
- Devansh Chaudhary
- Aditya Kumar Susawat

Implementation Info

- Github Link: `https://github.com/swapnilnarad2000/Khazad-Block-Cipher.git`