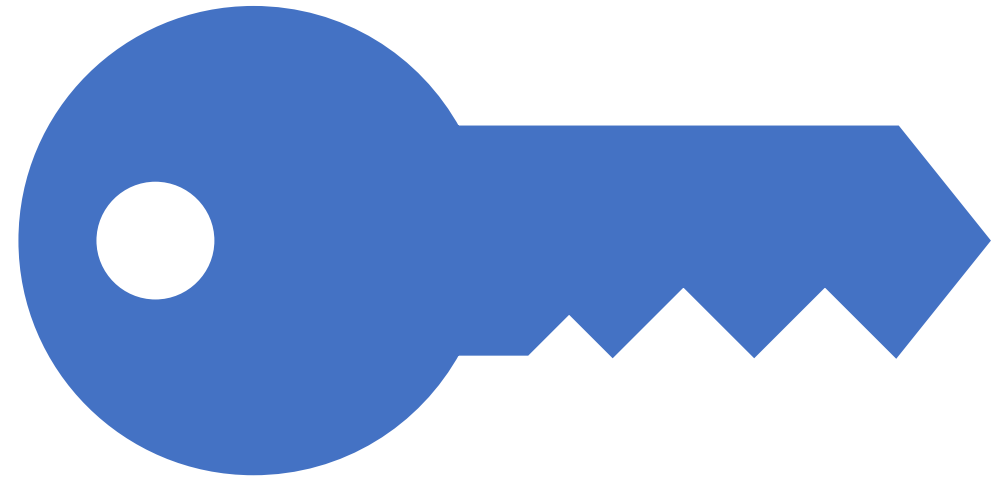





Ryan Sadler

A Study of Effectiveness of Historic and Modern Encryption Methods





What is this Study About?

This study aims to improve the data available to others about the security and effectiveness of various historic and modern algorithms for encrypting data. It should also be user accessible with a good UI for easy use by anyone.

Research Background

By far the best paper for this study was: [\(Rizvi, S.A.M, Hussain, S.Z. and Wadhwa, N., 2011, pp.76-79\)](#) as this was the only paper that I was able to find that detailed audio encryption as well as images and text. It also features detailed descriptions of TwoFish and AES.

The second most helpful paper was definitely: [\(Yadav and Majare, 2016, pp. 70-73\)](#) as it helped me detail how I would score the algorithms, based on Throughput, Key Size, Encryption and Decryption Speed and Time, Encryption Ratio and Level of Security Issues. However, there is no details on image or audio encryption.



Methodology

This project is being developed using the agile model, as it will be built and tested, piece by piece until it is all fully completed. For version management, GitHub will be used as it is very good managing different versions of software and I am able to roll back code if I ever encounter any major issues.



Brief Description Of Algorithms

- Caesar Shift – With a key (n) between 1 and the length of the alphabet, shift characters right in the alphabet n times. For example, using the standard alphabet and a key of 3, the plaintext “encode” becomes “hqfrgh”
- Substitution cipher – This uses a shuffled version of the alphabet as a key, and you simply swap characters with what is in their position in the key. For example, with a key of “ecbadfghjklmnrqspwvuzxty”, the plaintext “encode” would become “dobrad”.

Original Message: Hello World

Encrypted Message: Horel ollWd

- A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
B B C D E F G H I J K L M N O P Q R S T U V W X Y Z A
C D E F G H I J K L M N O P Q R S T U V W X Y Z A B
D D E F G H I J K L M N O P Q R S T U V W X Y Z A B C
E E F G H I J K L M N O P Q R S T U V W X Y Z A B C D
F F G H I J K L M N O P Q R S T U V W X Y Z A B C D E
G H I J K L M N O P Q R S T U V W X Y Z A B C D E F
H H I J K L M N O P Q R S T U V W X Y Z A B C D E F G
I I J K L M N O P Q R S T U V W X Y Z A B C D E F G H
J J K L M N O P Q R S T U V W X Y Z A B C D E F G H I
K K L M N O P Q R S T U V W X Y Z A B C D E F G H I J
L L M N O P Q R S T U V W X Y Z A B C D E F G H I J K
M M N O P Q R S T U V W X Y Z A B C D E F G H I J K L
N N O P Q R S T U V W X Y Z A B C D E F G H I J K L M
O O P Q R S T U V W X Y Z A B C D E F G H I J K L M N
P P Q R S T U V W X Y Z A B C D E F G H I J K L M N O
Q Q R S T U V W X Y Z A B C D E F G H I J K L M N O P
R R S T U V W X Y Z A B C D E F G H I J K L M N O P Q
S S T U V W X Y Z A B C D E F G H I J K L M N O P Q R
T T U V W X Y Z A B C D E F G H I J K L M N O P Q R S
U U V W X Y Z A B C D E F G H I J K L M N O P Q R S T
V V W X Y Z A B C D E F G H I J K L M N O P Q R S T U
W W X Y Z A B C D E F G H I J K L M N O P Q R S T U V
X X Y Z A B C D E F G H I J K L M N O P Q R S T U V W
Y Y Z A B C D E F G H I J K L M N O P Q R S T U V W X
Z Z A B C D E F G H I J K L M N O P Q R S T U V W X Y

- Vigenère Cipher – This uses a table called the tabula recta (above), a table of Caesar shifts each with a key 1 more than the last. The key can be a random assortment of letters, though it is more common to use a word, as it is more memorable.

Brief Description Of Enigma

The enigma machine depends on 4 main things:

- Plugboard Settings – What letters are swapped on input/output
- Rotor Positions – The position the rotors are put into the machine
- Key Settings – Acts as a caesar shift on one side of the rotor
- Rotor Alignments – Acts as a caesar shift on the other side of the rotor



Brief Description Of RSA

The main complexity of this algorithm comes from the key generation.

The public key is (e,n) and the private key is (d,n) . The numbers required for 1024-bit RSA are below:

- p and q – two 1024-bit prime numbers
- n – the product of p and q
- $\phi(n) - (p - 1) * (q - 1)$
- e – a number that is coprime with $\phi(n)$
- d – a number where $(e * d) \text{ MOD } \phi(n) = 1$

To encrypt data you simply convert the plaintext into an integer (m) and to get the ciphertext, the equation is $c = m^e \% n$. To decode the ciphertext, the equation is $m = c^d \% n$.

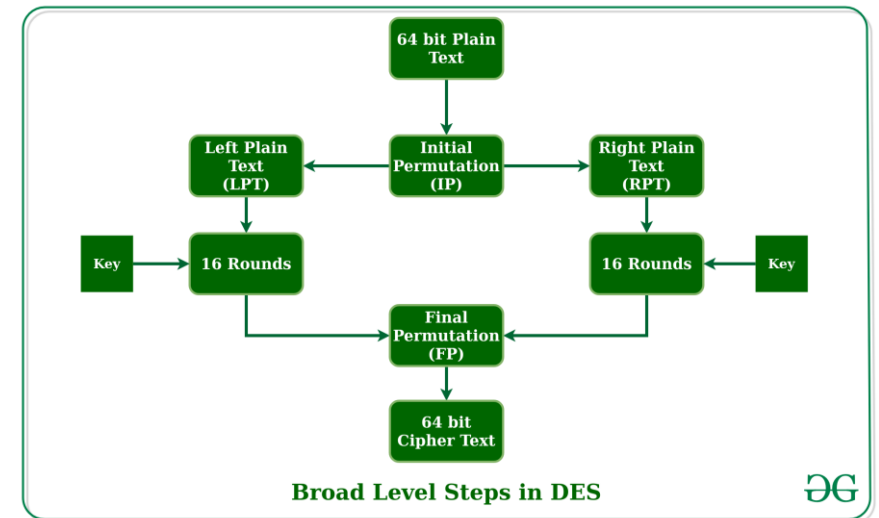
Brief Description Of DES/3DES

DES works using a variety of sub-boxes and XORS to encode data in 64-bit blocks using a 64-bit key.

3DES is exactly the same but with 2 or 3 keys, and you encode 3 times, using all of the available keys.

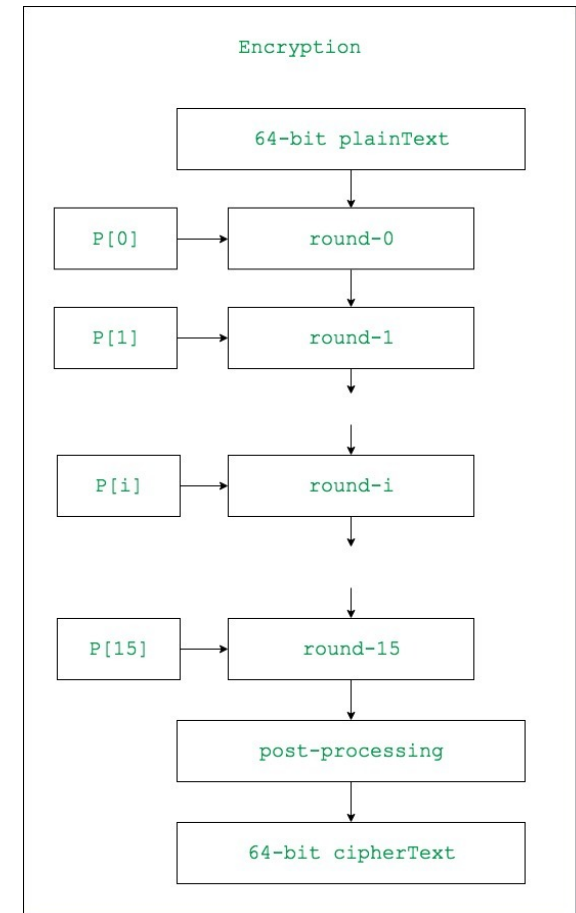
Example (and picture from) at:

<https://www.geeksforgeeks.org/data-encryption-standard-des-set-1/>



Brief Description Of Blowfish

Blowfish was originally designed by Bruce Schneier as a substitute/replacement for DES. The key can be anywhere from 42 bits to 448 bits, and is expanded using a substitution box generated by the digits of pi. Then you encode the plaintext using similar steps to DES, with an additional post-processing stage at the end, where you XOR each half of the plaintext with the last 2 expanded keys.

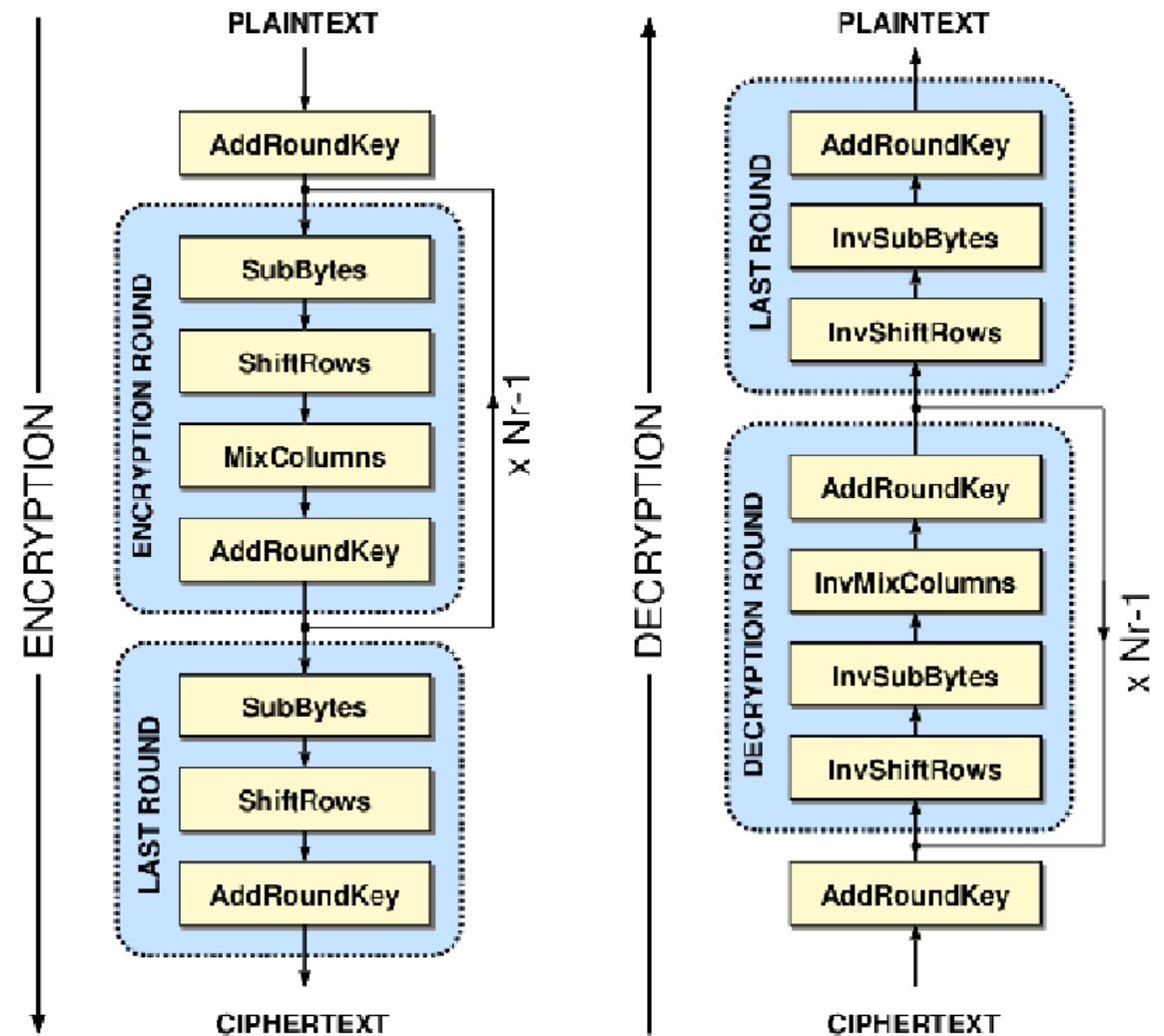


Brief Description Of AES

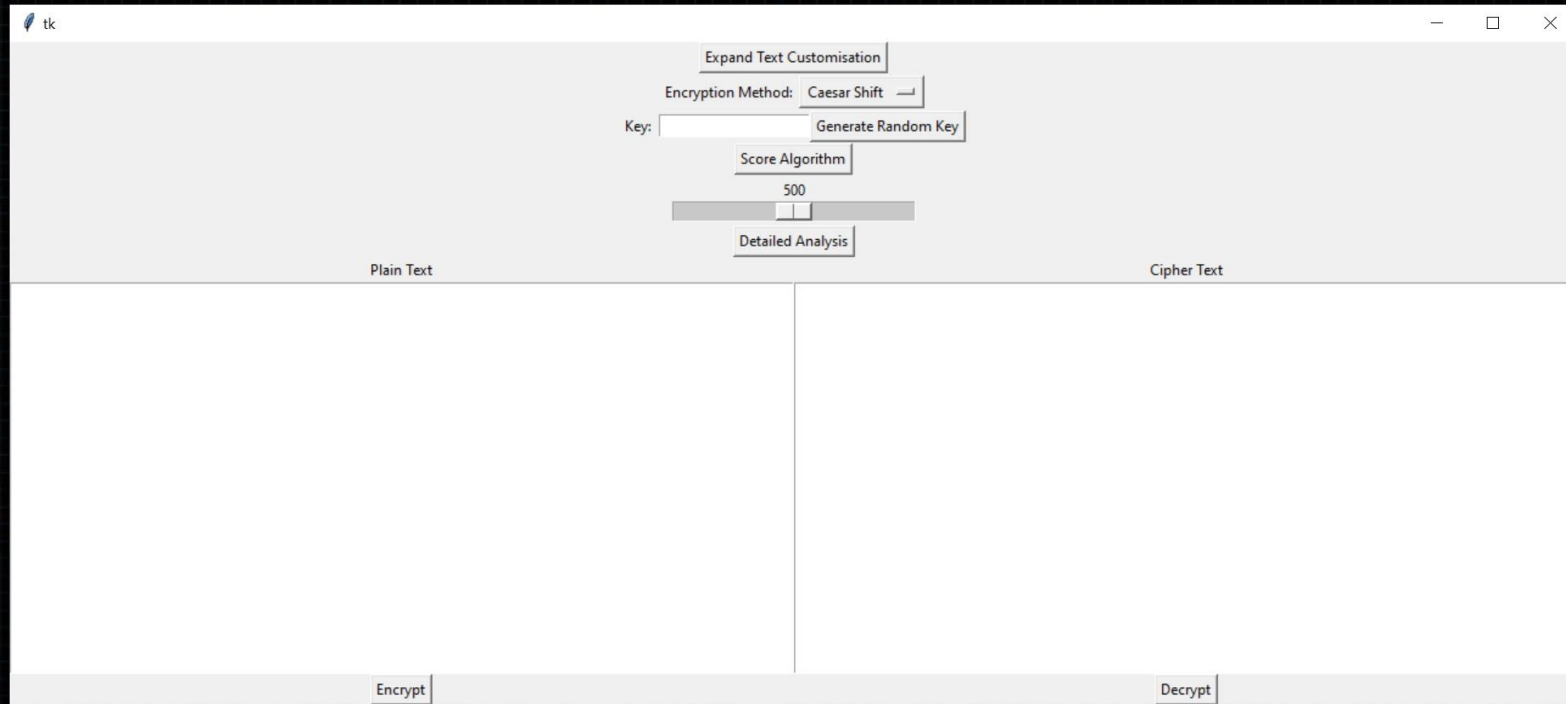
To encode with AES-128, you simply need a 128-bit key, which is expanded using a series of sub-boxes and a key schedule.

Then you perform rounds on the plaintext using the functions featured on the left.

To decode, you reverse the keys, and perform the inverse counterparts of the original functions in reverse order.



Video of Software



Screenshots of Reports Generated

```
Average Values
-----
Key Size: 120 bytes
Key Generation Time: 0.0019373178482055664 seconds
Encryption Time: 0.027068710327148436 seconds
Decryption Time: 0.02726747989654541 seconds
Encryption Ratio: 1.0
-----

Cycle 1
-----
Key Size: 120 bytes
Key Generation Time: 0.003974437713623047 seconds
Encryption Time: 0.027317047119140625 seconds
Decryption Time: 0.02880692481994629 seconds
Encryption Ratio: 1.0
-----

Cycle 2
-----
Key Size: 120 bytes
Key Generation Time: 0.0019867420196533203 seconds
Encryption Time: 0.026818513870239258 seconds
Decryption Time: 0.026820659637451172 seconds
Encryption Ratio: 1.0
-----

Cycle 3
-----
Key Size: 120 bytes
Key Generation Time: 0.0014896392822265625 seconds
Encryption Time: 0.027317523956298828 seconds
Decryption Time: 0.028310298919677734 seconds
Encryption Ratio: 1.0
-----

Cycle 4
```

Algorithm-Specific Report

```
Key Generation Time
-----
1. Rail-Fence Cipher 0.0003641772270202637 seconds
2. Vigenere Cipher 0.0003724884986877441 seconds
3. Substitution Cipher 0.0011868000030517579 seconds
4. Caesar Shift 0.001308131217956543 seconds
5. DES 0.0014104604721069335 seconds
6. Enigma 0.0014106154441833497 seconds
7. AES 0.0015048766136169433 seconds
8. Triple DES 0.0015645027160644531 seconds
9. Blowfish 0.0017477107048034669 seconds
10. RSA 1.4334786081314086 seconds

Encryption Time
-----
1. Caesar Shift 0.00036745071411132814 seconds
2. Substitution Cipher 0.0004056310653686523 seconds
3. Vigenere Cipher 0.002259845733642578 seconds
4. Rail-Fence Cipher 0.007438099384307862 seconds
5. Enigma 0.027852845191955567 seconds
6. DES 0.2571110320091248 seconds
7. Blowfish 0.29335062742233275 seconds
8. AES 0.6620354294776917 seconds
9. Triple DES 0.7622636246681214 seconds
10. RSA 1.007973208427429 seconds
```

Detailed Report

Current Issues

As of completion, there is only one issue with this code, which is that when it comes to decrypting with RSA, python does not handle reading in massive numbers very quickly, so as you can see by the screenshot provided, the decryption time is severely impacted and produces inaccurate results.

Average Values

```
-----  
Key Size: 59.0 bytes  
Key Generation Time: 1.2927618741989135 seconds  
Encryption Time: 0.0810093879699707 seconds  
Decryption Time: 22.690035915374757 seconds  
Encryption Ratio: 597.725  
-----
```

Source Code

<https://github.com/ryanbutbored/ComputingProject>

Bibliography

1. Rizvi, S.A.M, Hussain, S.Z. and Wadhwa, N. (2011) 'Performance Analysis of AES and TwoFish Encryption Schemes' 2011 International Conference on Communication Systems and Network Technologies, Katra, India, pp. 76-79. DOI: [www.doi.org/10.1109/CSNT.2011.160](https://doi.org/10.1109/CSNT.2011.160)
2. G. Yadav and A. Majare (2016) 'A Comparative Study of Performance Analysis of Various Encryption Algorithms' *International Journal on Recent and Innovation Trends in Computing and Communication*, 5 (3) pp. 70-73 Available at [https://ijritcc.org/download/conferences/ICEMTE_2017/Track_2_\(EXTC\)/1487794878_22-02-2017.pdf](https://ijritcc.org/download/conferences/ICEMTE_2017/Track_2_(EXTC)/1487794878_22-02-2017.pdf)
Accessed: 04/10/2023
3. Design and Implementation A different Architectures of mixcolumn in FPGA - Scientific Figure on ResearchGate. Available at: https://www.researchgate.net/figure/The-basic-AES-128-cryptographic-architecture_fig1_230853805