Image Encryption Using FEAL Cipher Encryption

Didier Lessage and Robert Veira

Abstract—FEAL encryption presents a fast and efficient method of encrypting data that relies on multiple keys as well as breaking down an image into sections. Work has been done both in encryption of color images and gray scale images. One paper in particular written by Nithin, GBongale, and Hegde report their own favorable results using the FEAL encryption method. We set out to emulate their encryption approach and determine whether their results can be replicated.

Index Terms—FEAL, Image Encryption, OpenCV.

1 Introduction

At the beginning of encryption, users simply had to worry about encrypting textual messages. However, with present times, users are sharing much more than simple text messages and text based emails. Photos are being shared through email attachments and social media sites such as Facebook, Instagram, and Twitter. Users want a way to ensure the images they share solely are viewed by their loved ones, limiting the amount of exposure these images may receive from strangers.

There are several different approaches to encrypting an image. Most of these encryption methods rely on the concept of one or two keys for encrypting and decrypting. Encrypting an image is done through one key, while either the same key or a second key (depending on the method of encryption) is used to decrypt the image. This method means that the intended receiver knows what kind of encryption was used on the received file or message. The security comes into play due to potential attackers having no knowledge or idea of the type of encryption used on the file. For the attacker, opening up the image can result in either a different image displayed or the file itself not even being recognized as an image file type.

FEAL encryption works on the same principle, albeit with some caveats that make it unique from other encryption and decryption methods. Possibly of most note is its utilization of multiple keys. Rather than using a single key to decrypt a message, FEAL relies on multiple keys, groups of which are used for different stages of the encryption. For instance, if 12 keys are generated to decrypt and encrypt the message, four could be used to concatenate the image, four could be used to generate cipher operation on the bytes, while the final four could be used to generate the cipher text. This would be the encryption stage. In order to decrypt the image, one would need to do the same process, only in reverse.

FEAL is not restricted to just image encryption. FEAL can also be used to encrypt and decrypt text files as well as image types. Due to FEAL's implementation of manipulating raw bytes, the actual file type would not be as big of a concern as other, more specifically targeted encryption models.

FEAL encryption has been shown to not be as secure as the more typical DES encryption method. In some cases, it has been shown to be much easier to crack than comparable DES methods. During its earlier stages of development, papers such as Bert's Cryptanalysis of F.E.A.L[?] revealed weaknesses within the FEAL method of encryption.

2 Tools

As far as the tools we used to handle the image processing, we decided first to program in C++. This decision was made primarily due to familiarity with the language. As far as image processing, we opted to go for the OpenCV library. Throughout our research, the biggest common suggestion in terms of handling images in C++ was OpenCV. OpenCV provided built in ways to take chunks of an image, as well as perform bitwise operations on portions of the image. In addition, OpenCV also has tools to display and save images, which would be incredibly useful for showing the encrypted and decrypted images, as well as saving out the images for analysis.

3 Goals

Our intention is to implement the FEAL algorithm brought up in Nithin, Bongale, and Hegde[2], and attempt to achieve similar results to theirs. Most importantly, can we encrypt and decrypt an image using the FEAL algorithm provided we have the correct key. We also wished to compare our results to Hegde as well as the original Shimizu and Miyaguchi[1] paper which introduced FEAL. Our questions we wished to answer are whether or not FEAL can be used to encrypt/decrypt as well as are Hegde's provided descriptions accurate in their implementation of the FEAL algorithm. We will document any discrepancies we find between their implementation and ours as well as provide examples of our algorithm encrypting and decrypting images.

4 Implementation

As far as implementation, we decided to create a suite of utility functions to represent each stage of the data randomization and key scheduling. After analyzing the descriptions of the two sides of FEAL, we determined FEAL could be divided into the following sections: F and F_k, the SBox function, the input and output gate, the data randomization (DRE) and the creation of the L and R portions of the image. When combined together along with the key scheduler's provided key, encryption and decryption were possible. Figures 1 and 2 show a common representation of the FEAL algorithm for both decryption and encryption.

For the manipulation of the bytes of each image, we took advantage of the Mat object provided by OpenCV. What this allowed us to do was grab specific parts of the image (rows and columns) and feed them into their own Mats which could then be manipulated through XOR functions as well as swapping values during encryption and decryption.

To step through our process of implementing the FEAL algorithm, an image was first passed to our executable to be encrypted and decrypted. The image was then split into 8 byte sections. Each of

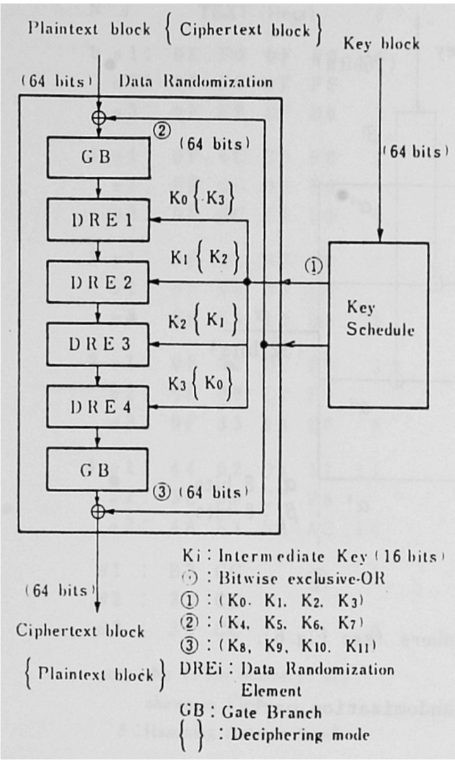

Fig. 1: Diagram of the overall FEAL implementation

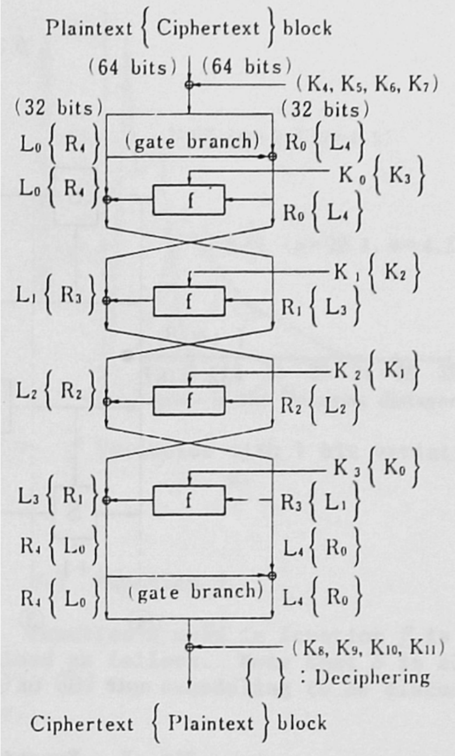

Fig. 2: An inside look of the DRE implementation

these sections was then fed into our DRE function, which first split the image into a left and right half. This can be seen through Figure 2's graphical illustration of the DRE. After passing through a gate branch which simply performed an XOR on both halves and passed the result to the right half, we moved these image chunks through four identical data randomizations. Within each of these was an S box function which would take keys 0 through 3 and XOR them with the image chunk in each DRE pass. The final pass would then move the chunks through a final gate branch before combining them into the encrypted or decrypted image.

Whether we are currently encrypting or decrypting determined which side of the DRE the halves would begin, as well as the final swap after the final gate branch.

5 Results

Using both papers during our implementation, we were able to successfully perform FEAL encryption on several 256 by 256 images. However, some discrepancies were found within the Nithin, Bongale, and Hegde[2] paper which lead us to believe that either the paper was unintentionally misleading or some aspects of FEAL were not fully understood by the writers.

For one, the paper claims to have split a 256 by 256 grayscale image into 12 16 by 16 chunks. However, through some analysis, it can be seen that when such an image is split into 16 by 16 chunks, the resulting number of chunks is far greater than 12.

Second, it appears that the splitting of the image into such large chunks is not necessary. Splitting the image into 8 byte chunks is more than enough to conduct a FEAL encryption and decryption of the image. One benefit we can see with splitting the image into chunks is through the usage of multi-threading. With multi-threading and large images, the encryption and decryption process would be greatly sped up.However, for a small 256 by 256 image, speed and bogging down of the system is nowhere near a realistic concern.

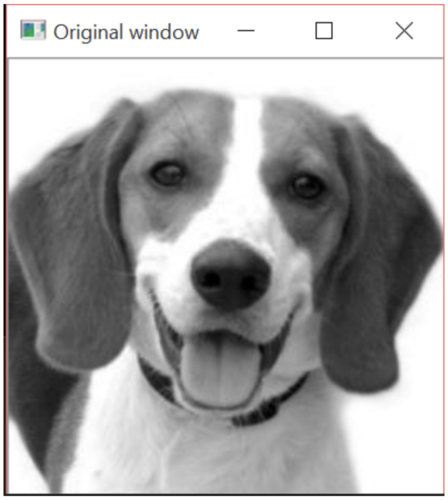

Fig. 3: Original grayscale dog image

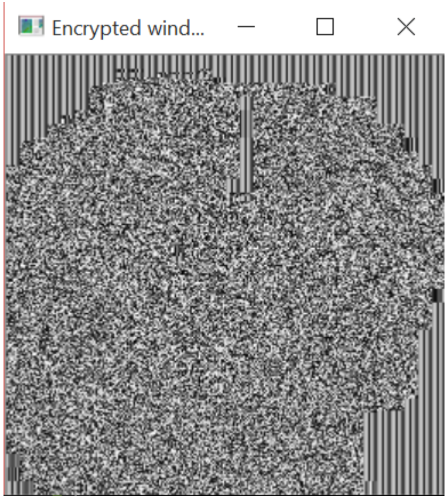

Fig. 4: Encrypted dog image

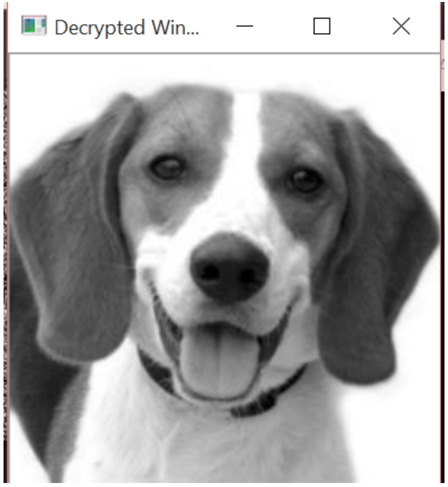

Fig. 5: Decrypted Dog Image

We did also encounter some issues with our own implementation of FEAL. More specifically, it was difficult to ascertain the exact implementation of FEAL based off of the papers alone. A good deal of debugging was needed to find how exactly each byte would be manipulated and swapped to achieve true decryption and encryption. In certain instances, the decrypted image would be close, but with each column swapped backwards, resulting in a shifted image from the original. Issues like this coupled with the sparse explanation of FEAL in

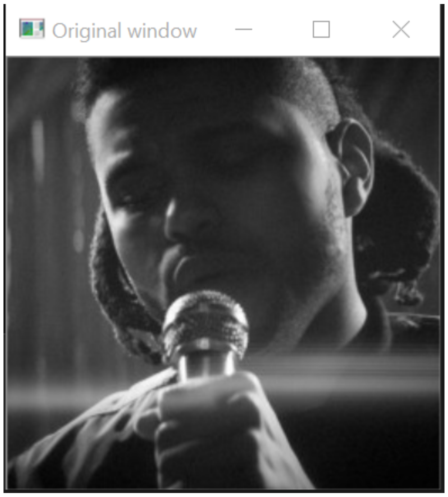

Fig. 6: Original grayscale face image

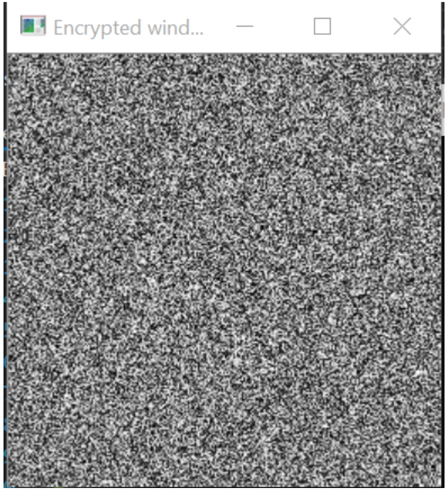

Fig. 7: Encrypted face image

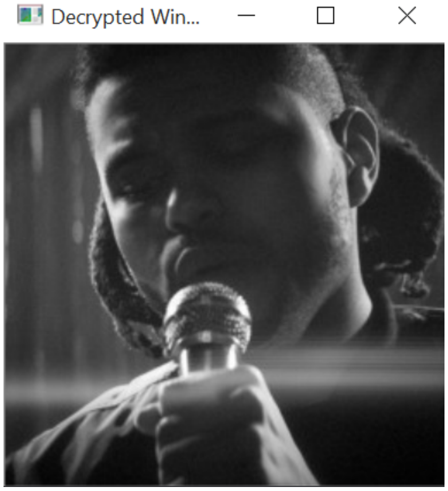

Fig. 8: Decrypted face image

the past caused a good deal of trouble in creating a successful FEAL encryptor and decryptor.

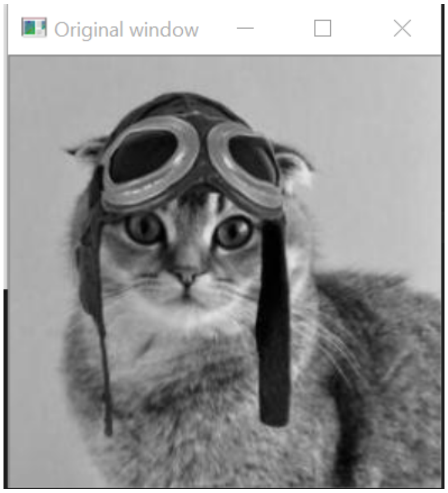

Fig. 9: Original grayscale cat image

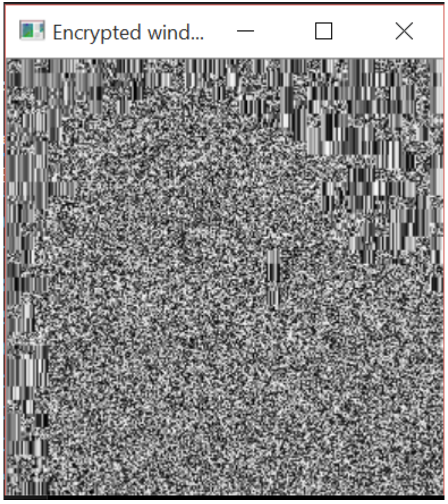

Fig. 10: Encrypted cat image

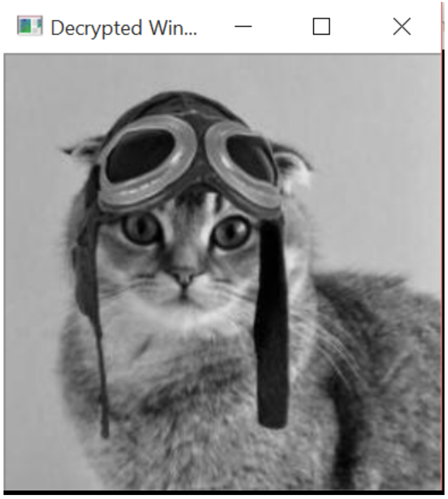

Fig. 11: Decrypted cat image

6 Conclusion

The encryption of a grayscale image with FEAL is indeed a possibility. However, the actual safety and practicality of FEAL has been proven by others to not be as safe as it was originally proposed.

7 Future Work

Our implementation only focused on encrypting and decrypting grayscale images of a specific size. In terms of further work on FEAL, it is plausible that larger images could be worked on, as well as images of color. This broadening of FEAL would only require other bytes to be manipulated in the RGB scale of the color image, or for the written code for the encryption and decryption to be designed in a fashion that was ignorant of the actual size of the passed in image. This generality of the code means that FEAL can be applied to a multitude of image types as well as other file types. FEAL simply cares about receiving an even number of bytes to split inside of its data randomization implementation.

Also, an increase in the number of keys or the complexity of the keys themselves could assist in the discovered weakness of FEAL encryption and decryption.

References

[1] Nithin N, Anupkumar M Bongale, G. P. Hegde *Image Encryption based on FEAL algorithm*. International Journal of Advances in Computer Science and Technology, Manipal, Karnataka, India, 2013.

[2] Akihiro Shimizu, Shoji Miyaguchi *FEAL - Fast Data Encipherment Algorithm* Systems and Computers In Japan, Vol. 19, No. 7, 1988

[3] Bert den Boer *Cryptanalysis of F.E.A.L.* Advances in Cryptology-EUROCRYPT '88, 293-299, 1988