A Critical Review Of Matrix-Based Secret Sharing for Reversible Data Hiding in Encrypted Images

Thomas Ingram

December 8th, 2023

1 Abstract

Reversible Data Hiding in Encrypted Images (RDHEI) is a crucial aspect of computer security, especially in the era of cloud computing and communication. This paper is a critical review of a matrix-based secret sharing scheme for RDHEI. The proposed scheme utilizes a (**r**,**n**)-threshold matrix-based secret-sharing technique, providing redundancy and security against data hider and other points of failure. The encryption process involves block scrambling, matrix-based encryption, and a novel Block Error Mixture Encoding (BEME) method to enhance embedding capacity. The paper discusses the implementation details, including matrix generation, image encryption, and data hiding. Furthermore, the post-processing steps for decoding and recovering the original image are explained. The strengths and weaknesses of the proposed scheme are thoroughly analyzed, highlighting its potential applications in fields like healthcare.

2 Introduction

Reversible Data Hiding in Encrypted Images (RDHEI) is currently a very prominent research topic in the field of computer security. When broken down RDHEI is made up of two main components Reversible Data Hiding (RDH) and Image Encryption. RDH is a technique of embedding data into a medium such as images, audio, or video in such a way that the original signal can be perfectly reconstructed after the embedding. Image Encryption then extends the RDH by supplying a medium for the data to be hidden within that can also contain sensitive information itself. RDHEI has become increasingly prominent due to the rise of cloud storage and computing as a means of communication. RDHEI has many practical uses because of the aforementioned increase in reliance on the cloud for many aspects of our daily lives. This rise in use also requires concerns around security and vulnerabilities especially when it comes to systems that pertain to users' private information. Some examples include medical information as well as financial information not all of this data can be represented with simple encryption thus more steps need to be taken to keep this information safe.

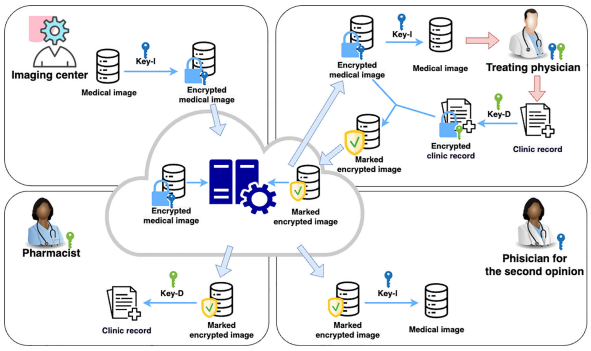

Figure 1: Data management with RDHEI for healthcare.

If we take healthcare for example, it is a field that could see heavy use of this technology. Medical documents are used in tandem with practitioners' comments and instructions. RDHEI can be used to centralize this information pertaining to specific documents and allow for certain users to access parts of the data on a need-to-know basis. For example, A CAT scan image could be encrypted with one key and embedded with Doctor A's remarks on it with another key then passed to Doctor B who could further add a prescription. A Pharmacist could receive the image and use the encryption key for the embedded data to recover the remarks without ever having to fully decrypt the image maintaining the data safety of the patient while keeping all relevant information in a centralized and portable format.

RDHEI as a concept was first proposed in 2008 by Puech et al.[2] and has continued to rapidly

evolve since then with many other schemes looking to iterate on and improve the concept by trying to avoid the weaknesses in embedding capacity. There are two general approaches to RDHEI: reserving room before encryption (RRBE) and vacating room after encryption (VRAE). RRBE is normally performed by the image owner in the initial stages and is thus an expensive process, especially for less performant devices. Whereas VRAE operates on the already encrypted image by embedding directly into the encrypted image and the vacation is performed by the data hider. This then has a lower initial cost on the image owner's end as once the image encryption is done the process can then be offloaded to cloud computing as all sensitive data from then on out will be safe. The downside is the embedding capacity is more limited due to the restrictions of operating in encrypted space.

This report will go through and discuss the methods used in the proposed paper and the merits and drawbacks of each approach.

3 Discussion

The scheme discussed in this paper is a VRAE-based RDHEI scheme based around a Matrix-based secret-sharing technique. Traditionally to retain pixel redundancy for the data embedding after encryption lightweight encryption approaches are used such as an exclusive-or (XOR) operation in addition to block scrambling and co-modulation. The downfall of the approach is that it is simply insecure even with scrambling and co-modulation. Taking a heavier approach to encryption requires more compute power, the exact gain VRAE looks to leverage. In the paper discussed, It proposes a (**r**,**n**)-threshold matrix-based secret sharing scheme using matrix theory. This approach will allow for the use of multiple data hiders to avoid a single point of failure of one data hider. This will be done by utilizing the threshold matrix to encrypt the original image into multiple images (**n** images) and will send each to a single different data hider with high data redundancy. Furthermore, the method utilizes a block error mixture encoding (BEME) method to encode the images in the encrypted domain. The proposed BEME encoding is able to exploit the high data redundancy and can vacate a large amount of room for embedding. The receiver will need only **r** of the **n** encrypted images to recover the secret as well as the original image while also being able to recover the encoded message. Additionally, with the data redundancy of **n** matrices, it can withstand n-r points of failure in conjunction with its embedding capacity and image security.

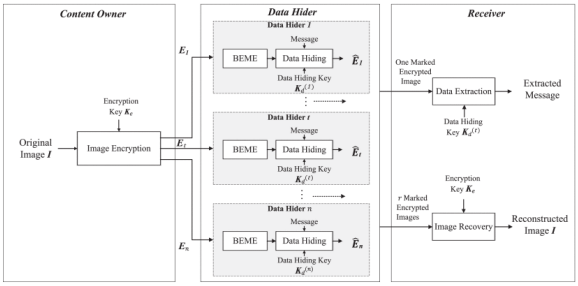

Figure 2: Overview of the proposed RDHEI.

Image encryption is quite a simple process that is fast and works incredibly well for this problem. To do encryption you first perform block-based scrambling where you break the image blocks into small chunks of **B** size that are defined by the user as parameters. these blocks are scrambled into an order generated by the encryption key such that it is a reversible process. Then for each individual block, it is encrypted by constructing an (r,n)-threshold matrix through the process outlined in the paper. once again it is seeded by the encryption key such that the matrix is reproducible. In my implementation, I used the offset of the block index to seed my PRNG generator for the sets used in the creation of the matrix such that I have reproducible but sufficiently random sets. for each block, you multiply each pixel and the data within the set with the pixel to create your encrypted shares.

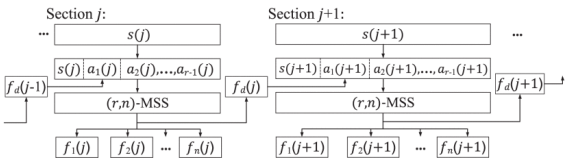

Figure 3: Share Generation

Once each pixel has its individual set of encrypted shares it is broken out such that each element in the **n**-element set is used as the new pixel value at the original pixel's location in the nth image. After reconstructing the n images by using the different pixel values we have our encrypted images.

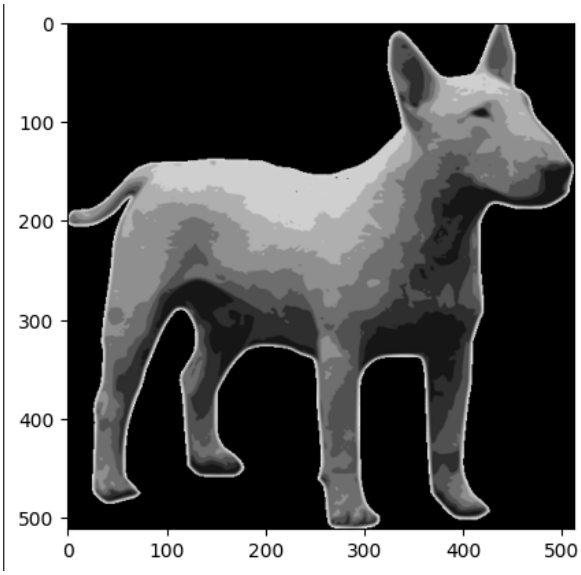

Figure 4: Original Image

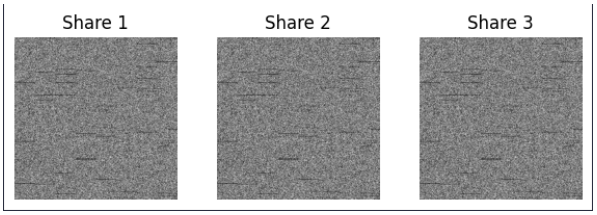

Figure 5: Encrypted Image

This is a very novel approach for image implementation and works incredibly well for the purpose as well. It not only contributes greatly to the cause of redundancy but allows for incredibly fast image encryption while accounting for failure points of the data hiders. The method further shows its merits when you take into account its reliance on the encryption key as well. Though this does add an extra layer of friction when it comes to implementation, a lot of the methods are heavily dependent on the implementation. Any small changes in implementation such as PRNG seeding and random number generation cause incongruities resulting in massively different results possibly making it such that encryption with one is not equivalent to encryption with another with a small change to processes mentioned. The paper uses a two-dimensional sine chaotification system[4] (2D-SCS) for random number generation which doesn't have a way to seed for PRNG which makes the requirement of reproducibility difficult.

After image encryption post-processing must be done on the image such that decryption is possible without requiring the other user to have more information than just the encryption key. First, any pixels with a value of 256 are modified such that each individual pixel can be stored using only 8 bits.

The locations of the original 256 values are stored in a bitmap the compressed with arithmetic coding which I decided to replace with run-length encoding due to the sparsity of the matrix and the efficiency of the arithmetic encoding (see Figure 4 for benchmarks). With this algorithm replacement, there was a great boost to time efficiency when looking at encoding time as well as payload size decrease. using run-length encoding there was an average decrease of 3x for the length of the result of the encoding.

```
Original Length:  262144

Run Length Encoding:
Encoded Length:  4034
Time Elapsed (seconds):  0.07288908958435059

Arithmetic Coding:
Encoded Length:  11007
Time Elapsed (seconds):  24.075805187225342
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Original Length:  262144

Run Length Encoding:
Encoded Length:  3870
Time Elapsed (seconds):  0.05225539207458496

Arithmetic Coding:
Encoded Length:  10671
Time Elapsed (seconds):  23.733716011047363
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Original Length:  262144

Run Length Encoding:
Encoded Length:  4114
Time Elapsed (seconds):  0.05023527145385742

Arithmetic Coding:
Encoded Length:  11157
Time Elapsed (seconds):  23.180333852767944
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Figure 6: Encoding Benchmarks

The original least significant bits we replaced when storing the side information of the encrypted image are then added to our compressed bitmap then encoded into the image using difference expansion reversible data hiding[3]. This is done by computing the difference expansion using pairs of pixels with a reversible integer transform. Once the transform is computed the individual pairs are split into subsets of whether they are expandable or changeable. Expandable and changeable pixels are pixels in the image that can be identified through the transform even after the least significant bit is replaced. Using these subsets, pixels are selected and their bits are replaced with the original values being added to the payload along with some other side information for the algorithm.

Figure 7: Difference Expansion Results on Encrypted Images

Once the side information has been embedded into the encrypted images, they can be sent off to individual data hiders. Data hiders vacate space within the image reversibly such that our payload of secret data can be encrypted within. BEME is used to further expand the embedding capacity of our images. This is done by dividing all pixel errors into different classes and then it encodes the last bits of the pixel errors alongside their classes (see Figure 5). Once BEME is complete the embedding space is the pixels of the entire image block except for the first pixel. From the side information within the whole image, the data hider can find the position of the first available space and embed data within it. When looking to embed data for additional security AES is used with a data-hiding key to encrypt the data so it cannot be easily recovered.

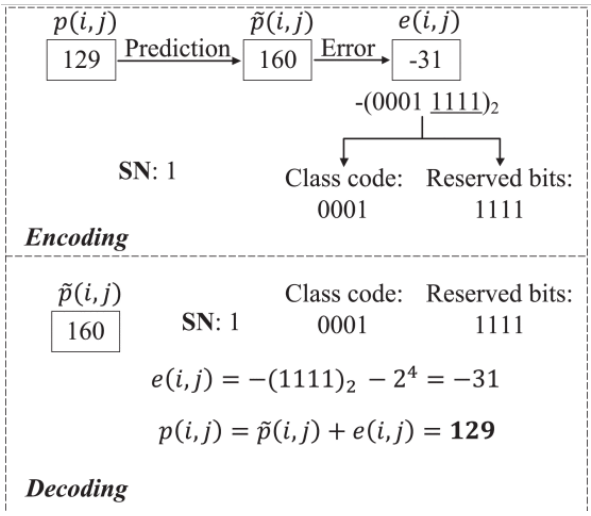

Figure 8: Example BEME Usage

BEME is a great approach to further expand the method to allow for lots of space for data hiding within the image and the inclusion of AES makes it secure and reliable across any implementation along with being fast and easily reproducible. Once BEME is complete the encryption and data hiding is complete and the image can be stored and transported safely without any worries of security.

When a receiver is looking to access the image it can be done quite simply. First, they must recover the embedded information by decoding the BEME process bits utilizing the recovered classification.

Once done they will have the encrypted information from which the side information can be recovered by gathering and replacing the information in the least significant bits of the expandable and changeable bits and replacing them with the encoded originals. With the side information recovered it is trivial to reproduce the threshold matrix with our encryption key. Since the original image can be recovered by only having **r** of the **n** encrypted images the threshold matrix X_r can be used for our inverse transformation. X_r is acquired by taking the i columns in our threshold matrix that each image corresponds to. Then by calculating the inverse of our $r \times r$ matrix, we can use that to decrypt our shares which can be reconstructed by images by using the values of each pixel at a specific point as a value in a vector. This decryption is done by multiplying our recovered encrypted shares vector by the inverse of our threshold matrix columns. once all the pixel values are recovered from the decrypted shares vector the original image can be reconstructed by remaking each individual block and then reversing the block scrambling by recovering the scramble order from the encryption key.

Being able to reconstruct the original image from only r-many of the encrypted images is one of the many upsides to this approach as it provides strong redundancy and data protection. Corruption and loss of a single image are not an issue and provide coverage from even further points of failure at the loss of only one so long as **n** and **r** are large enough.

4 Conclusion

RDHEI is an incredibly important field of study for computer scientists as we continue to move into the age of the cloud. Techniques and approaches like this must be advanced to keep our data safe and secure while also providing great utility to those who use it. The RDHEI method proposed in the paper is a very strong approach to the problem at hand. It has great data redundancy as well as payload space within the encrypted image. Using VRAE allows the method to leverage cloud computing while keeping the secret information safe. So long as parameters are selected properly the points of failure protect the user from any inability to recover the original image. It is still important to note the ambiguity in the implementation and provide more specifics when talking about PRNG when looking for general implementation as deriving reproducible results from the encryption key with inputs is important in RDH.

5 References

[1] “Matrix-Based Secret Sharing for Reversible Data Hiding in Encrypted Images — IEEE Journals Magazine — IEEE Xplore,” ieeexplore.ieee.org. https://ieeexplore.ieee.org/document/9933877.

[2] W. Puech, M. Chaumont, and O. Strauss, “A reversible data hiding method for encrypted images,” SPIE Digital Library, Mar. 18, 2008. https://www.spiedigitallibrary.org/conference-proceedings-of-spie/6819/1/A-reversible-data-hiding-method-for-encrypted-images/10.1117/12.766754.short?SSO=1

[3] “Reversible data embedding using a difference expansion — IEEE Journals Magazine — IEEE Xplore,” ieeexplore.ieee.org. https://ieeexplore.ieee.org/document/1227616/ (accessed Dec. 09, 2023).

[4] “Two-Dimensional Sine Chaotification System With Hardware Implementation — IEEE Journals Magazine — IEEE Xplore,” ieeexplore.ieee.org. https://ieeexplore.ieee.org/document/8738838 (accessed Dec. 09, 2023).

[5] ”Reversible Data Hiding With Hierarchical Block Variable Length Coding for Cloud Security — IEEE Journals Magazine — IEEE Xplore,” ieeexplore.ieee.org. https://ieeexplore.ieee.org/document/9940594 (accessed Dec. 09, 2023).