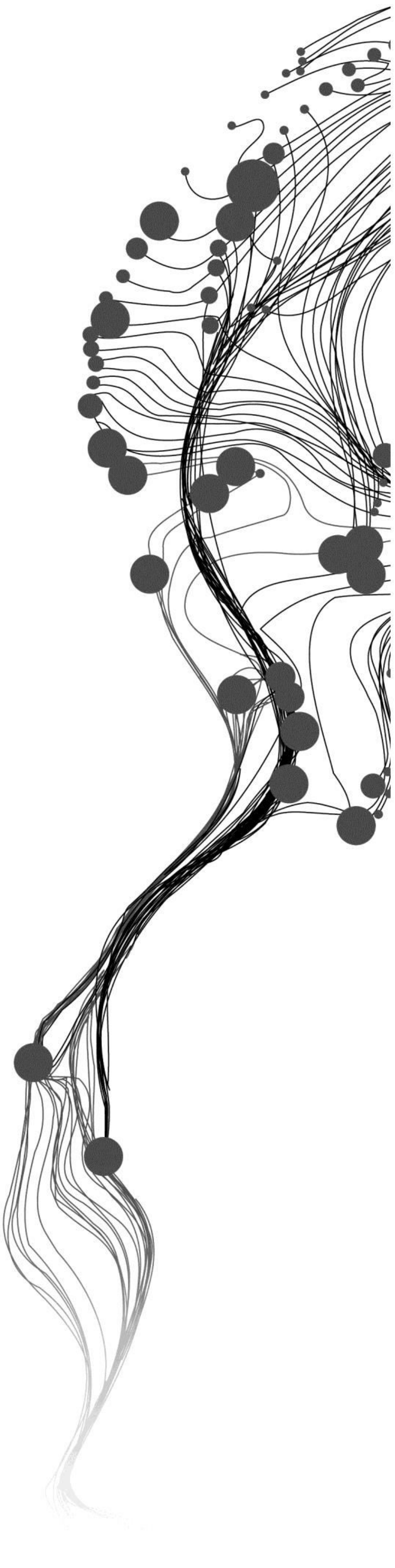
COMPRESSION AND ENCRYPTION FOR SATELLITE IMAGES: A COMPARISON BETWEEN SQUEEZE CIPHER AND SPATIAL SIMULATIONS

GETACHEW MEHABIE MULUALEM
FEBRUARY, 2015

SUPERVISORS:
Prof. Dr. Ir. Alfred Stein
Dr. Ir. Rolf A.de By



COMPRESSION AND ENCRYPTION FOR SATELLITE IMAGES: A COMPARISON BETWEEN SQUEEZE CIPHER AND SPATIAL SIMULATIONS

GETACHEW MEHABIE MULAUELM
ENSCHEDE, THE NETHERLANDS, FEBRUARY, 2015

Thesis submitted to the Faculty of Geo-Information Science and Earth Observation of the University of Twente in partial fulfilment of the requirements for the degree of Master of Science in Geo-information Science and Earth Observation.
Specialization: Geoinformatics

ABSTRACT

For solving network bandwidth, and security problems, image compression, and encryption technologies need to be combined. This paper presents two techniques for real-time compression and encryption aimed at satellite images that are performed simultaneously.

The paper analyzes a modification of the LZW dictionary-based compression algorithm that is adapted to provide security, this algorithm known as the squeeze cipher algorithm is implemented for satellite images. The proposed method can protect and compress satellite images while keeping the contents of the image intact.

Spatial simulations are investigated as an alternative approach to the existing image compression algorithms. Spatial simulation models were established to predict the DN values of satellite images from neighbor pixels. To compare the simulated and actual values several statistical analysis tests such as variogram analysis, RMSE and MSE tests are performed between the restored and the original image. Advanced encryption standard (AES) is then added as a post-processing on top of the proposed compression algorithm to provide security for communications. The performances of image compression and encryption methods employed in this paper are compared on the basis of the compression ratio, elapsed time and differences between the input image and restored image.

ACKNOWLEDGEMENTS

First and above all, I praise God, the Almighty for providing me this opportunity and granting me the capability to proceed successfully.

I offer my sincerest gratitude to my first supervisor Prof. Dr. Ir. Alfred Stein, who has supported me throughout my thesis. Without his encouragement and patient instruction, this thesis would not have been completed.

I would like to express appreciation to my second supervisor Dr. Ir. Rolf A.de By, for constructive comments and suggestions that have been invaluable to this work to take its present shape.

I am also grateful to Dr. Andreas Peter for the in-depth discussions, suggestions, critical and useful comments. I remain grateful to him for he helped me find the path to the completion of this work.

I would especially like to thank Bas Retsios, Claudio Piccinini and Getachew Feleke for providing me technical support during the thesis work.

I thank my parents; they have always loved me and supported my every choice. I also like to thank all my friends and ITC HABESHA communities for the great joys and happiness that make our stay in Netherlands enjoyable and full of memories.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Acronyms

AES	Advanced Encryption Standard
LZW	Lempel–Ziv–Welch
GIF	Graphic Interchange Format
TIFF	Tagged Image File Format
PDF	Portable Document Format
JPEG	Joint Photographic Expert Group
SGS	Sequential Gaussian Simulation
MPS	Multiple Point Statistics
DES	Data Encryption Standard
IDEA	International Data Encryption Algorithm
RSA	Rivest – Shamir - Adelman
PNG	Portable Network Graphic
Cdf	Cumulative Distribution Function
GGS	Gaussian Geostatistical Simulation
NIST	National Institute of Standards and Technology

1. INTRODUCTION

1.1. Motivation and problem statement

Images and other multimedia files are frequently transmitted via computer networks (Forouzan, 2010). The transmission of information across untrusted networks endangers the security of the information. For example, due to continuous attempts of hackers, images end up in the hands of illegal third parties during communication that might profit or amend them without the awareness of the appropriate receiver (Huang et al., 2012). Such alterations cause a major disaster since reliable transmission of images, is needed in many applications such as military operations, business transactions and medical systems (Lee, Chen, & Lee, 2003). We, for this reason, need to protect information from unauthorized access, guarantee their content from the change and prevent them from network attacks during transmission (Stallings, 2006).

For this reason, a secure way, to communicate in the presence of adversaries, is important. This demand, to prevent the contents of messages from intruders, gave birth to cryptography, the science and art of sending messages in secret. This technique will change the transmitted information into an unreadable form by encryption; the process of converting the original data into a scrambled unreadable format. So, encryption plays utmost step to assure confidential communication (Khan & Shah, 2014). Subsequently, no eavesdropper, who has access to the transmission channel, gets back the plain message.

On one hand, the fast growing demand of transmitting images via public networks has raised up interest in image compression (Yuen & Wong, 2011). The magnitude of data sent has become bigger, and thus data compression is becoming a vital tool (Aldossari, Alfalou, & Brosseau, 2014). In particular due to the increase in resolution and size of satellite images they have become an important unaffordable source of information. Thus, frequent transmission of these images in different corners of our universe is inevitable, this uplifts a high demand to compress such images. By compressing images, we aim to ease worthless and redundant data in order to be able to stock or communicate them in an efficient form. Accordingly, the size of the images and also the time needed to send images over the network will become shorter. Study on compression of images has been carried out for long (Yuen & Wong, 2011) with the goal of reducing the image size for sufficient storage and fast communication (Zhu et al., 2013).

Independent compression and encryptions of images, however, are slow to meet the demand for many multimedia applications (Moo, 1999). Consequently many scholars including (Zhu et al., 2013), (Zhou, Bao, & Chen, 2014) and (Muhaya & T., 2011) propose algorithms that does not require extra processing to perform simultaneously encryption and compression. One idea is to push in the encryption operation into the compression process and perform them in a single step (Xiang, Qu, & Xiao, 2014). Especially by means of dictionary-based encoding, compression and encryption are addressed simultaneously (Kelley & Tamassia, 2014) as a single simplified process. Otherwise, we treat them as two independent procedures, one after the other. In such cases, encryption comes after compression since in theory encrypted data is incompressible (Xiang et al., 2014).

In this paper, we consider combining compressions and encryption by taking both approaches; joint compression and encryption and compression followed by encryption for satellite images.

1.2. Research identification

The research paper aims to use the squeeze cipher and spatial statistics approach to perform compression and encryption of satellite images. The squeeze cipher algorithm, as designed by James & Roberto in 2014, is a text encryption and compression cipher based on the standard LZW algorithm. On the other hand, spatial statistics provides procedures to express the distribution of data in space, study the spatial patterns of the data, designate spatial relationships and create a model from sampled data. Note that, spatial simulations have a potential to reflect a spatial dataset as if it could have been by generating it using spatial correlation. We developed scientific interest to what the spatial simulations are doing and make comparisons between the squeeze and spatial simulations.

1.2.1. Research objectives

The proposed MSc research primary purpose is to evaluate and compare the squeeze cipher encryption and compression technique (S1) with the geostatistical based encryption technique (S2).
The paper includes the following specific objectives:

- ✓ To adapt S1 to encrypt and compress satellite images.
- ✓ To investigate data loss caused by image compression in S2.
- ✓ To apply both S1 and S2 to satellite images.
- ✓ To investigate the efficiency of both S1 and S2.
- ✓ To study the effects of different input restrictions.
- ✓ To analyze image compression performed together with encryption systems.

1.2.2. Research questions

In this study, we will address the following issues:

- ✓ Should S1 be adapted to encrypt and compress satellite images?
- ✓ How can we minimize data loss during image compression in S2?
- ✓ What degree of compression gives an acceptable result in S2?
- ✓ How to examine the efficiency of both S1 and S2?
- ✓ How to use additional information on the input image to improve the compression accuracy?
- ✓ Should image compression and encryption be done in one go? Alternatively, independently?

1.2.3. Innovation aimed at

Many researchers have tried various approaches to conceive procedures that compromise combinations of encryption and compression schemes for images (Zhu, Zhao, & Zhang, 2013) and(N. Zhou, Zhang, Wu, Pei, & Yang, 2014). Theoretically it is believed that compression and encryption are two independent processes (Xiang et al., 2014). However, because encrypted files are incompressible due to their randomness and most compression algorithms fail to isolate such redundancies, compression has to be performed before encryption. Figure 1.1 depicts the standard way to combine image encryption and compression.

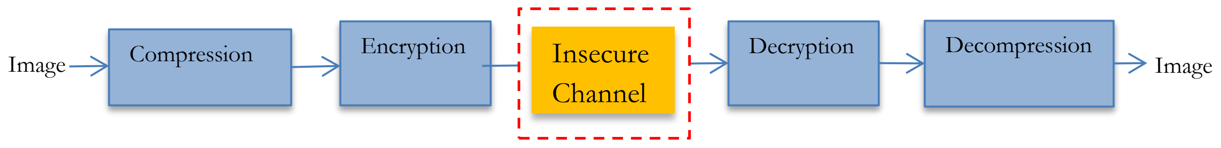


Fig 1.1. Image compression and encryption

According to(Shine, 2001) spatial statistics, gives an approach to compression of enormous images. By adding an encryption on top of a standard compression method, we can meet the needs of secure compression. In general, the spectral values of neighboring pixels of a satellite image are highly correlated. Due to this it is not required to represent all neighboring pixels independently in an image. Pixels can be reconstructed from neighboring ones by applying spatial redundancy methods. Spatial statistical methods are employed to generate data at unsampled locations from data collected at a finite number of places. Since random sampling gives all pixels equal chance to be chosen, it is used as a way to get a representative of the whole image. The outcome of the sampling is an obscure version of the original image that is small

in size and has less detail. We consider the sampling as a compression technique. At the receiver end, we reverse the compression by applying conditional simulations.
(Lantuéjoul, 2002)suggest that conditional simulations can be used as an alternative to the existing image compression algorithm. Conditional simulation generates realizations that approximate data and honor the sampled observations. Such procedure is considered, and the results will be evaluated in this study.
In addition, the research adapts the squeeze cipher algorithm to compress and encrypt satellite images. Originally squeeze cipher is designed for encrypting and compressing texts in one go. The cipher bases on the LZW algorithm, named after the developers Lempel–Ziv–Welch.

1.3. Method adopted

LZW is a standard dictionary-based data compression algorithm developed by Welch in 1984. Dictionary based compression algorithms create a dictionary when the data is processed and look for repeated strings. Based on the strings recognized in the dictionary a much shorter string will be used as a replacement. This process outputs a compression of the overall information. The method is found in several image file formats, such as the (GIF), (TIFF) and PDF (Fulton, 1997). By changing the management of the dictionary of the LZW, a random dictionary is formulated (Kelley & Tamassia, 2014). The new dictionary incorporates encryption, and it is named as squeeze cipher.
In addition, we explore in depth the theories of spatial statistics to integrate them into a standard compression method. In each step, by selecting pixels from the whole image randomly we can form compressed image. The features, size and quality of the compressed image heavily rely on the eminence of the sampling technique. The sampling is done randomly in an unbiased way to produce an unclear version of the original image. The outcome of the sampling is a set of randomly selected pixels and their corresponding coordinates. We next form a variogram fit and simulation model. The models contain relevant information and assumptions to recover the original image (Lantuéjoul, 2002). After the compression and decompression methods are fixed we then, apply encryption on top of this process. Upon communications, to the receiver we send the selected pixels and the model parameters encrypted with AES counter mode. The legitimate receiver provided with the right key will then recover the randomly selected pixel together with their locations and the model parameters. Decompression is achieved by running spatial simulation models. In such simulations, we use a model already developed together with computer programs to determine the simulation iteratively over time (David & Perry, 2013). With this encryption scheme, an attacker in the middle of the communication will not defeat our method and use our resources.
The two proposed approaches take as input satellite image and produce a scrambled unclear image. From such vague image, the receiver will recover the original input image. The methods will be carried out by following the workflow depicted in Figure 1.4.

1.4. Overview of the research

The thesis is organized into six chapters. Chapter one describes the motivation behind the combination of compression and encryption, the research objectives, the method adopted and the utility of the study. Following the introductory is chapter 2, which acquaints us to the basic concepts and review the literature of the existing compression and encryption techniques. Chapter 3 deals with the resources that are used and present the overall compression and encryption algorithm including a thorough description of the decompression methods. The squeeze cipher and spatial statistics based encryption are discussed. Chapter 4 discuss the implementations, experimental results, and the aftermath of the results supported by figures. Chapter 5 presents a discussion of the results and future research directions. Finally, Chapter Six presents the conclusions we made and enclosed the recommendations. At the last we include the appendix. Appendix contains the details of the algorithms and additional information.

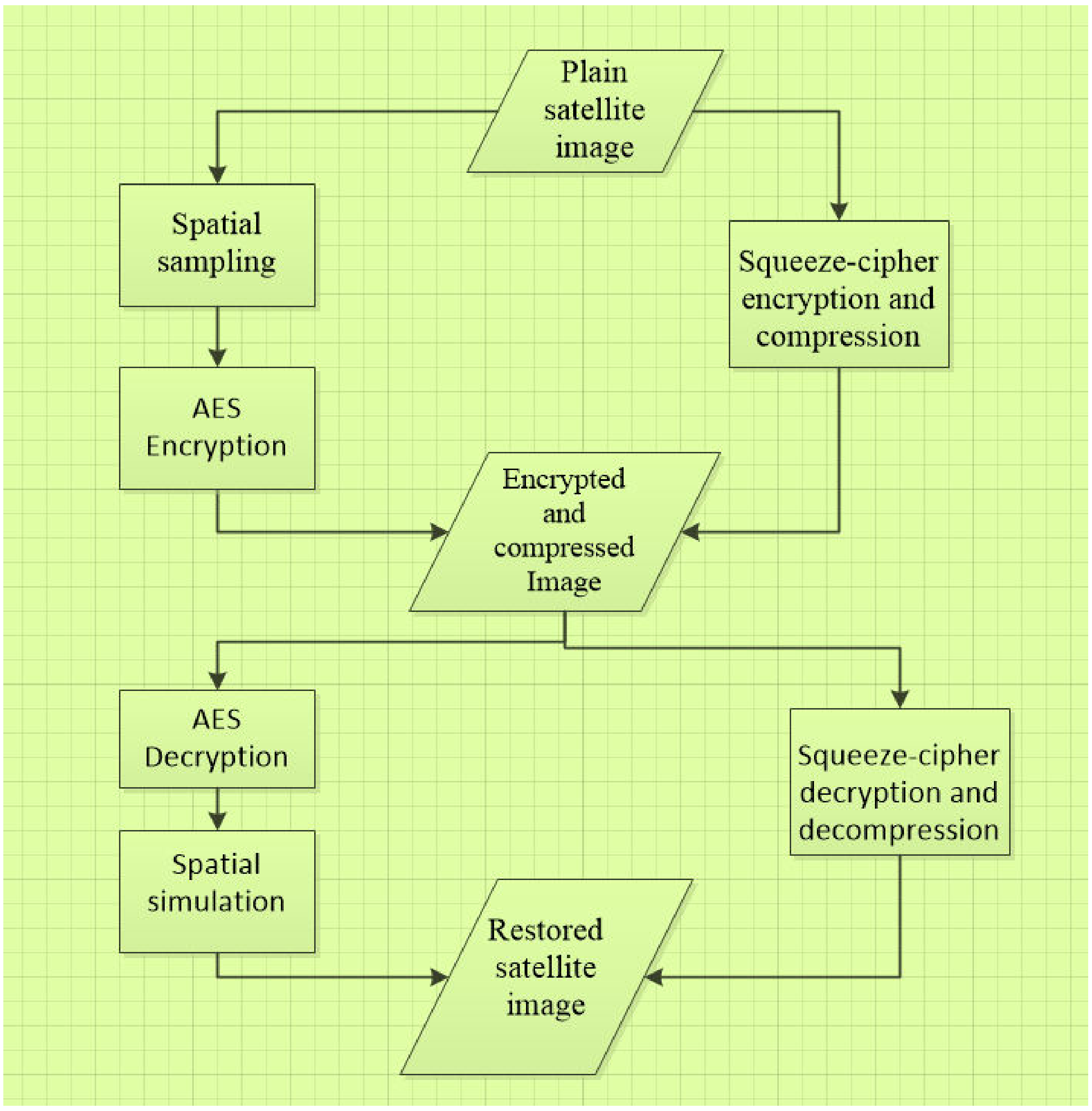

Plain satellite image
Spatial sampling
Squeeze-cipher encryption and compression
AES Encryption
Encrypted and compressed Image
AES Decryption
Squeeze-cipher decryption and decompression
Spatial simulation
Restored satellite image

Fig 1.2: Flowchart showing the steps used by both methods

2. LITERATURE REVIEW

The aim of this chapter is to provide background information on the proposed strategies of image compression and encryption. The first two sections are dedicated to the broader concepts of compression and encryption. The last sections establish the significance of the general field of study and identify areas where a new approach could be used.

2.1. Image compression

The fast growth of remote sensing technology accelerate the collection of vast amounts of satellite data (Ghosh & Singh, 2009). This data has become the source of information; nevertheless its enormous size limits their emerging application in different areas of our life. The increase in spectral and spatial resolutions of satellite images raises the cost of transmission, processing and storage of these image data. Consequently, image compression practices become vital despite the advance in storage capacity.

Compression targets to shrink the size of the file by removing redundant information. It converts data into a new format that takes small bandwidth and small space. Most compression algorithms are designed to use effectively storage capacity and reduce transmission time with the aim of saving resources (Mukesh & Smiley, 2012).

In uncompressed data, the adjacent values contain repetitive information. For instance, in satellite images the spatial, spectral and temporal correlation between adjacent pixels is high, which makes sense to compress them (Munish & Anshul, 2014). Not only images but other types of digital files require compression too namely text files, source codes, audio data and videos. Nowadays, there are many compression techniques widely accessible that are developed for different data types.

A file can be reduced with a loss in quality of the data when it is uncompressed, which is known as lossy compression or without a loss in quality that is referred as lossless compression. Lossy compression algorithms are "tunable" i.e. we can turn the compression ratio [1]up to the level we want, but at a loss of quality (Sheldon, 2001). Lossy image compression can accomplish a high compression rate but then an image reconstructed with this method is corrupted relatively to the original (Sha, 2008). The reconstructed image from this method is often indistinguishable for a human eye though they are bitwise different. The JPEG and fractal compressions are the most widely used lossy compression techniques.

In contrast, in lossless compression all the information is recovered after the file is uncompressed which upturn its importance in many fields such as medical image analysis, security, defence and remote sensing (Hu & Chang, 2000). However, lossless compression can only attain a reasonable extent of compression (Sha, 2008). A lossless data compression is implemented using either statistical or dictionary-based approach (Nandi & Mandal, 2013). Statistically based compression reads in and encodes a single character by considering the probability of that character's presence. On the other hand, dictionary-based compression uses a unique code (entries) to replace string of characters (Nandi & Mandal, 2013). Some traditional lossless compression algorithms are PNG, TIFF, JPEG 2000 and GIF.

2.1.1. Dictionary-based methods

The dictionary-based compression schemes are among the most common compression algorithms used in practice (Langiu, 2013). This process is widely held because it offers equally noble compression ratio and firm decompression technique. The notion is to exploit repeating sequences by using a dictionary (Seong & Mishra, 2006). A dictionary-based compression stores strings in a dictionary and assign distinct integers known as indexes. The algorithm constructs a dictionary of strings and replaces substrings of an input text

[1] Compression ratio is defined as the size of the compressed data divided by the size of original data. It is regarded as measure of efficiency of compression.

with indexes in the dictionary. New strings generated by adding character to the end of an existing string will be added to the dictionary until it is full.

Dictionary-based compression uses two kind of dictionary namely static and adaptive. The static dictionary is built up before compression occurs, and it does not change during the processing of data. The receiver of the message cannot reconstruct the same dictionary by processing the decompressed file. So, the dictionary has to be transmitted along with the file that results in a certain add to the compressed file (Nandi & Mandal, 2013). Adaptive dictionary scheme helps avoid this problem by reconstructing the dictionary while the data is compressed.

The introductory (Ziv & Lempel, 1978) dictionary-based algorithm is the beginning of nearly all the well-known dictionary compression algorithms such as gZip, Zip, RAR, and LZMA (Langiu, 2013). Different variant of this algorithm has existed that improve the speed of compression, compression ratio and the formation of the dictionary. The most popular one considered as the standard dictionary-based data compression technique is LZW.

2.1.2. Geostatistics-based methods

To reduce repetition of an image data and able to transmit them in small size we can use sampling as a way of compression. The primary intent of sampling is to gather samples in 1-, 2-, 3-dimensionional spaces and estimate the parameters of the whole population(Wang, Stein, Gao, & Ge, 2012). It selects a subsection from a population to assess features of the entire community. By repeatedly sampling an area spatial sampling, ensures analogous result will be obtained at a lower cost. Thus, a well-established approach of sampling is crucial to compression procedures. A computer-supported random sampling is used to generate unbiased samples. Such random samples have no pattern and serve as a representative of the whole image. Further, to decompress the image we build simulation models and estimate their parameters based on the natures of the samples.

2.2. Image encryption

In a digital age, due to the fast increase in communication technology there is a huge plea for information security. Image conveys indispensable data and is extensively used in numerous fields of society and equivalent to its practice, it is essential to keep them safe (ur Rehman, Liao, Kulsoom, & Abbas, 2014). Encryption is one way to scramble image data so that unapproved person cannot recognize its content (Panduranga & NaveenKumar, 2013). Several text encoding schemes such as DES, IDEA, AES, and RSA were used for images (ur Rehman et al., 2014). However, in line with the inherent characters of the image such as an immense size and high correlation (Socek et al., 2007), encryption of image demands its own approach (Jin, 2012).

For many purposes, AES guarantees the tightest security currently available. Nevertheless, it has shortcomings such as high computation costs and hardware requirements (Salim Muhsin Wadi & Zainal, 2014). A minor modifications of AES existed for enciphering images, a typical one by (Salim M. Wadi & Zainal, 2013) replace the S-Box with the new S-Box to decrease hardware requirements. To enhance the key space and strength the security of satellite images chaotic and AES-based satellite image cryptosystem have been used (Usama, Khan, Alghathbar, & Lee, 2010) and (Muhaya & T., 2011) such schemes give high level of security.

A contemporary trend developed due to the computational requirements is selective encryption. This method moderates the execution time since it encrypts only portion of an image (Panduranga & NaveenKumar, 2013). This paper discusses AES encryption methodology that is applied to the randomly selected pixels of an image.

2.3. Image compression and encryption methods

In recent years study has been done to combine compression and encryption for secure and fast image communication. Several procedures have been suggested to combine compression and encryption so as to reduce the whole processing time. However, they are either insecure or computationally rigorous (Cheng, 2000).

In 2005, Xie & Kuo formulated an algorithm to include security on the dictionary-based compression. In particular, they presented data compression method that incorporates encryption algorithm into the LZ78 compression method. The main idea is to construct multiple dictionaries with different entries order and randomly select one of the dictionaries by using a pseudo-random sequence. Unless someone gets the correct pseudo-random numbers used to choose the dictionaries, the decompression will lead to an incorrect result. So the key components of this approach are the construction of multiple dictionaries and the formation of random sequence that is used to select the dictionaries. In 2008, (J. Zhou, Au, Fan, & Wong, 2008) came up with an algorithm that inserts and permutes the dictionary entries of the LZW. The basic idea was to add the dictionary entries randomly and randomly permute them and also use a key scheduler. The key scheduler will generate key streams. Then, the final output of the LZW algorithm is XORed with the key stream generated by the key scheduler. The security analysis performed show that the algorithm can provide a high level of safety. Further, Yuen & Wong in 2011 also proposed a chaos–based joint image compression and encryption algorithm that uses the discrete cosine transform (DCT) and hash function SHA-1. The DCT, which is adopted in the JPEG compression standard, transforms the spatial domain into the frequency domain i.e. It destroys the correlation between image data. Also, SHA-1 is designed in such a way that it is fast and sensitive to input the image; a small change in the plain image will affect the whole process. The key sensitivity, plain-image sensitive tests performed reveal that the scheme has a high-level security and large key space. In addition, (Zhu et al.., 2013) introduced a technique to compress and encrypt images simultaneously using a hyperchaos and the Chinese remainder theorem. The 2D hyperchaos shuffles the plain image and applies the Chinese remainder theorem to diffuse and compress the shuffled image. Keyspace, histogram, correlation, key sensitivity and information entropy analysis carried out in the paper show that the method is valid.

3. MATERIAL AND METHODS

3.1. Study area

Remotely sensed images constitute a record of distinct spatial properties of the Earth's surface (Zhang et al., 2009). Images are treated as field data depicted by varied digital numbers (DN). In this study, the data being analyzed is a small area subset from an ETM+ panchromatic image acquired on 2009-11-03 of Bahir Dar; a city in the north-western of Ethiopia and the capital of Amhara Region.

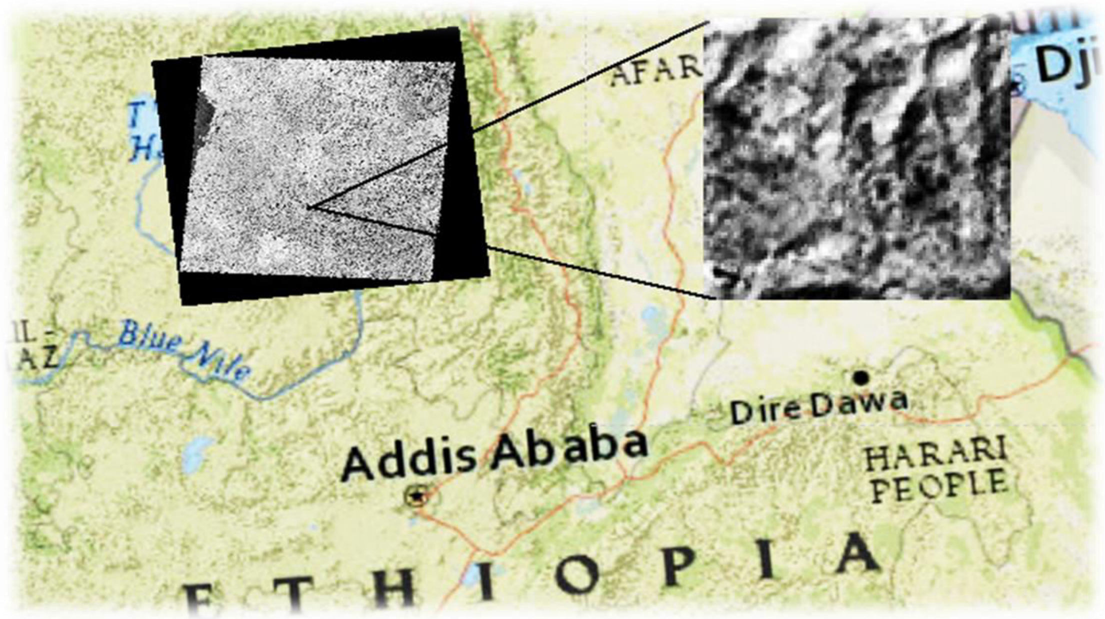


Fig 1.3 Study Area

A 101×101-pixel subset was selected from the image and highlighted in the image. The subset comprised a total of 10,201 pixels. The image is 15 m spatial resolution. It covers 22,725 m^2 of the region. The subset image, as shown in Fig 3.1, is stored as a tiff file and has a size of 11,686 bytes. The data sets were exported as ASCII text files. The data file from this image contains the coordinates of the pixels and the pixel values. The data file was then imported into R statistical software and converted into a geostatistical data set.

3.2. LZW compression

LZW is a widely used text compression algorithm (LAKHANI, 2006). It is a dictionary-based compression algorithm that constitutes a lossless algorithm since the original data is entirely reconstructed from the compressed data. Let us pause here to define terms that are used throughout this document:

Character: is a single object used to represent text, numbers or symbols. In computer representation of text, one character is commonly equal to one byte (consisting of eight bits).

String: a sequence of characters, length zero or higher. The 'empty string' is a character sequence of length zero, and we denote it as Λ.

Concatenation: Placing two or more separate strings side-by-side following each other so that they become a single string, this operation is denoted by $||$.

Prefix: a string that precedes a character of interest.

Index: an integer number, used as an address in a dictionary.

Entry: is a string, stored in a dictionary at some index.

Dictionary: a table that stores (index, entry) pairs, and because index values are unique within the dictionary, it realizes a functional relationship that maps index values to entries.

XOR: is a boolean operator that operates as the exclusive disjunction operator and its denoted by $\oplus$.

In the process of LZW compression, the first step is to initialize the dictionary. For that, we need to choose with how many bits a character is represented. Let us assume that a character representation size is 12 bits, and then our initial dictionary will hold $2^{12} = 4096$ entries in the dictionary, as for each string of length one the initial dictionary has an entry holding that one-character string (Blackstock, 2001). In the case of a panchromatic image, with an 8-bit quantization, the first 256 entries of the dictionary are assigned to the gray levels encoded as \00, \01, \02, ..., \FF. [2] The remaining entries in the table are left blank. The byte strings formed, which are not in the dictionary, are later assigned to additional entries in the dictionary. For instance, if the first two pixels of an image are black then the sequence '\00\00' is inserted as an entry of the dictionary index 256, which is the first available free slot in the dictionary.

The decompression starts by receiving the continuous stream of encoded output values (indices) that refer to the dictionary table. Then, the LZW decompression is merely copying bytestrings from the dictionary according to these indices (Xie & Kuo, 2005). To start, we look at the first index of the encoded output, the bytestring corresponding to this index is part of the initial dictionary thus we output it. Make this index 'prefix'. Then look at the next index, this new index may not have an entry associated with it in the dictionary. Assume for now it is; output the bytestring corresponding to it. Then find the first character in the bytestring converted, call this c. Add this to the bytestring created by 'prefix' to form a new string 'bytestring || c', add this string 'bytestring || c' to the dictionary entry and set the 'previous' to the new index. We repeat this process until we are set. In the exceptional case where the bytestring associated to an index is not in the dictionary (or not defined yet), it outputs the first character of the bytestring related to the 'prefix' index and concatenates it with the 'bytestring' itself and add this new string to the dictionary. In the end, an identical dictionary to that of compression is constructed.

The LZW compression algorithm (Kelley & Tamassia, 2014)**:**

```
Input: Character stream $I$

Output: Stream of table indices

Initialize dictionary $T$ to contain all single-character strings.

$prefix \leftarrow \Lambda$ ------------------- ⊳ The current prefix of the input is empty

$i \leftarrow$ number of single-character strings---------------- ⊳ The index of the first free entry in $T$

while there is more input in $I$ do

read next character $c$

if $prefix \,||\, c$ is an entry in $T$ then
```

[2] The notation \XY in which X and Y is any single character in the range [0..9A..F] is the hexadecimal (base-16) representation of a byte.

```
$prefix \leftarrow prefix \,||\, c$----------------------▻ Extend prefix with $c$ and continue processing input
else
Output index of $prefix$ in $T$------------------------------------ ▻ $prefix \,||\, c$ not in$T$: add it to $T$
$T[i] \leftarrow prefix \,||\, c$
$i \leftarrow i + 1$
$prefix \leftarrow c$------------------------------------------- ▻ Continuing processing the input starting at c
if $prefix \neq \Lambda$ then-------------------------------------- ▻ Output index of the remaining input
Output index of $prefix$ in $T$
```

The decompression starts by receiving the continuous stream of encoded output values (indices) that refer to the dictionary table. Then, the LZW decompression is just copying bytestring from the dictionary according to these indices (Xie & Kuo, 2005). To start, we look at the first index of the encoded output, the bytestring corresponding to this index is part of the initial dictionary thus we output it. Make this index 'prefix'. Then look at the next index, this new index may not have an entry associated with it in the dictionary. Assume for now it is; output the bytestring corresponding to it. Then find the first character in the bytestring converted, call this c. Add this to the bytestring created by 'prefix' to form a new string 'bytestring || c', add this string 'bytestring ||c' to the dictionary entry and set the 'previous' to the new index. We repeat this process until we are set. In the exceptional case where the bytestring associated to an index is not in the dictionary (or not defined yet), it outputs the first character of the bytestring related to the 'prefix' index and concatenates it with the 'bytestring' itself and add this new string to the dictionary. In the end, an identical dictionary to that of compression is constructed.

The LZW decompression algorithm (Kelley & Tamassia, 2014)

```
Input: Sequence of indices $I$
Output: Stream of characters or the error symbol $\perp$
 Initialize dictionary $T$ to contain all single-character strings.
$c \leftarrow \Lambda$ --------------------------------------▻ first character of next output string, initialize to empty
$prefix \leftarrow \Lambda$---------------------------------------▻the previously output string, initialize to empty
i ← number of single-character strings while
while there is more input in I do
read next index $k$
if $T[k]$ is defined then ------------------------------------------------------------ ▻ $k$ is a valid index
$c \leftarrow head(T[k])$---------------------------------------------------- ▻ Take the first character of $T[k]$.
$T[i] \leftarrow prefix \,||\, c$ -------------------------------------------- ▻ $prev \,||\, c$ was the value inserted after encoding $prefix$.
```

```
i←i+1
else
if $T[k]$ is not defined and $k = i$ then--- ⊳ Special case: original input was $prefix \,||\, prev \,||\, c$
$c \leftarrow head(prev)$---------------------------------- ⊳ the first character of $prefix$ is c.
$T[k] \leftarrow prev\ o\ c$ --------------------------------- ⊳ We're decoding $prefix \,||\, c$.
$i \leftarrow i + 1$
else
Output ⊥ and exit--------------------------------------------------⊳ Decoding failed: invalid index
Output $T[k]$ ---------------------------------------------- ⊳ Finally: output $T[k]$ and update $prefix$
$prev \leftarrow T[k]$
```

For better understanding of the LZW algorithm, let us consider a 4×4, monochrome 8-bit image.

\3A---\3A---\15---\15

\3A---\3A---\15---\15

\3A---\3A---\15---\15

\3A---\3A---\15---\15

The starting point of the dictionary is the table shown below and consists of just the single characters.

Dictionary T

index	**entry**
0	\00
1	\01
…	…
255	\FF
256	-
…	…
511	-

Table 3.1 Initial LZW Dictionary

In the beginning, the dictionary locations above 255 are not assigned. After receiving the first character \3A, it matches with the entry at index value 58. But the pattern \3A\3A (prefix || ch) does not have a matching bytestring entry in the dictionary, in which case the index for the string, i.e. the longest substring that is in the dictionary (\3A) is sent out to the output, thus outputting the index value 58 and we add bytestring \3A\3A to the dictionary as a new entry. The whole encoding table is depicted below.

Current prefix	Current string	Seen this before?	Encoded Output Value	Dictionary Index	Dictionary Entry
Empty	\3A	Yes	Nothing	None	None
\3A	\3A	No	58	256	\3A\3A
\3A	\15	No	58	257	\3A\15
\15	\15	No	21	258	\15\15
\15	\3A	No	21	259	\15\3A
\3A	\3A	Yes	Nothing	None	None
\3A\3A	\15	No	256	260	\3A\3A\15
\15	\15	Yes	Nothing	None	None
\15\15	\3A	No	258	261	\15\15\3A
\3A	\3A	Yes	Nothing	None	None
\3A\3A	\15	Yes	Nothing	None	None
\3A \3A \15	\15	No	260	262	\3A \3A \15\15
\15\3A	\3A	No	259	263	\15\3A\3A
\3A	\15	Yes	Nothing	None	None
\3A\15	\15	No	257	264	\3A\15\15
\15	\15	No	21		

Table 2: LZW compression table

The outcome of the decompression for the image is presented in the table below.

Current prefix	Encoded Input	Encoded Output	Dictionary Index	Dictionary Entry
Empty	58	\3A	Nothing	Nothing
58	58	\3A	256	\3A\3A
58	21	\15	257	\3A\15
21	21	\15	258	\15\15

21	256	\3A\3A	259	\15\3A
256	258	\15\15	260	\3A\3A\15
258	260	\3A\3A\15	261	\15\15\3A
260	259	\15\3A	262	\3A\3A\15\15
259	257	\3A\15	263	\15\3A\3A
257	21	\15	264	\3A\15\15

Table 3: LZW decompression table

There are repeating strings, and, as a result, the LZW manages to reduce the original 128-bit image to 90 bits. If the file contains, repetitive data LZW compression works best. If the file does not contain repetitive information, the compressed file can grow bigger.

3.3. Basic cryptography

Cryptography is concerned with the transmission of information in the presence of a foe, who is working firmly to retrieve the contents of data transmitted. Though ancient people were practicing it starting from 500 BC, it emerged as part of science with the development of computers in the past 50-60 years (Goldreich, 2007). It has gone from art to a science to secure systems for ordinary people (Katz & Lindell, 2007). It encompasses encryption, decryption, authentication, integrity to name a few.

The fundamental issue in cryptography is how to make secure communication over insecure network. An answer to this problem is to encrypt the information to be communicated i.e. converting data into a cipher, a form that can't be easily understood. An encryption converts plaintext to ciphertext and decryption does the reverse. For the basic cryptography terminology we refer the reader to (Goldreich, 2007). Suppose Party A want to send the plaintext p over insecure channel to Party B**,** the encryption procedure can be done as $c = E(K_e, p)$, where K_e is the encryption key and E is the encryption function. A key is simply a piece of information that is used to specify the transformation of plaintext into ciphertext and vice versa. Similarly, the decryption is $p = D(K_d, c)$ where K_d is the decryption key and D is the decryption function. When, $K_e = K_d$, it is called private-key encryption or symmetric key encryption. In such schemes, the key must be transmitted from the sender to the receiver via a secured secret channel or prior communication between both parties to agree up on a key is required. When $K_e \neq K_d$, and the encryption is called a public-key encryption or asymmetric encryption. I this scheme its computationally infeasible to get from K_e to K_d. In public key encryption the key K_e is published, and the decryption key K_d is kept private, for which no additional secret channel as well as no prior communication is required.

Note that the requirement of an encryption scheme is that for every key K and every plaintext message p it holds that

$$D_K(E_K(p)) = p.$$

An encryption system must have the property that decrypting a ciphertext with the appropriate key yields the original message that was encoded (Katz & Lindell, 2007). Apparently if the adversary knows the decryption function and the key k, the adversary is in a position to decrypt all communications. Thus, in designing a secure cryptosystem we assume the attacker knows the details of our cryptographic algorithm, and only the key is kept secret and hence the security relies on the knowledge of the keys. This is commonly referred to as Kerckhoff's principle. The encryption and decryption process is shown in Fig 1.4.

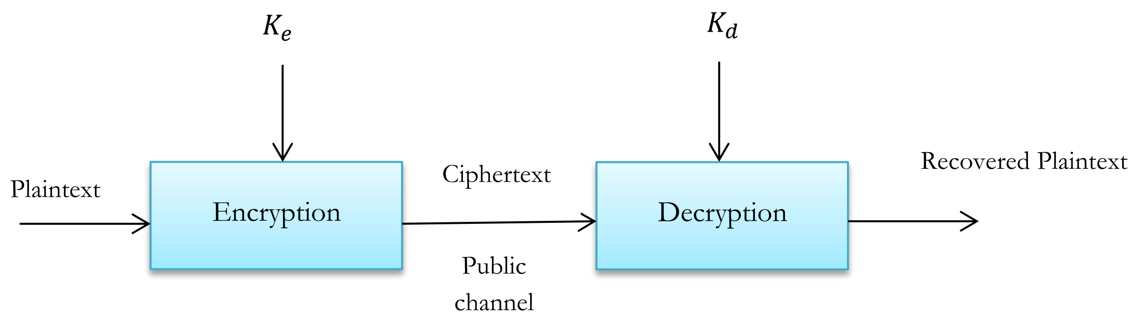


Fig 1.4: Encryption and decryption procedures

The key and the cryptographic algorithm are the two primary components of cryptography. The algorithm is a mathematical function used for applying protection to our data, and the key is a parameter used by the function. The key determines the operation in such a way a person with the appropriate key can reproduce the operations. The strength of the keys is vital since the reliability of the entire cryptographic practices depend upon them. It is important to utilize a longer key length to resist brute force attack. In computer cryptography, we use integers as keys. It then requires the generation of keys using pseudo-random number generators (PRNG). PRNG is a function that is initialized with a random value called the seed and output a sequence of numbers that "appears random": if an outsider does not know the value of the seed, then he cannot differentiate the output of PRNG from that of a true random sequence(van Tilborg & Jajodia, 2011). The process is deterministic as it always produces the same sequence when initialized with the same seed. In cryptography, we demand a highly secured random number that resist attacks in which nobody using advanced cryptanalysis methods can predict the output. Some well-known pseudo-random generator examples are the Linear Feedback Shift Registers and the Blum-Blum Shub generators. The application of PRNG in cryptography include the generation of cryptographic keys and initialization vectors (Fischer & Bernard, 2011).

3.4. Squeeze cipher

On top of the LZW algorithm, a secure squeeze cipher encryption scheme is designed with the following assumptions:

- High security: -The encryption algorithm plans to use random swap and random insert functions that provide high security to protect the LZW compression (Kelley & Tamassia, 2014).
- Light encryption cost: - It aims to reduce the high encryption cost that comes along with the traditional first compress then encrypt way of first compressing and only then encrypting (Xie & Kuo, 2005).

The squeeze cipher is designed to combine encryption and compression of texts by attaching encryption as a post-processing to the LZW compression algorithm. Its central idea is on developing a way of handling the dictionary formed in the process of compression. The dictionary constructed will have the same entries as in LZW though the entries are in some random order. The initial dictionary is filled with all the single characters, but unlike to LZW their entry is picked arbitrarily. The dictionary entries are partially altered for each step of the algorithm. The dictionary entries are permuted by a pseudo-random number generator (PRNG) function G. This means from a set of predefined algorithm G produces a sequence of random integers that is indistinguishable from a true random but on real sense it is a deterministic process. It follows that encryption can be achieved by changing correct indices to some other random index value (Xie & Kuo, 2005). To process the squeeze cipher encryption an initialization vector (IV), sometimes called a nonce is required at the beginning. An IV is an arbitrary number that can be used along with the secret key for data encryption. It avoids repetition in data encryption so as to make life difficult for a cracker to break a cipher. One instance could be if there are repeated sequences in the encrypted file, an attacker may assume that their corresponding sequences in the plain message were also identical. In such cases, the IV will prevent the appearance of identical duplicate sequences in the ciphertext (TechTarget, 2014). The squeeze cipher algorithm requires the seed for the pseudo-random number generator G to be constructed at both ends, thus we employ pseudo random-permutation (PRP[3]) f for this purpose.

Given a PRP function f:

$$f: \{0,1\}^{\lambda} \times \{0,1\}^{\lambda} \rightarrow \{0,1\}^{\lambda}$$

where λ is the length of the IV, and its inputs are the shared common key k and the IV, it computes the seed sd of the PRNG as $\mathrm{f_K(IV) = sd}$ (Kelley & Tamassia, 2014). The IV and K have the same length and the key K is big enough to provide security. The output of this function is computationally indistingushable from random. As long as we keep the key K secret, even if we send the IV on the clear, the chance of constructing the same seed by an attacker is significantly low or impossible. Because of this, the IV is attached to the output of the compressed data that makes decryption to be possible for the receiver of the message to generate the same initial seed as the sender.

Let the first source alphabet be $\mathrm{A} = \{\mathrm{a_1, a_2, \cdots, a_N}\}$. let us also assume that the input string, which is to be encoded, has a sequence $\mathrm{M} = \mathrm{m_1 m_2 \cdots m_L}$. Assume that we use $\mathrm{z} = 12$-bits to denote each dictionary index i.e. the size of the initial dictionary is $2^{\mathrm{z}} = 4096$. Let sd be the seed generated from the cryptographically secure PRNG G. From sd we then create a pseudo-random key stream k. The squeeze cipher encoding procedure is as follows.

Step 1: We first construct an empty dictionary with 2^{z} entries. Then randomly N entries are selected according to a random key k and those entries are filled, one at the time with $\mathrm{a_i}$, with $1 \leq \mathrm{I} \leq \mathrm{N}$. Recall in the LZW scheme that only the first N entries are assigned to $\mathrm{a_i}$, where $1 \leq \mathrm{i} \leq \mathrm{N}$. In this case, we obtain a more sparsely populated dictionary.

Step 2: We read the characters from M one by one. After each character c is read and concatenated to the prefix, the dictionary is searched for $\mathrm{prefix} \,||\, \mathrm{c}$. If the $\mathrm{prefix} \,||\, \mathrm{c}$ is found in the dictionary, then prefix becomes $\mathrm{prefix} \,||\, \mathrm{c}$, and the process continues. However, when prefix is in the dictionary, but string $\mathrm{prefix} \,||\, \mathrm{c}$ is not, we move to the next step.

Step 3: We perform the following:-

i. Output the dictionary index that points to the prefix.

ii. Swap the prefix entry with another entry selected at random.

iii. Randomly select an empty entry in the dictionary and insert $\mathrm{prefix} \,||\, \mathrm{c}$ into that entry.

iv. Initialize prefix to c.

In LZW algorithm, the string $\mathrm{prefix} \,||\, \mathrm{c}$ is inserted into the dictionary entry next to the entry used by the previous strings in a sequential way.

As proposed by different authors (Kelley & Tamassia, 2014), (Xie & Kuo, 2005) and (J. Zhou, et.al., 2008), the significant difference between the randomized dictionary and the traditional LZW algorithm boils down to the following :

[3] PRP- is a permutation selected at random from the family of all permutations.

I. Initial random dictionary: - In the squeeze version, we arrange the first dictionary into a random order different from the order in LZW. The order follows a random permutation, which is generated by the pseudo-random cryptographic algorithm (Xie & Kuo, 2005).

II. Random swap and insert: - Here, we select an arbitrary position to a prefix that was matched and exchange it. Also, new entries are not added to the dictionary by sequential order as in LZW rather an empty position is selected at random from the entries of the dictionary (J. Zhou et al., 2008).

The detailed algorithm and implementation of the squeeze cipher are found in the paper by (Kelley & Tamassia, 2014). To help us better understand the squeeze cipher algorithm; let us work out some of the preliminary steps to show the significant difference between LZW and the squeeze cipher. Like the LZW, the initial dictionary contains all the entries but in a random order.

Index	**Entry**
0	\15
1	\3E
…	…
255	\3A
256	-
…	…
511	-

Table 4: Initial Squeeze dictionary

We first match first \3A, and output index 255 and then swap entry at index 255 with a random position, say the entry at index 5. Then, insert \3A\3A into a random empty position, say entry 10.

Index	**Entry**
0	\15
1	\3E
5	\3A
….	
10	\3A\3A
….	
255	\14

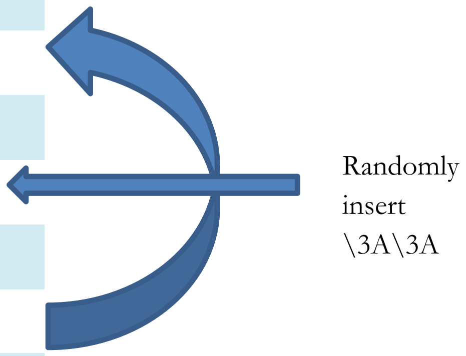


Table 5: Squeeze dictionary after first iteration

Then next we match character \3A, output index 5 and then swap entry at index 5 with a random position, say the entry at index 100. Then, insert \3A\15 into a random empty position; say the entry at index 20.

Index	Entry
0	\15
1	\3E
…	
5	\12
20	\3A\15
…	
100	\3A

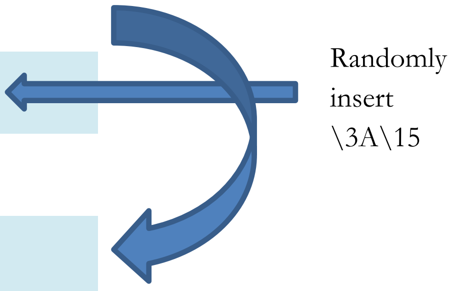


Table 6: Squeeze dictionary after second iteration

The process continues and generates a sequence of indexes. To enable the decryption process, the initialization vector will be added to the set of indexes during communication.

3.5. Geostatistics

Geostatistics is an area of statistics that deals with spatially distributed data. It differentiates from classical statistics in the assumptions that data are correlated, as nature produces only in rare occasion's phenomena that are not spatially associated. The goal of spatial analysis is to model spatial relationship between observations at different locations and use them in the process of interpolation (Vanloocke, 2012). The famous interpolation practice that relies on spatial analysis is kriging.

3.5.1. Variogram and kriging

In order to make estimations based on correlation between spatial data, one has to know how the variable at the location to be estimated correlates with this variable in the surrounding area (Vanloocke, 2012). Suppose that the variance between the observation $Z(x), Z(x+h)$ which are located at x and $x+h$ respectively depends only on the spatial distance between the observations h i.e. the variance is a function only of h. The variogram function $\gamma(h)$ is defined as:

$$\gamma(\mathrm{h}) = \frac{1}{2}\mathrm{E}\{[\mathrm{z(x)} - \mathrm{z(x+h)}]^2\}$$

It is simply the expected squared difference between two data values separated by h. According to (Cressie, 1993) the first estimate of the variogram is computed from the experimental variogram. The experimental variogram measures the average degree of dissimilarity between unsampled values and a nearby data value (Deutsch & Journel, 1998) and thus can depict autocorrelation at various distances. The value of the experimental variogram for a separation distance of lag h is half the average squared difference between the value of $\mathrm{z(x_i)}$, and the value of $\mathrm{z(x_i + h)}$ (Robinson & Metternicht, 2006). Consider the following observations z_i , $i = 1, \dots, n$ at the locations $x_1, x_2, \dots, x_n$ the empirical variogram $\hat{\gamma}(h)$ is defined as:

$$\hat{\gamma}(h) = \frac{1}{2N(h)} \sum_{i=1}^{N(h)} [z(x_i) - z(x_i + h)]^2 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \text{Eq. (1)}$$

Where $N(h)$ is the set of pairs separated by h, $z(x_i)$ and $z(x_i + h)$ are the data values at locations x and $x + h$, respectively. The experimental variogram increases from small values near the origin to larger values as h increases. The function becomes stable for large h values; this maximum value is called the sill**,** and it represents the total variance. The lag h, for which the sill is reached, is referred to as the range beyond this value the observation becomes independent. The value at the origin is called the nugget. If the values of $z(x_i)$ and $z(x_i + h)$ are autocorrelated the result of Eq. (1) will be insignificant compared to an uncorrelated pair of points. Commencing the study of the experimental variogram, the appropriate model such as spherical, exponential is then fitted, and the parameters range, nugget and sill are used in the prediction process (Robinson & Metternicht, 2006).

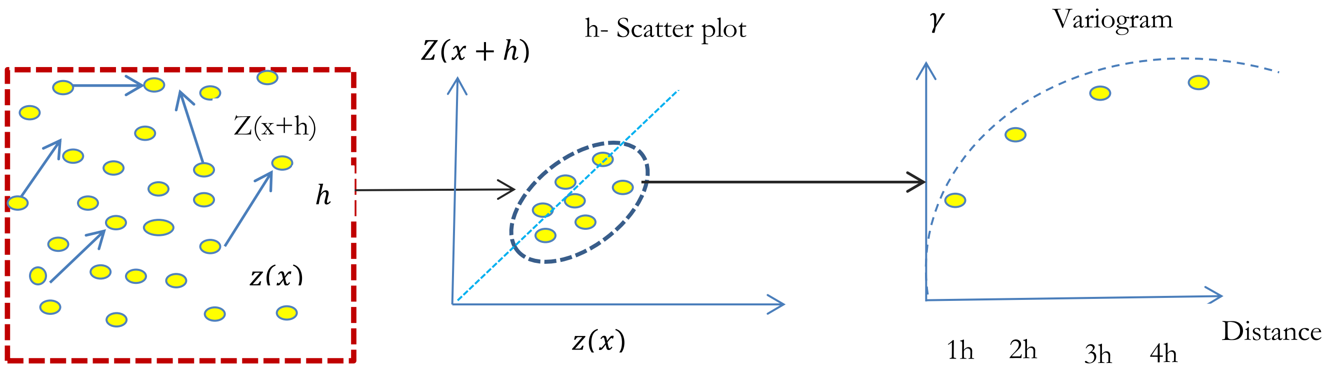


Fig 1.5.: The construction of an experimental variogram (Vanloocke, 2012)

Kriging is an interpolation procedure with the objective of providing estimates that are statistically close to the unknown true value at any particular location. In order to do kriging for estimation of data at unsampled locations, we assume the variance of the spatial data depends merely on the lag h and the mean is constant, these two assumptions are referred to as intrinsic hypothesis. Kriging is a linear regression method for estimating values at any location of a region. It calls for the variogram model of spatial correlation. The outcome of kriging is the kriging mean and kriging variance calculated for every single point in the study area. Nevertheless, the spatial distribution of kriging estimates tend to be too smooth (Arpat & Caers, 2007).

3.6. Geostatistical simulation

Simulation requires a model be developed; the model represents the behaviors of the selected physical or abstract process. The simulation then replicates reality using a model defined a priori (J., Carson, et.al., 2001). Geostatistical simulation maintains the variance observed in the data unlike to interpolation that preserve the mean (PetroWiki, 2014). Specifically Gaussian Geostatistical Simulation(GGS) is suitable for continuous data. GGS assumes the mean, variance, and variogram do not change over the spatial domain of the data. Sequential Gaussian Simulation(SGS), comes with a solution to the smoothing complication of kriging (Arpat, 2005) by enhanced descriptions of the local variability. Consequently, a simulation map characterizes the spatial distribution of the data more accurately than a kriged map. Geostatistical simulations try to replicate the entire random field, and conditional simulation does it while preserving the data. The conditional simulation procedure generates an array of simulated values consistent with known values at measured locations, and with a given variogram model (Delbari, Afrasiab, & Loiskandl, 2009). Unlike kriging interpolation, the goal of conditional simulation is to represent the inherent spatial variability of a medium. It, for this reason, reproduces the second-order statistical attributes such as the

mean value and auto-covariance function of the considered dataset. One should note that; conditional simulation does not accurately replicate the complete random field, but it provides an impression of what it might be like. Further, conditional simulations have many desirable applications in mining, soil science, hydrology, environmental control, agriculture and other earth sciences (Journel, 2008).
Simulations evaluate the uncertainties in unsampled values of a spatial attribute at a single and several multiple locations (Emery & Lantuéjoul, 2006). It leads to the two mainstream families of conditional simulations: pixel-based and object-based simulation (Liu, 2006). Pixel-based methods are important to reproduce the variogram (R.Gebbers & Bruin, 2010), but are unable to reproduce shapes and patterns that exist in the simulations and have difficulty in reproducing complex geometric shapes(Remy, Boucher, & Wu, 2009). Some of the most traditional pixel-based algorithms include the turning bands, sequential Gaussian, sequential indicator and simulated annealing (PetroWiki, 2014). On the other hand, object-based approaches function on sets of pixels that are coupled and organized to represent shapes of features (PetroWiki, 2014). The object-based methods simulate many grid nodes at one time. Their realizations are built by releasing one object or pattern at a time onto the simulation grid; hence they are more close to the geometry of the object (Remy et al., 2009). They do not generate values at the nodes of a pre-defined grid, but rather produce shapes in space according to some probability laws (Allard, Froidevaux, & Biver, 2007). However, in contrast to pixel-based modeling respecting conditioning data is more complicated.

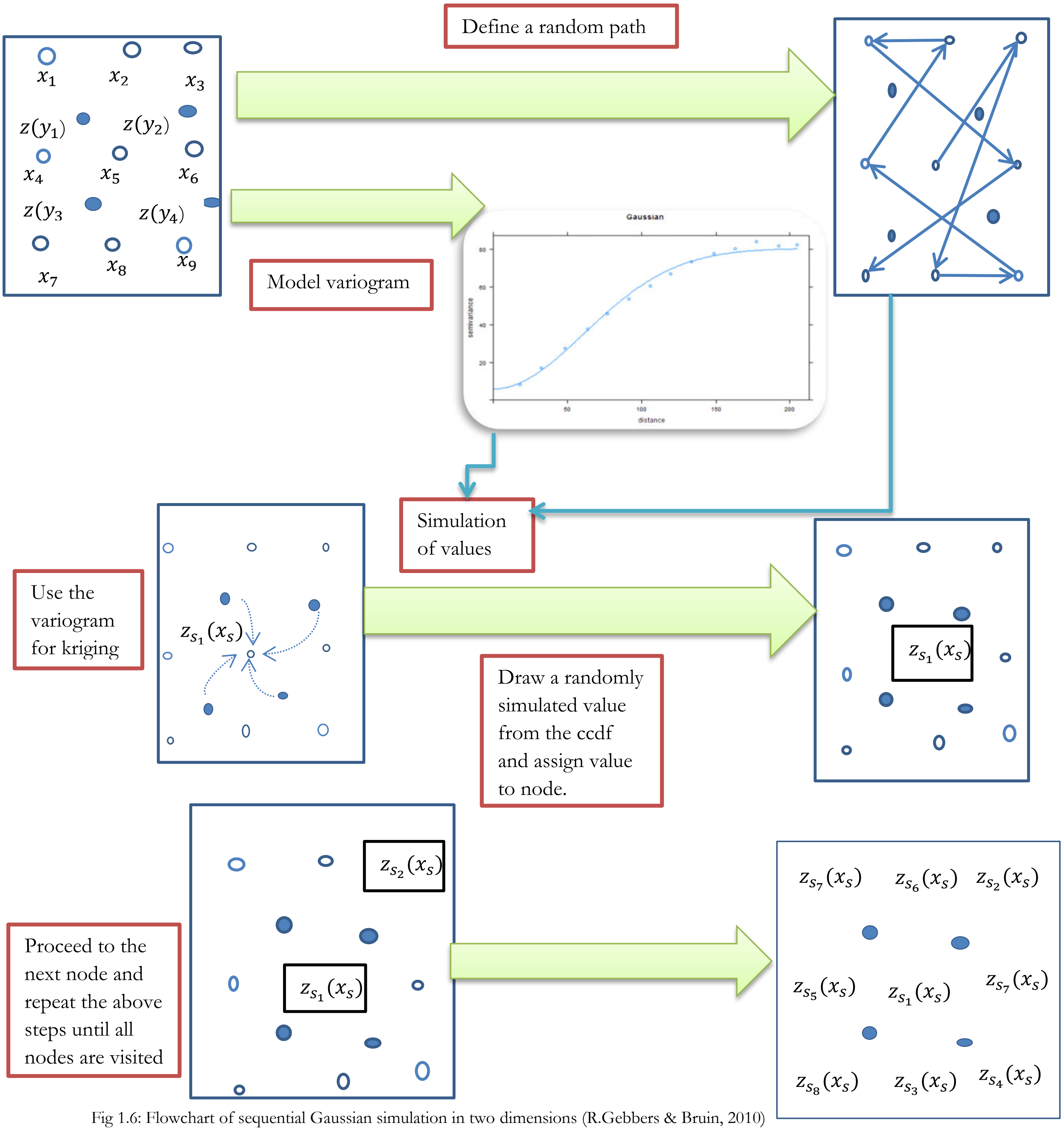
3.7. Sequential Gaussian Simulation (SGS)

In this report, we apply the SGS algorithm to generate realizations. SGS is widely used to model spatially distributed continuous variables because it is simple, fast and efficient (Pyrcz & Deutsch, 2014). Sequential simulation reproduces the histogram and spatial covariance of the attributes being simulated (Arpat, 2005). Their implementations consist of reproducing the desired spatial properties through the sequential use of conditional distributions. Consider a set of N random variables $Z(u_i), i = 1,\dots,N$ defined at N locations u_i. The aim is to generate L joint realizations $\{z^l(u_i), i = 1,\dots,N\}$ with $l = 1,\dots,L$ of the N random variables, conditional to n available data and reproducing the properties of a given multivariate distribution. To achieve this goal, the N-point multivariate distribution is decomposed into a set of N univariate conditional distributions:

$$\begin{aligned} F\,(u_1,\dots,u_N;\, z_1,\dots,z_N\,|(n)) &= F\,(u_N\,;\, z_N\,|(n + N - 1)) \times \\ &F\,(u_N - 1;\, z_N - 1|(n + N - 2)) \times \dots \times \\ &F\,(u_2;\, z_2|(n + 1)) \times F\,(u_1;\, z_1|(n)) \end{aligned}$$

Where $F\,(u_N\,;\, zN\,|(n + N - 1)) = Prob\,\{Z(u_N) \le zN\,|(n + N - 1)\}$ is the conditional cdf of $Z(u_N)$ given the set of n original data values and the $(N - 1)$ realizations $Z(u_i), i = 1,\dots,N$ of the previously simulated values. This decomposition of equation is analytically available for multivariate Gaussian distribution. The governing spatial law of the Gaussian model is characterized by the variogram model (Arpat, 2005).
SGS starts by assigning the conditioning data to the grids (Pyrcz & Deutsch, 2014). Next, it selects one grid node at random and applies simple kriging at the selected location by using neighborhood data and previously simulated values. The simulated value assigned to this grid node is drawn at random from a normal distribution whose variance and mean are set to the kriged variance and the kriged value, respectively. We add this new value to the conditioning data. We then select a new grid node at random, and we repeat the process until all the nodes are visited. These steps provide the first realization. Then, the same process is repeated to produce several realizations using a different random number sequence (Syed, 2003). To reconstruct the same realization, the seed of the pseudo-random number generator utilized in the random selection needs to be retained as it allows generating a realization by sequentially visiting each node on the simulation grid. Suppose that we have data at locations $z(y_1), z(y_2), z(y_3),$ and $z(y_4)$ and we would like to simulate the values at the locations $x_1, x_2, \dots, x_9$ shown in Figure 3.7. The SGS algorithm

first defines a random path visiting each node of the grid. For each node, u, we collect the conditioning data, consisting of neighboring original hard data and previously simulated values. We then draw a value from that Gaussian cdf and add the simulated value of the data set to construct realizations. At each simulation grid, to determine the Gaussian cdf, the algorithm uses simple kriging. In case, the random function $\mathrm{Z(u)}$ is non-Gaussian it needs to be converted into a Gaussian random function $\mathrm{Y(u)}$. The simulated values are then be back-transformed (Nicolas, Alexandre, & Jianbing, 2009).



Define a random path
x_1 x_2 x_3
$z(y_1)$ $z(y_2)$
x_4 x_5 x_6
$z(y_3)$ $z(y_4)$
x_7 x_8 x_9
Model variogram
Gaussian
semivariance
distance
Simulation of values
Use the variogram for kriging
$z_{s_1}(x_s)$
Draw a randomly simulated value from the ccdf and assign value to node.
Proceed to the next node and repeat the above steps until all nodes are visited
$z_{s_2}(x_s)$
$z_{s_7}(x_s)$ $z_{s_6}(x_s)$ $z_{s_2}(x_s)$
$z_{s_5}(x_s)$ $z_{s_1}(x_s)$ $z_{s_7}(x_s)$
$z_{s_8}(x_s)$ $z_{s_3}(x_s)$ $z_{s_4}(x_s)$

Fig 1.6: Flowchart of sequential Gaussian simulation in two dimensions (R.Gebbers & Bruin, 2010)

3.8. Geostatistical simulation validation

One method of validating geostatistical simulation is by performing cross-validation against a data set (Emery, 2008). Cross-validation is a way to evaluate the predictive ability of a model. The word model in this context refers to the entire geostatistical model, the mean, and the spatial dependence structure. Generally, cross-validation is achieved by eliminating information at a particular location and estimate the value at those locations by forming a model with the remaining data, and finally compute the difference between the actual and estimated value for each data location (Robinson & Metternicht, 2006). As (Cressie, 1993) points out, cross-validation is used to assure the model predictions are okay. Cross-validation practices were commonly used to validate the accuracy of an interpolation. To compare the results of the conditional simulations and the accuracy of the spatial analysis methods, we frequently examine the difference between the data values and the predicted data using the mean error and the root mean squared error. Moreover, the correlation coefficient which is a normalized measurement of how two variables in a data are linearly related is also used to compare the linear relationship between the variables. These indicators are an important parameter in GIS and remote sensing. Suppose $z(x_i)$ is the observed value and $\tilde{z}(x_i)$ is the predicted value for location x_i and N be the number of values in the dataset. The Mean error ME and the root mean square RMSE are defined as follows:

$$ME = \frac{1}{N}\sum_{i=1}^{N}(\,z(x_i) - \tilde{z}(x_i))$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(z(x_i) - \tilde{z}(x_i))^2}{N}}$$

The correlation coefficient is defined as $r = \frac{\sigma_{xy}}{\sigma_x\sigma_y}$.

In this formula σ_x and σ_y are the standard deviations of the variables x and y and σ_{xy} is the covariance. The correlation coefficient takes on values on the range between -1 and 1. If the simulation results are worthwhile at the end we expect a strong positive linear relationship between the simulated and sampled values, a mean error close to zero and a small root mean square error. In addition, different graphical plots such as scatter plots that measure the predicted values versus the original sample values, the spread of prediction errors, the qq plots and histogram analysis can be performed to validate the outcome of simulations.

After deciding which model we use, we can apply different encryption schemes on top of the model to ensure our communication is secured. In this study, we employed the advanced encryption scheme for encrypting and decrypting the file that is transmitted to the receiver.

3.9. Advanced Encryption Standard (AES)

AES, which is also known as Rijndael, was developed by Daemen & Rijmen in 2002 (Heron, 2009). It emerged from a competition in which cryptographers were asked to submit a secure encryption scheme. It was evaluated comprehensively and then accepted by NIST in 2001 for categorized communications (Westlund, 2002). Nowadays, it materialized as an attractive choice for encrypting secured data such as online transactions.

AES is a symmetric block cipher. So, information that is to be encrypted will be broken up into 128 bit blocks, and the operations will be carried out in blocks. The standard implementation of AES applies 128 bit block size (16 bytes). The 16 bytes are treated as a 4 x 4 byte matrix. This matrix is referred to as the state array. Padding is added if the information is not a multiple of 128bits. The algorithm works for different encryption key lengths, and here we use a key of 128 bits. This key is also ordered in the form of a matrix of 4 × 4 bytes. The encryption consists of 10 rounds of processing. All other rounds are identical except the last round. Encryption and decryption consist of identical procedures, but the order to perform the procedures is different.

Before all processing begins, the input plain block is XORed with the first round key(the round keys come from a 16 byte AES key using key expansion), and then we apply certain functions that include substitution, permutation, and linear transformations. Then XORed with the next round key, we do that again and again ten times with the exception of the last round, where the Mixcolumn step is missing. In every phase, we keep the four by four matrices, so the size of the output is 16 bytes. The overall structure of AES encryption is shown in Figure 1.7.

BytesSubs is a static byte-wise substitution and can be represented by a single lookup table. We took the plain128 bit block byte by byte and replace them according to the table. The entries in the lookup table are formed with the ideas of multiplicative inverses in the $GF(2^8)$.[4] The result goes to ShiftRows, here the transformation operations do not shift the first row of the 4 x 4 matrix but they rather shift the remaining three rows. Shift the second row of the matrix by one byte to the left; shift the third row of the matrix by two bytes to the left and finally shift the last row of the matrix by three bytes to the left (Kak, 2014). The result goes to MixColumns, which is a more complicated substitution. Instead of doing it byte-wise substitution it is substituting four bytes by another four byte. Thus, the initial 128-bit plain block undergo mixing and turning and get more complicated treatments as a result it's more difficult to take the cipher text and roll it back to obtain the plain block.

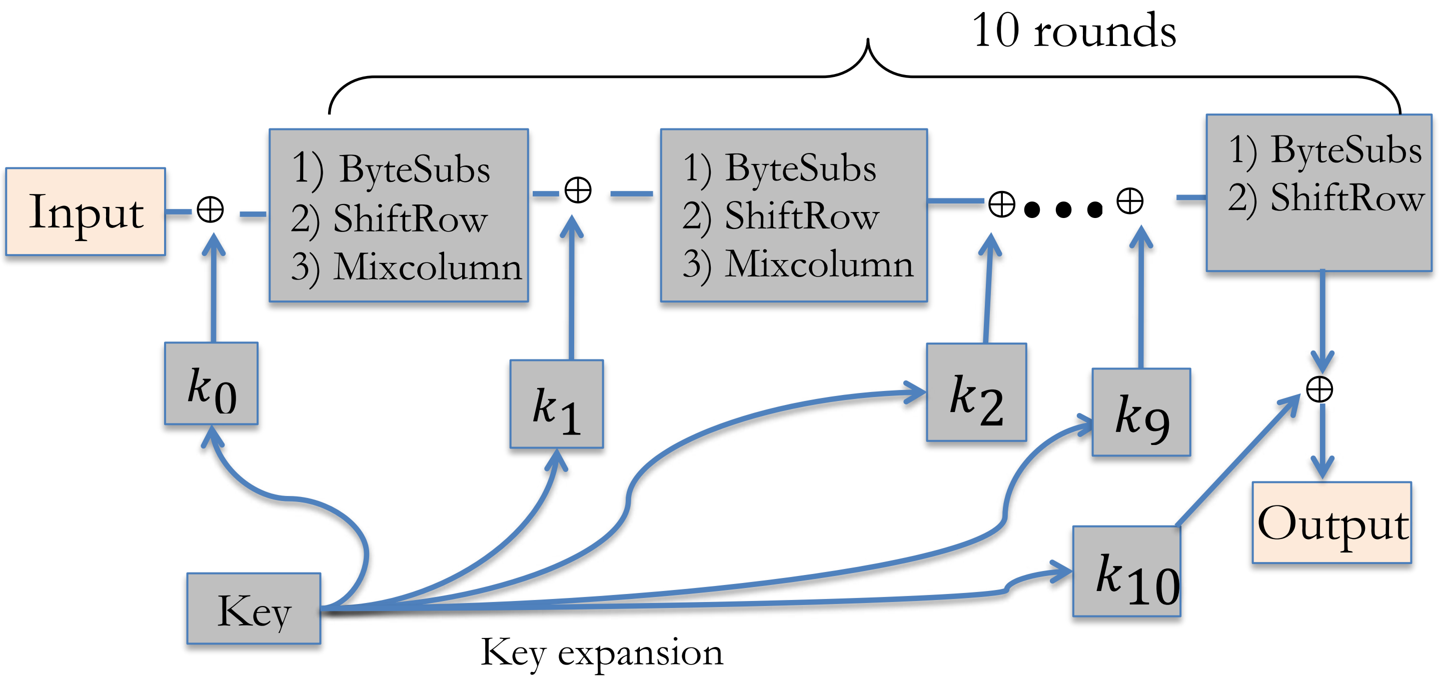


Fig 1.7 AES-128 encryption process

We encrypt large files by partitioning them into several blocks of size 18-bits and choose the respective mode of encryption. A mode of operation define how these several blocks are linked with each other to transform larger data (Menzes, Oorschoot, & Vanstone, 1996). The mode of operation selected determines the security of encryption. Different encryption methods such as the electronic code book (ECB), Cipher Block Chaining (CBC), Cipher Feedback (CFB), Output Feedback (OFB) and Counter

[4] $GF(2^8)$ is a finite field consisting of 256 elements.

mode (CTR) are supported for symmetric-key block ciphers (Paar, Pelzel, & Preneel, 2011). Most cryptographers recommend the CTR mode for encrypting large files as it is appropriate to function on a multi-processor device where blocks are encoded in parallel. The cipher operation takes place for each block using a secret key and a unique counter. The counters are started from an initial random value and increment it to process the next blocks; this ensures uniqueness of the cipher. In CTR mode, the counter bits are encrypted together with the secret key and then XORed with the input plaintext block. So the input plaintext is not directly encrypted. Because each block in CTR mode is independent of each other, CTR mode runs more rapidly than other modes of AES without losing the security requirements of a secure cipher. Usually, a unique nonce is joined with the counter to make sure the same plaintext generate different ciphertexts during encryption. This number is prepended to the ciphertext to facilitate decryption.

3.9.1. AES CTR mode on OpenSSL

Down to known tested experiments against ill-natured attacks of ciphers this paper used the AES counter mode to safeguard the content of the file transmitted. Recall that spatial simulation requires the input data i.e. the selected samples together with their coordinates and the variogram model that quantify the spatial variability of the data. These two together with the header file form the text file that requires protection during communication to the receiver. We then just encrypt the resulting file. The whole process of encryption and compression will then take the form shown in Fig 1.8.

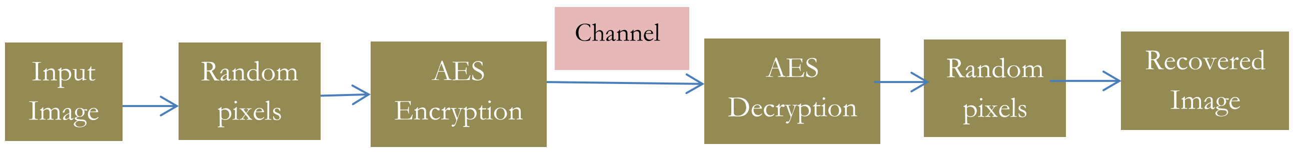


Fig 1.8 The AES-128 encryption combined with the spatial compression

OpenSSL is one of the most broadly used cryptographic library in the open source environment (Parziale et al., 2013). It implements different cryptographic procedures such as symmetric key encryption, public key encryption, hash functions, digital signatures, random number generation, etc. Thus, we can utilize OpenSSL library and encrypt a file easily and quickly. OpenSSL toolkit comes installed with Ubuntu and includes a feature that offers command-line access to the most important cryptographic processes of the library. To see the available ciphers we can type 'openssl –h'. In general, for encrypting large amounts of data, less computationally demanding algorithms are chosen. To encrypt a file using OpenSSL we enter information in this format: 'openssl enc –cipher –salt -in file.txt -out encryptedfile.txt.enc'. Where, openssl is the command line tool, enc stands for encoding with ciphers, cipher represents the encryption cipher to be used, -salt defines the use of salt in key derivation process that add strength to the encryption -in file.txt specifies the input file and -out encryptedfile.txt.enc specifies the output file. Then we enter a password twice. Finally, the data will be encrypted and saved. To decrypt the file, we type 'openssl enc –cipher -d -salt -in encryptedfile.txt.enc -out file.txt.new'. The data receiver needs to know the password and the type of the encryption cipher used.

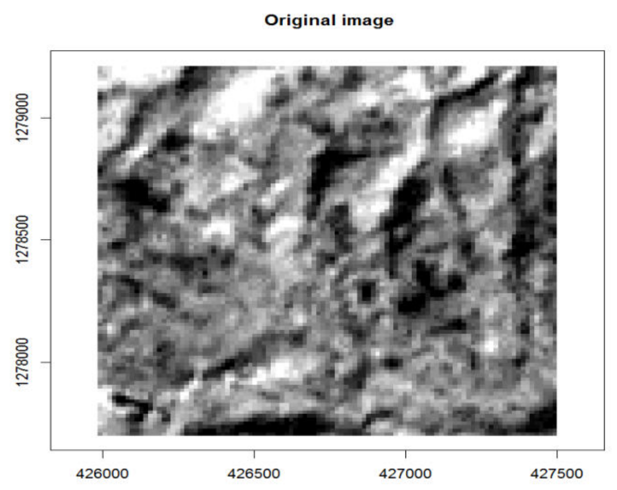
Note that salt is a random 8 byte string that is deposited on the header of the encrypted file with the purpose of altering the encrypted file when its encrypted frequently with a particular password. The salt and password are used in the computation of the key and IV. Upon decryption, the salt will be recovered and together with the password they will generate the key and IV. In this paper, we use the AES-128 counter mode algorithm for encrypting and decrypting data.

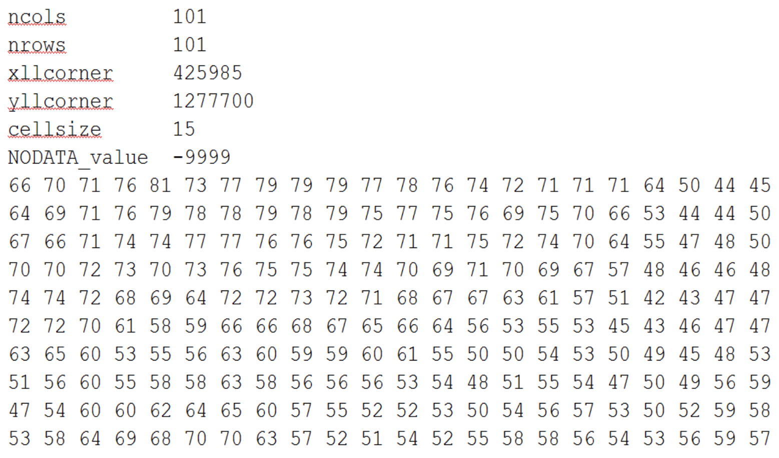
4. IMPLEMTATIONS AND RESULTS

This chapter discusses the implementation and obtained results. The study area is a square size of side length, 101. We present here the experiments that resulted from the different methods we used.

4.1. LZW outputs

In this section, we present the outcome of the LZW compression and decompression applied to the study area. The LZW algorithm implemented in Python takes as input a text file and compresses it. For that, we convert the raster dataset into an ASCII file that represents the raster data. The structure of the output ASCII file consists of header information containing the number of columns, number of rows, coordinates of left corner, cell size and the cell values in row order.





```
ncols         101
nrows         101
xllcorner     425985
yllcorner     1277700
cellsize      15
NODATA_value  -9999
66 70 71 76 81 73 77 79 79 79 77 78 76 74 72 71 71 71 64 50 44 45
64 69 71 76 79 78 78 79 78 79 75 77 75 76 69 75 70 66 53 44 44 50
67 66 71 74 74 77 77 76 76 75 72 71 71 75 72 74 70 64 55 47 48 50
70 70 72 73 70 73 76 75 75 74 74 70 69 71 70 69 67 57 48 46 46 48
74 74 72 68 69 64 72 72 73 72 71 68 67 67 63 61 57 51 42 43 47 47
72 72 70 61 58 59 66 66 68 67 65 66 64 56 53 55 53 45 43 46 47 47
63 65 60 53 55 56 63 60 59 59 60 61 55 50 50 54 53 50 49 45 48 53
51 56 60 55 58 58 63 58 56 56 56 53 54 48 51 55 54 47 50 49 56 59
47 54 60 60 62 64 65 60 57 55 52 52 53 50 54 56 57 53 50 52 59 58
53 58 64 69 68 70 70 63 57 52 51 54 52 55 58 58 56 54 53 56 59 57
```

Fig 1.9: Original image and its corresponding ASCII text file representing raster data.

We observe that satellite images contain repetitive sequences of DN values; the LZW compression algorithm focuses on removing such repeating sequences. To reconstruct the image from the uncompressed file we use the conversion tool, ASCII to Raster in ArcGIS. The LZW implementation in Python takes as input the study area, which is of size 30.2 KB and compresses it to 9.04 KB, with a compression ratio of approximately 1:3.

To check how similar the decompressed and the original satellite image are, we calculate the difference between the two images on a pixel by pixel basis and compute the root mean square error (RMSE). The difference image is a zero image, with all values zero. Thus, the RMSE is zero. Fig 1.10 shows the histogram of the two images; they are identical. No error has been registered in the decompression process.

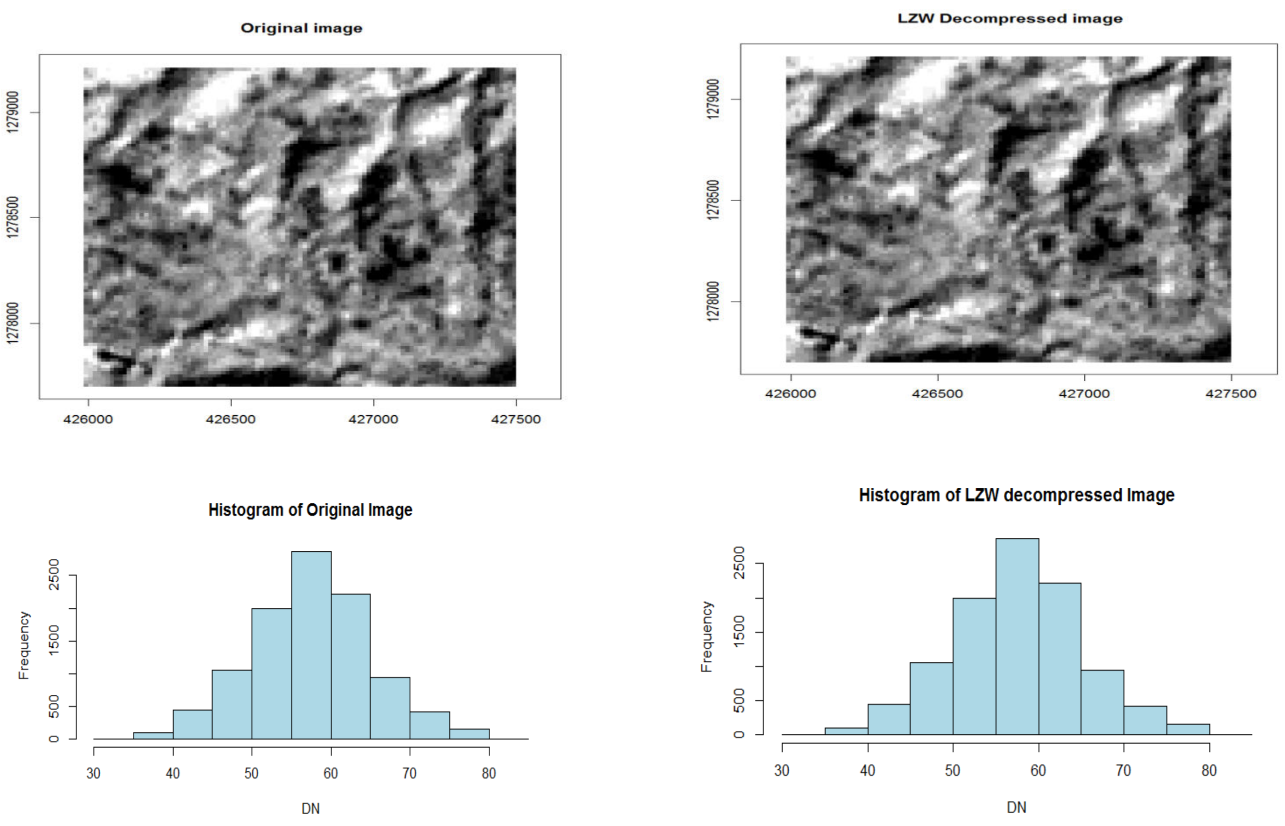
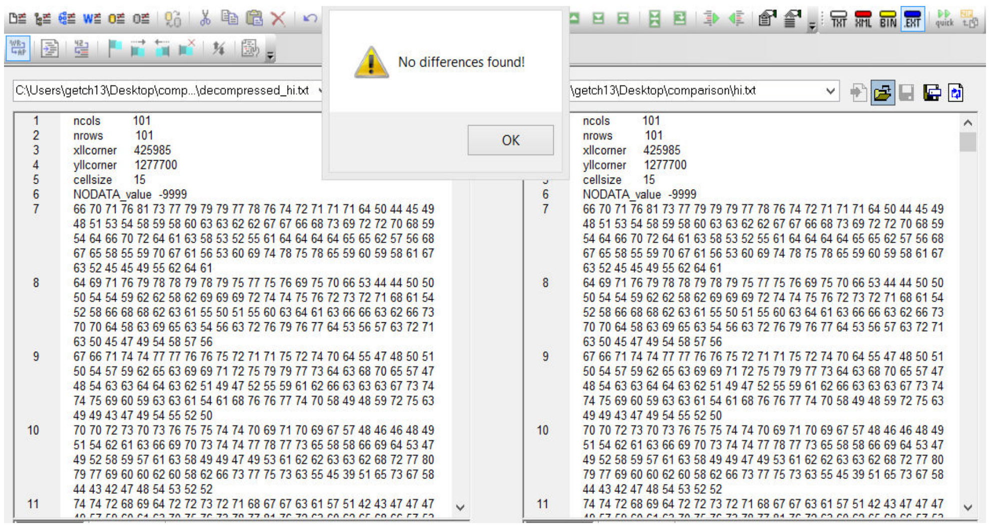


Fig 1.10 Histogram match of the original and LZW decompressed image

Alternatively we can use a change detection algorithm in ERDAS Imagine that compares two satellite images and or we can use comparison software such as Altova Diff Dog, Winmerge, or Linux's diff. A screenshot of the results of the Altova DiffDog are presented in Fig 1:11, it shows that LZW is a reversible process with no information loss.



No differences found!

OK

```
C:\Users\getch13\Desktop\comp...\decompressed_hi.txt

ncols        101
nrows         101
xllcorner    425985
yllcorner    1277700
cellsize     15
NODATA_value  -9999
66 70 71 76 81 73 77 79 79 79 77 78 76 74 72 71 71 71 64 50 44 45 49
48 51 53 54 58 59 58 60 63 63 62 62 67 67 66 68 73 69 72 72 70 68 59
54 64 66 70 72 64 61 63 58 53 52 55 61 64 64 64 64 65 65 62 57 56 68
67 65 58 55 59 70 67 61 56 53 60 69 74 78 75 78 65 59 60 59 58 61 67
63 52 45 45 49 55 62 64 61
64 69 71 76 79 78 78 79 78 79 75 77 75 76 69 75 70 66 53 44 44 50 50
50 54 54 59 62 62 58 62 69 69 69 72 74 74 75 76 72 73 72 71 68 61 54
52 58 66 68 68 62 63 61 55 50 51 55 60 63 64 61 63 66 66 63 62 66 73
70 70 64 58 63 69 65 63 54 56 63 72 76 79 76 77 64 53 56 57 63 72 71
63 50 45 47 49 54 58 57 56
67 66 71 74 74 77 77 76 76 75 72 71 71 75 72 74 70 64 55 47 48 50 51
50 54 57 59 62 65 63 69 69 71 72 75 79 79 77 73 64 63 68 70 65 57 47
48 54 63 63 64 64 63 62 51 49 47 52 55 59 61 62 66 63 63 63 67 73 74
74 75 69 60 59 63 63 61 54 61 68 76 76 77 74 70 58 49 48 59 72 75 63
49 49 43 47 49 54 55 52 50
70 70 72 73 70 73 76 75 75 74 74 70 69 71 70 69 67 57 48 46 46 48 49
51 54 62 61 63 66 69 70 73 74 74 77 78 77 73 65 58 58 66 69 64 53 47
49 52 58 59 57 61 63 58 49 49 47 49 53 61 62 62 63 63 62 68 72 77 80
79 77 69 60 60 62 60 58 62 66 73 77 75 73 63 55 45 39 51 65 73 67 58
44 43 42 47 48 54 53 52 52
74 74 72 68 69 64 72 72 73 72 71 68 67 67 63 61 57 51 42 43 47 47 47
```

```
\getch13\Desktop\comparison\hi.txt
```

```
ncols        101
nrows         101
xllcorner    425985
yllcorner    1277700
cellsize     15
NODATA_value  -9999
66 70 71 76 81 73 77 79 79 79 77 78 76 74 72 71 71 71 64 50 44 45 49
48 51 53 54 58 59 58 60 63 63 62 62 67 67 66 68 73 69 72 72 70 68 59
54 64 66 70 72 64 61 63 58 53 52 55 61 64 64 64 64 65 65 62 57 56 68
67 65 58 55 59 70 67 61 56 53 60 69 74 78 75 78 65 59 60 59 58 61 67
63 52 45 45 49 55 62 64 61
64 69 71 76 79 78 78 79 78 79 75 77 75 76 69 75 70 66 53 44 44 50 50
50 54 54 59 62 62 58 62 69 69 69 72 74 74 75 76 72 73 72 71 68 61 54
52 58 66 68 68 62 63 61 55 50 51 55 60 63 64 61 63 66 66 63 62 66 73
70 70 64 58 63 69 65 63 54 56 63 72 76 79 76 77 64 53 56 57 63 72 71
63 50 45 47 49 54 58 57 56
67 66 71 74 74 77 77 76 76 75 72 71 71 75 72 74 70 64 55 47 48 50 51
50 54 57 59 62 65 63 69 69 71 72 75 79 79 77 73 64 63 68 70 65 57 47
48 54 63 63 64 64 63 62 51 49 47 52 55 59 61 62 66 63 63 63 67 73 74
74 75 69 60 59 63 63 61 54 61 68 76 76 77 74 70 58 49 48 59 72 75 63
49 49 43 47 49 54 55 52 50
70 70 72 73 70 73 76 75 75 74 74 70 69 71 70 69 67 57 48 46 46 48 49
51 54 62 61 63 66 69 70 73 74 74 77 78 77 73 65 58 58 66 69 64 53 47
49 52 58 59 57 61 63 58 49 49 47 49 53 61 62 62 63 63 62 68 72 77 80
79 77 69 60 60 62 60 58 62 66 73 77 75 73 63 55 45 39 51 65 73 67 58
44 43 42 47 48 54 53 52 52
74 74 72 68 69 64 72 72 73 72 71 68 67 67 63 61 57 51 42 43 47 47 47
```

Fig 1.11: Screenshot of the results of the file comparisons using Altova DiffDog software

4.2. Squeeze cipher outputs

This section discusses the implementation of the squeeze cipher. As described in Chapter 3 the squeeze cipher is a lossless text compression and encryption method. The squeeze code is implemented in C and can take as input any binary file. The parameter we need to set is the size of the dictionary to be used. The dictionary sizes have entries 2^{12} and 2^{16} entitled as squeeze-12 and squeeze-16 respectively. The program implementation outputs the compressed and encrypted file, the compression ratio, and the elapsed time measured in MB/s. The Study Area is an 11.78 KB file (image in tiff format) that compresses to 9.1 KB binary file. As the input file contains modest repetitive information, there is a slight file reduction in the output data. The encrypted and compressed data can be reconstructed back to an image. Since images require header files, we can reapply the headers back to the encrypted and compressed output. The steps in Appendix B explain the process to put back the headers. The original and squeeze encrypted and compressed outputs are shown in Figure 1.12. The squeeze encrypted, and compressed image is completely random image that show no particular pattern that makes it infeasible for attacks during communications.

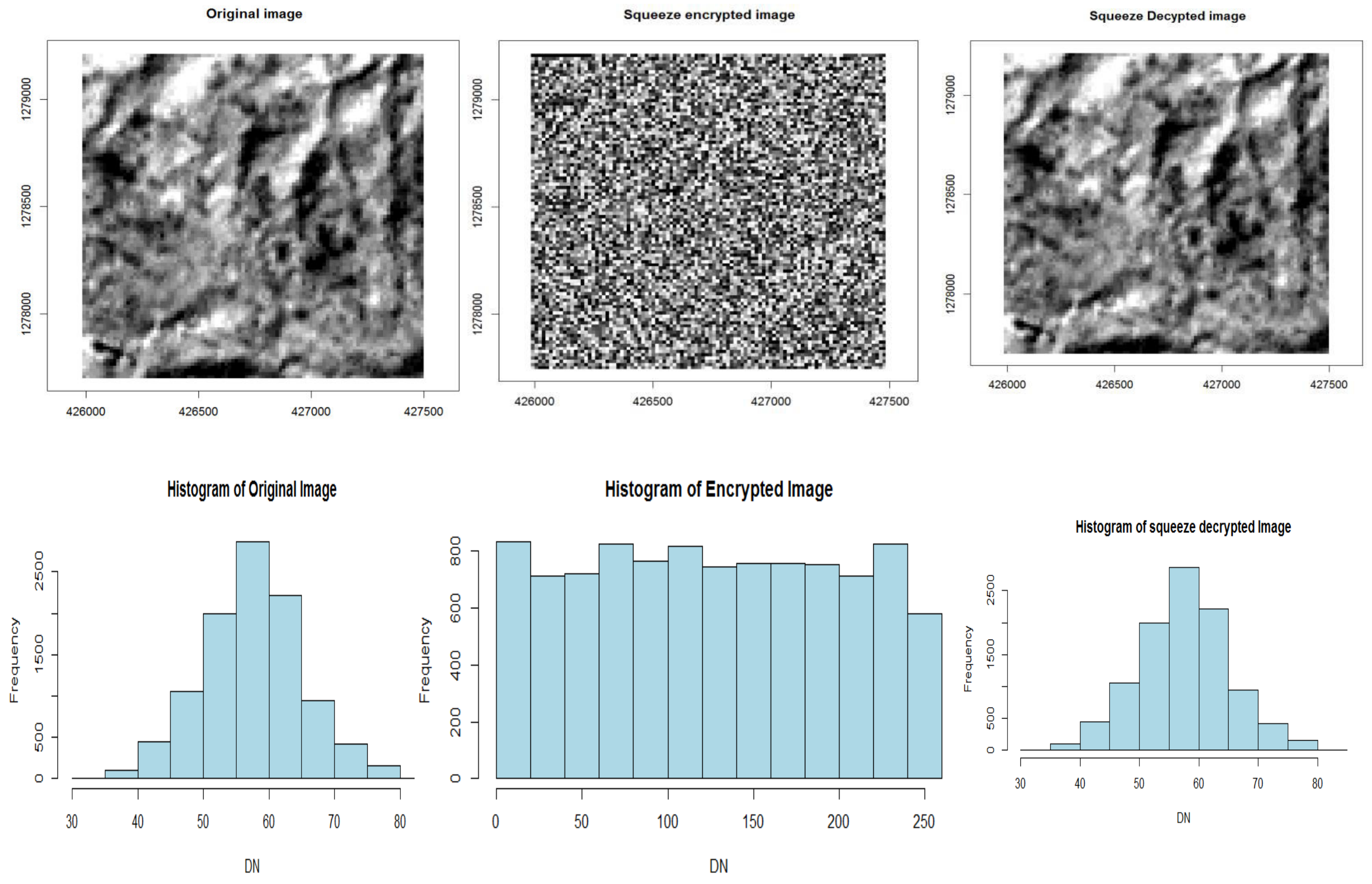


Figure 1.12 Experimental results of Squeeze cipher encrypted and compressed image.

Summary	Min	25%	Median	Mean	75%	Max
Original	33	53	58	58.09	63	83
Squeeze output	0	65	126	127.6	192	255

Table 7: Summary statistics comparison original image vs. the squeeze output.

As a result of the squeeze encryption, the DN values of the encrypted image are like random garbage values. In addition, the histogram and the summary statistics show that the squeeze has destroyed all the patterns that exist in the original image. Especially the histogram of the encrypted and compressed image is relatively uniform and significantly different from the histogram of the original image. Further analysis on the correlation between adjacent pixels, after they are compressed and encrypted, is shown in Fig.1.13. The correlation coefficient between two vertically and horizontally adjacent pixels of the original and squeeze encoded image together with their distribution support the claim that the encoded image has lost all configurations that occur in the original image. Consequently, it avoids any form of statistical attack.

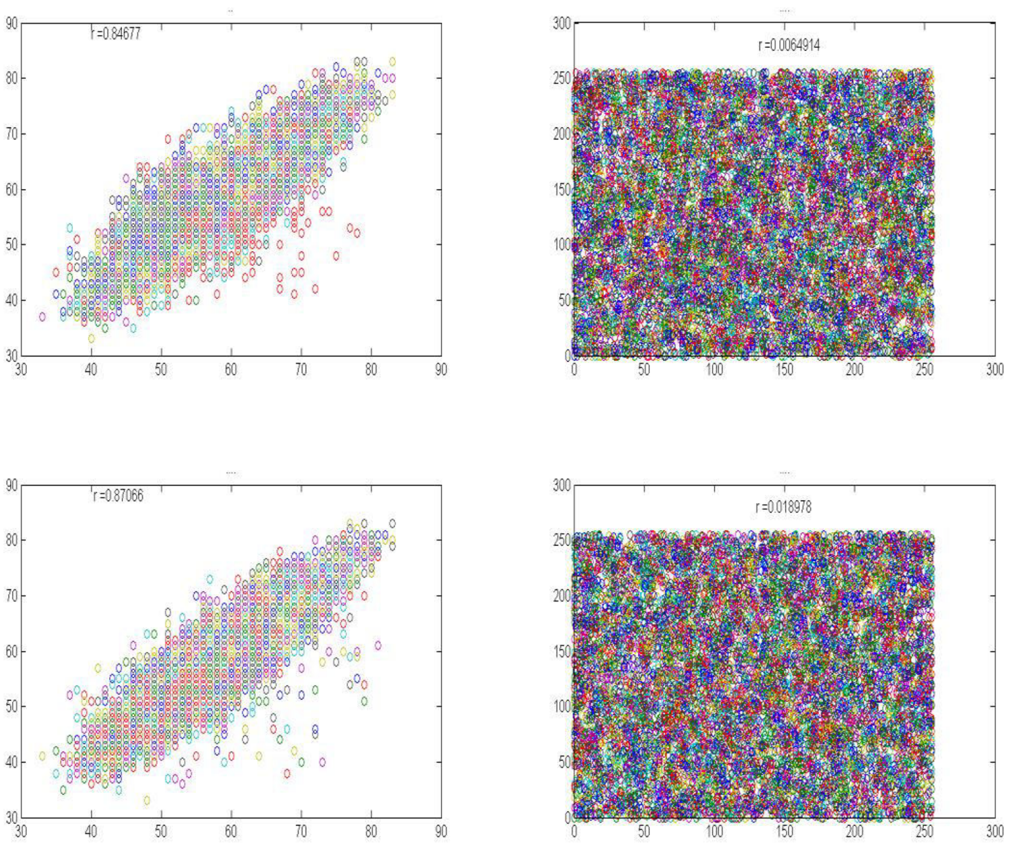
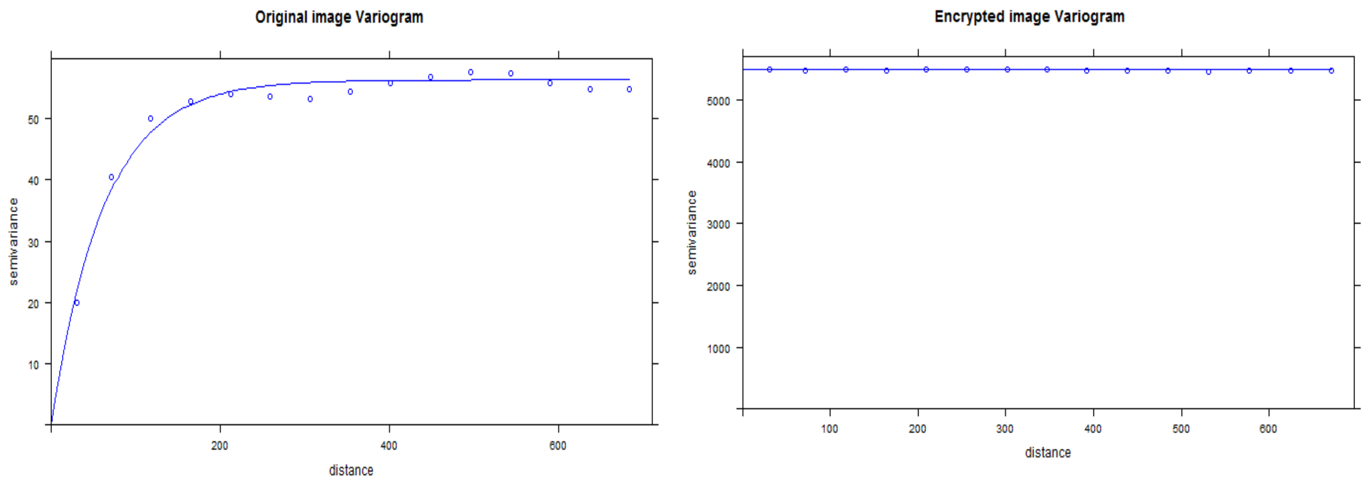


Fig 1.13: The first two top plots show the distribution of two horizontally adjacent pixels of the original image and the squeeze output image. Similarly, the bottom two plots show the distribution of two vertically adjacent pixels of the original image and the squeeze output image. The implementations of these plots are performed in Matlab.

Further, we compare the spatial structure of the original and the compressed and enciphered image by using variogram plots. The sample variogram points of the enciphered image fall on a straight line, and it results in a singular model. This is because of a nugget effect, i.e.; the squeeze encoded image has no spatial dependence. In other words, if we compute the covariance between the DN values at all distances h we will get zero.



	Model	psill	nugget	range
Original Image	Exponential	56.2778	0	62.67
Squeeze output	Exponential	0	5481.31	100

Fig 1:14 Variograms fit of the original and squeeze encoded image

4.3. Geostatistical simulations results

The study area image was processed in R (http://www.r-project.org/) statistical computing software and converted into a data frame with10201 observations of three variables. Figure 1.13 shows for the study area ten percent of the randomly selected pixels. These points are used as input points for the conditional simulation that tries to recover the remaining ninety percent of the missing data using Sequential Gaussian Simulation (SGS).

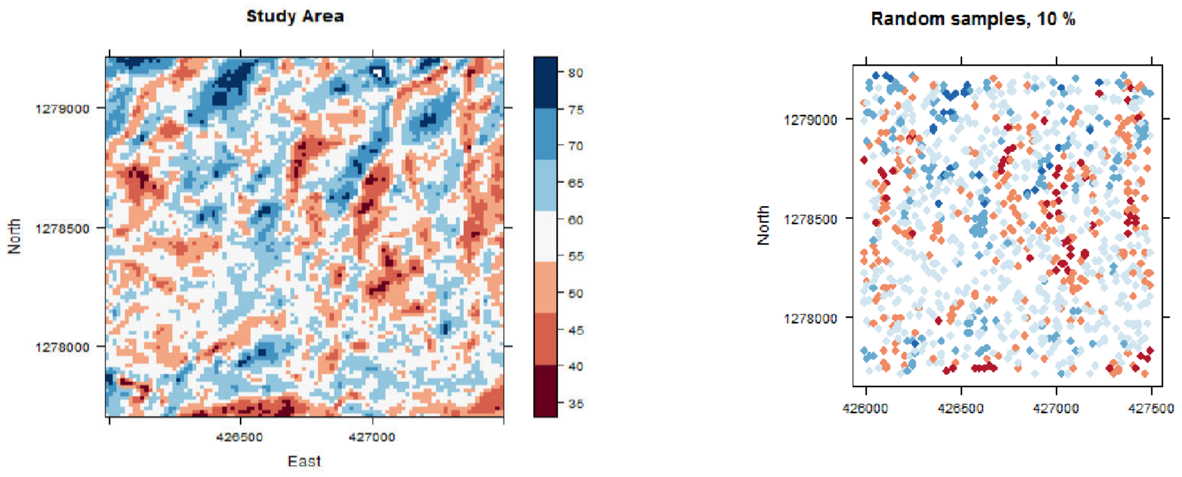


Fig 1.15: The study area and ten percent of the randomly selected pixels

The SGS algorithm we use is the one implemented in Gstat package in R, it is robust in using data and the simulated values in a local neighborhood to approximate the conditional distribution at that location. We can choose the number of nearest observations that contribute to the simulation process if we use a larger number of nearest observations the better the approximation but the slower the process. If we use a smaller number of nearest observations the simulation process will be faster. Gstat uses a bucket PR quadtree neighborhood search algorithm to select the closest observations data or previously simulated points (Pebesma, 2004).

4.3.1. Dataset analysis

If data are normally distributed and spatial stationary i.e. the mean and variance do not vary considerably in space then, geostatistical techniques can be applied. The first thing then is to test the data against these assumptions. If the data significantly differs from these assumptions, it will cause problems. In such cases, a transformation of the data is recommended. The scatter plots of the randomly selected pixels shown in Figure 1.16 consist of four subplots, which are an x versus y coordinate plot, data versus y coordinate plot, x versus data plot, and the histogram plot. By looking at the histograms of the subplots, severe deviations from normality are not observed.

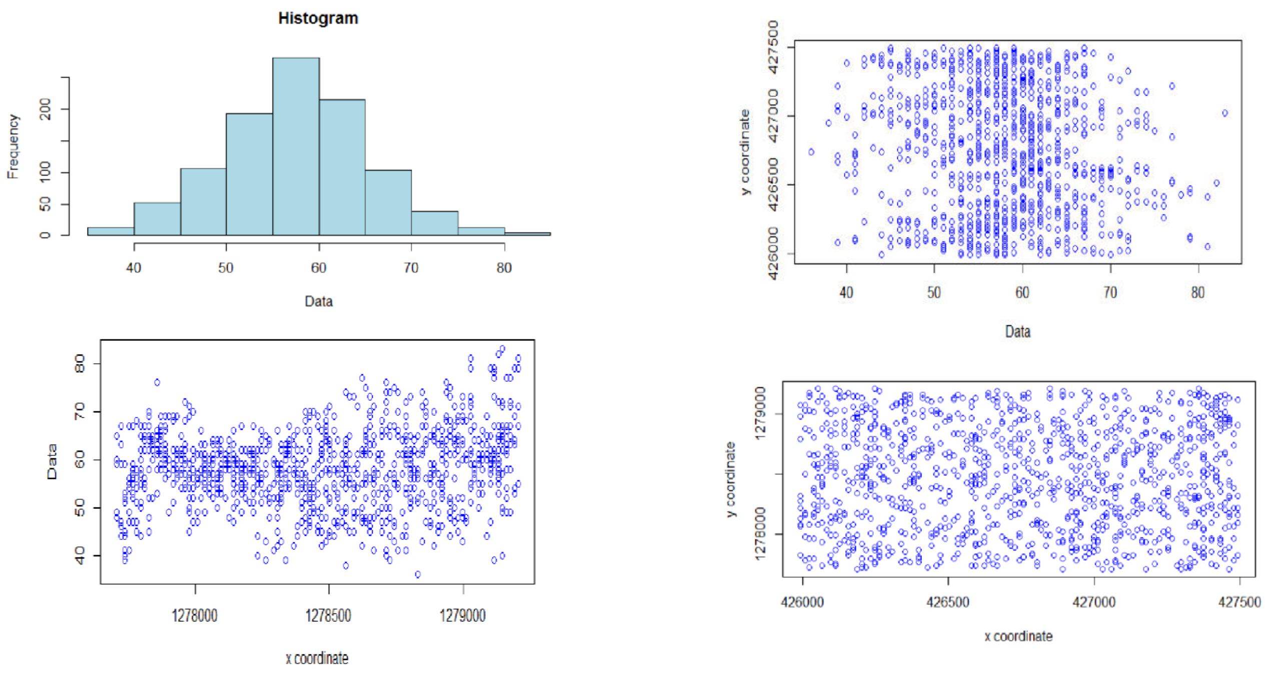


Figure 1.16: Scatter plots of the data

These spectral values of pixels are spatially auto-correlated, and their spatially dependent structures are represented by the variogram (Zhang et al., 2009). Variograms are computed and fitted with the exponential model (Figure 1:17). Since the semivariance does not increase continuously over the separation distance, it indicates there is no significant spatial trend in the variable. The range of the variogram is 65 pixels, the sill is 60, and the nugget is zero. Since there is no nugget effect, which indicates, the kriging predictions will go exactly through the observations. The kriging variance will be zero at the observations because there is no residual.

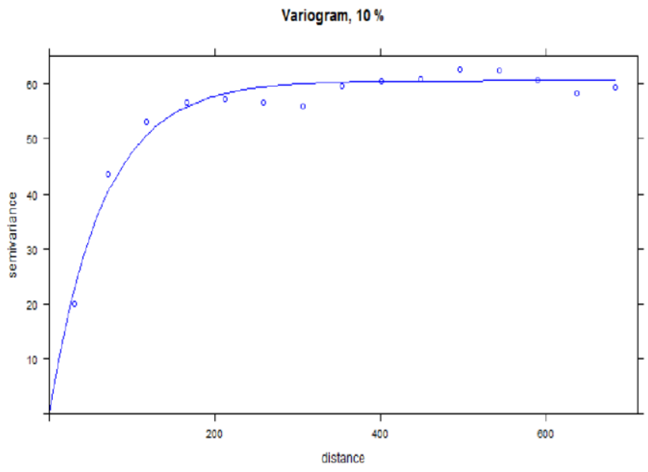


Fig 1.17: Variogram computed for the pixel values (DN) without a trend

Now, we have built a model of spatial dependence that can be used for prediction. We can use kriging as well as conditional simulations to predict at particular points. In Fig 1:18, we plot two realizations that are generated by the ordinary kriging and conditional simulation methods of Gstat. For each set of realizations in this section, we use set.seed so that the results presented here are reproducible.

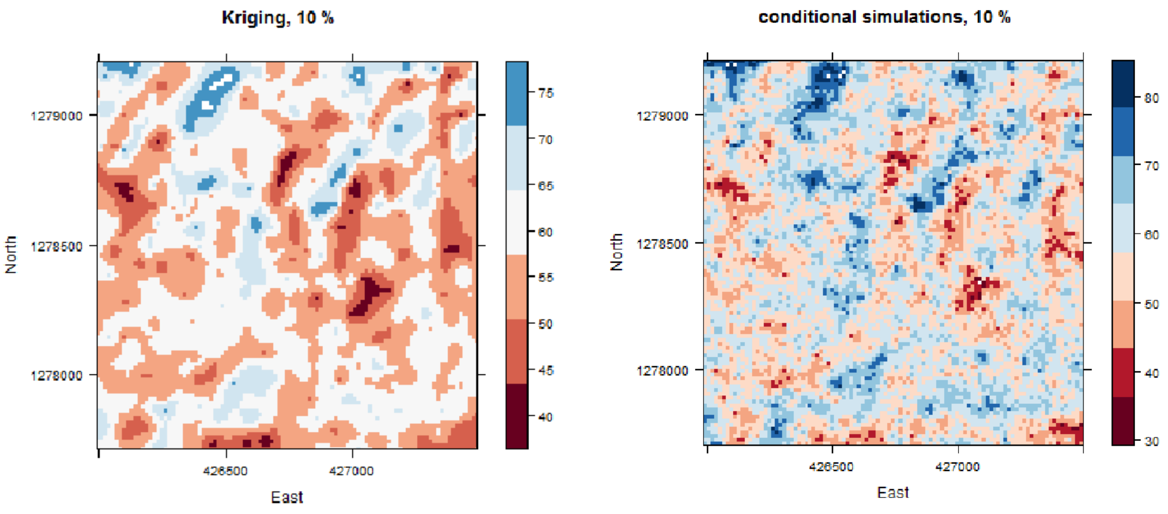


Fig 1.18: Realizations generated from kriging and conditional simulation

We see remarkable differences between the kriging and conditional simulation. We observe artificial smoothing in the kriging image, as it tends to overestimate the small DN values and underestimate the high DN values. A major weakness of the kriging map is that it does not account for spatial autocorrelation of errors, and thus does not provide a realistic picture (Evan, 1993). This property of kriging can be a severe problem when the resulting models are used for prediction. On the other hand, the conditional simulation realizations appear closer to the spatial structure of the original image, yet they are noisier. The conditional simulation algorithms provide realizations that honor the global structure specified by the variogram. However, a simulated value is not necessarily the best estimation in a statistical sense at a particular point (Lantuéjoul, 2002).

Remember that we used ten percent of the data i.e.1020 of the observations, the calibration dataset, to model the spatial structure using a variogram model. In addition, to predict the remaining 9180 points, where we also sampled data that is commonly referred to as the validation dataset. Since we do not use the 9180 points of the validation dataset either to make the model, these are an independent data set to test the model. We can compare the predictions with the actual values using numerical summary, histogram and the plot of the differences.

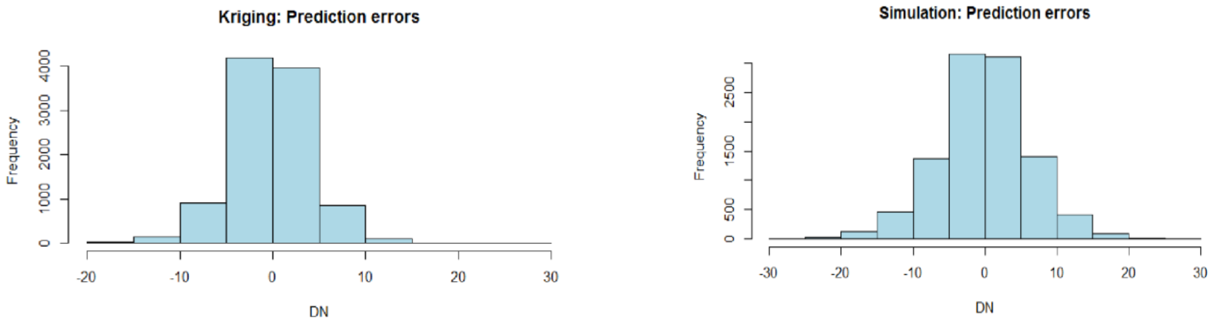


Fig 1:19 Histograms plots for kriging and conditional simulation (10%)

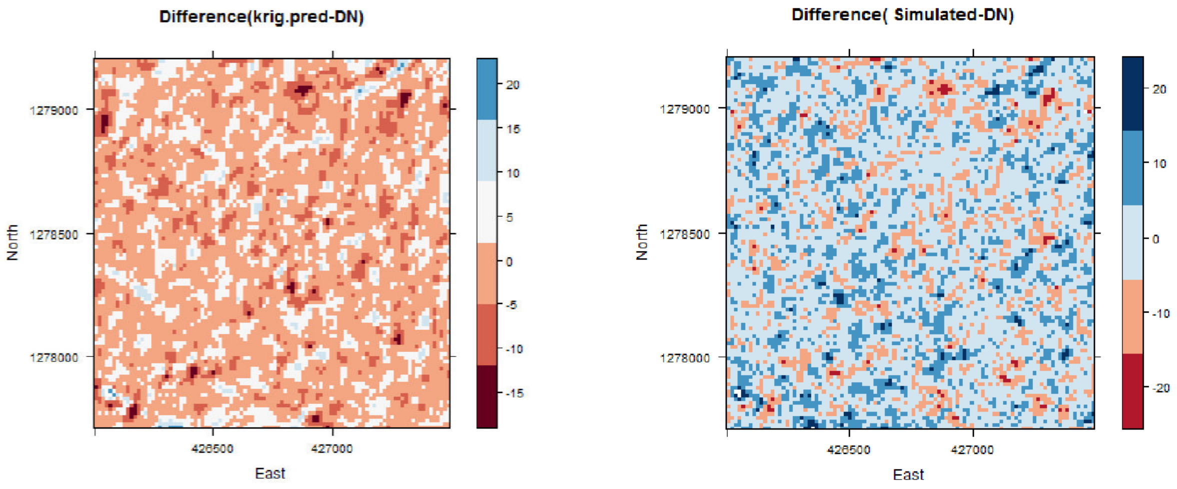


Fig 1.20: Spatial distribution of errors in cross-validation as represented by a color plot of the errors

The bias and the precision that are referred to as mean prediction error (MPE) and root mean square error (RMSE), respectively, are computed. The scatter plot in Fig.1.21 is made between the kriged vs. original and the simulation vs. original values.

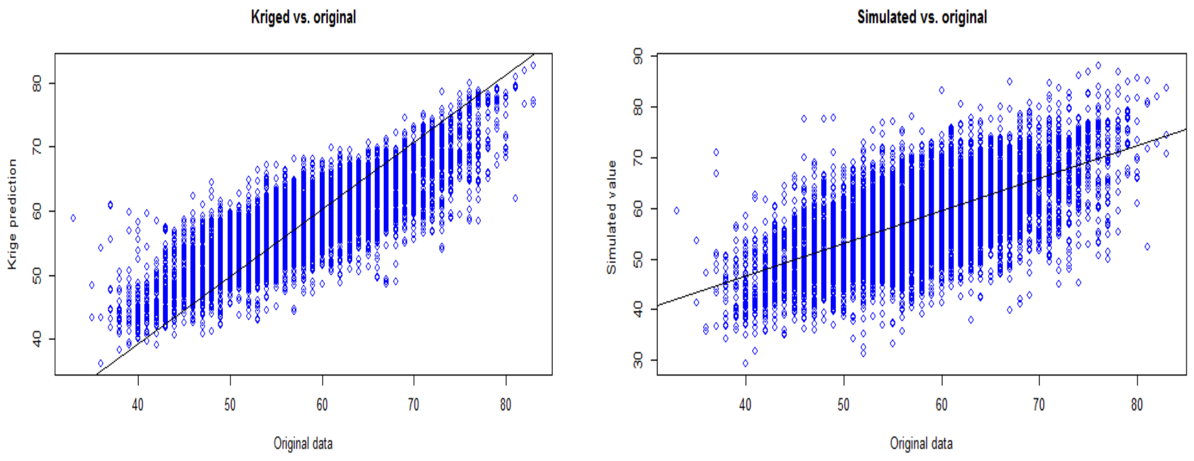


Fig 1.21: Scatter plot and regression line computed for the kriged vs. original and simulated vs. original values.

The MPE, RMSE and r values for kriging are -0.159, 4.151 and 0.83 whereas they are -0.110, 6.353 and 0.64 for the simulations. The kriging estimate appears to have an advantage over the simulation as it provides best estimate in a minimum error variance sense at each location locally regardless of estimates made at other places. On the other hand, sequential simulation algorithms are globally correct as they reproduce a global structured statistics. Finally, we combined, the image compression with a standard encryption algorithm AES as a post-processing step to obtain compressed and encrypted data.

4.4. Encryption of the random samples

We can perform simple file encryption in OpenSSL from the command line. All that we need is to remember a password used. So we encrypt the file that contains 80% percent of the samples by passing the following command: ‘openssl enc -aes-128-ctr –salt -in 80.csv -out 80.csv.enc’. It will then prompt for a password, for this study, the password we used is "BahirDar". Consequently, we create a binary file 80.csv.enc, in the same manner we did for the squeeze output we add the headers back to this file and visualize it as an image. To decrypt 80.csv.enc, the data recipient needs to remember the cipher and the password ‘openssl enc - d -salt -aes-128-ctr -in 80.csv.enc –out 80.csv.new’.

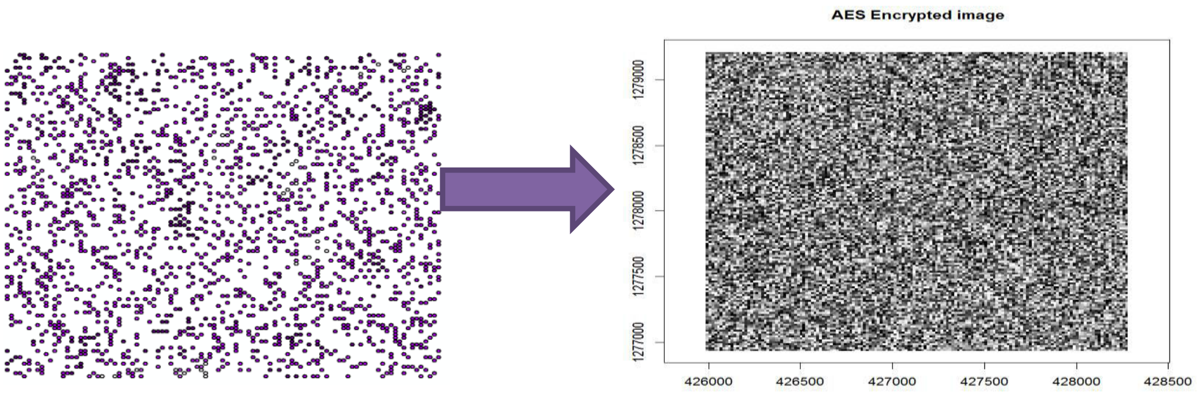


Fig1.22: Plots of the random samples and output of AES encryption

4.5. Realization from Sequential Gaussian Simulations

To investigate the effect of the sampling size, how the variogram fit and its parameters change with the sample size, the data set were randomly subsampled in R. The sub-samples were taken independently from each other. For each sub-sample, the locations were selected randomly and independently. To evaluate, we compare the simulations for different sizes of random samples. In particular we selected 30 %, 50 % and 80%. The figures shown below represent the outputs of the conditional simulations that resulted in increasing the sampled points.

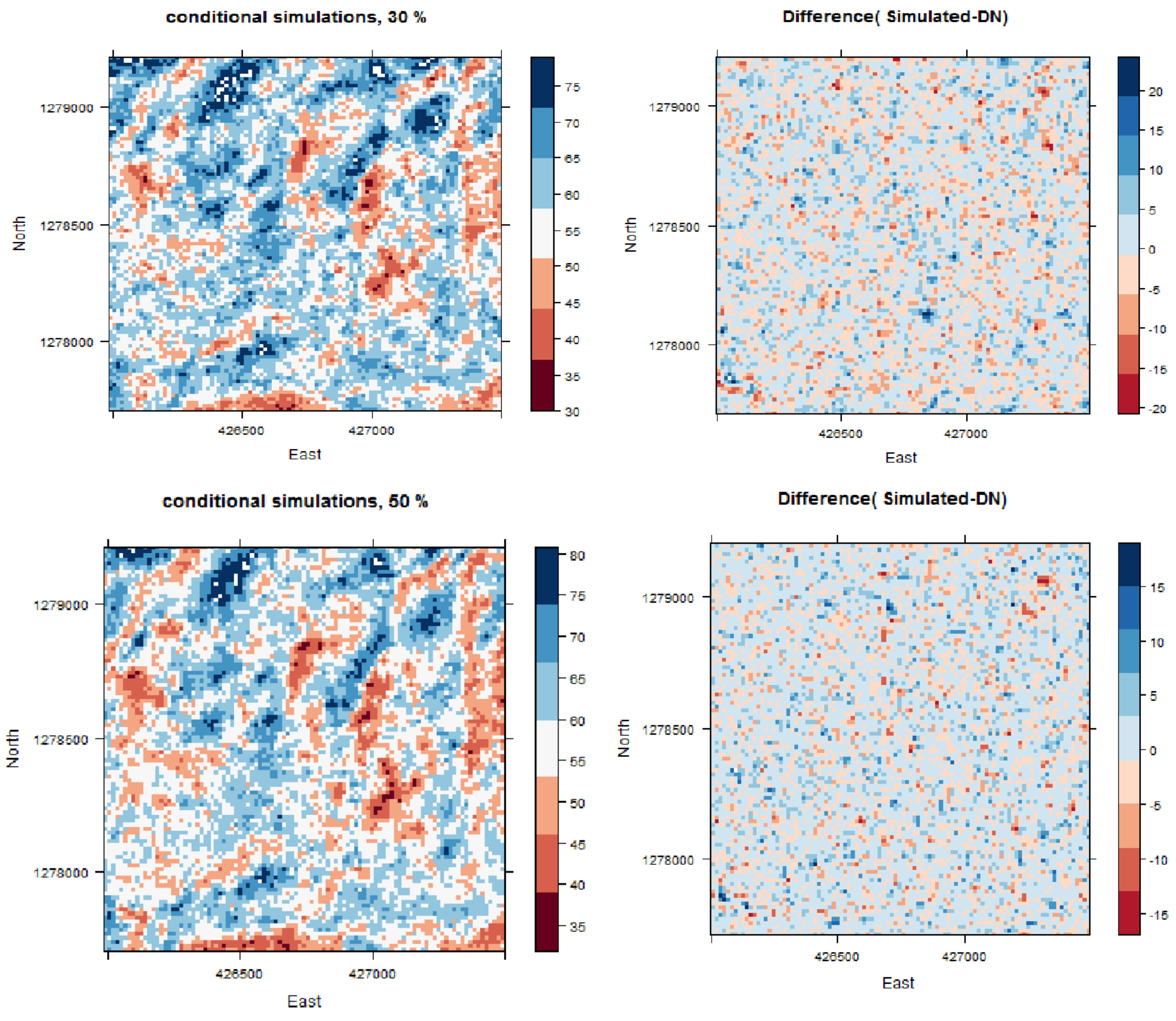

conditional simulations, 30 %
Difference(Simulated-DN)
conditional simulations, 50 %
Difference(Simulated-DN)
North
East
1279000
1278500
1278000
426500
427000
75
70
65
60
55
50
45
40
35
30
20
15
10
5
0
-5
-10
-15
-20
80

Fig 1.23: Conditional simulations and difference map for thirty and fifty percent.

These results of the simulation plots give an idea of the uncertainty of the estimates. Due also to the graphical information contained in the variogram models (Evan, 1993) all the simulated plots have textures. They are coarser but can be taken as continuous. A closer look at Fig 1.23 shows when the number of conditioning data increases the conditional simulations look more and more original image (Lantuéjoul, 2002). The mean error and root mean square, in Table 4.1, show they are gradually decreasing. Hence, to know how much further we compromise the compression ratio we compute realizations constructed by using 80% of the data. The histogram and normal qq plot summary analysis of the difference image show that the errors are distributed uniformly around zero.

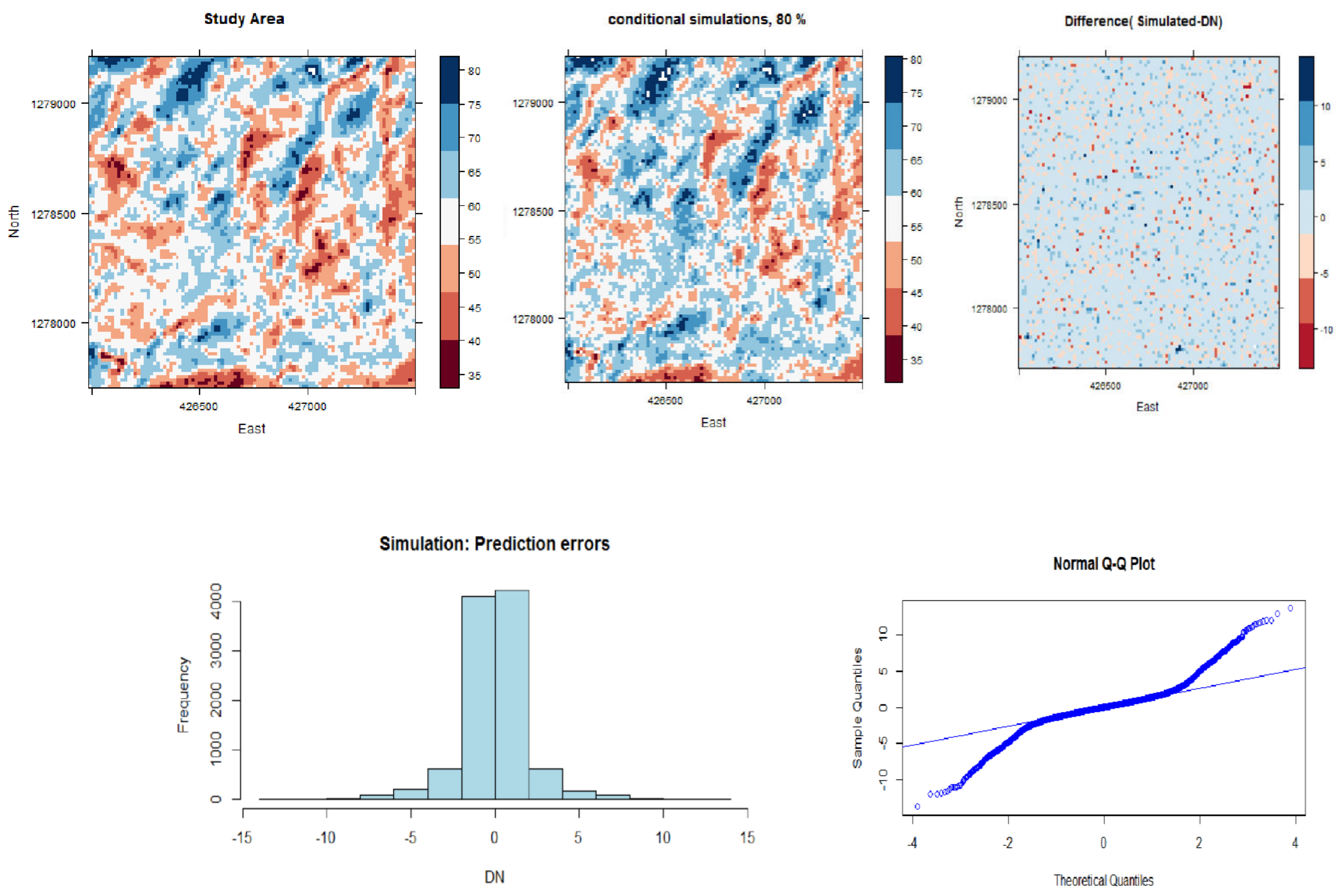


Fig 1.24: On the top, from left to right study area, simulation, the difference image, histogram, and normal qq-plot.

Samples Selected	SD	Minimum	25th	50th	Mean	75th	Maximum	*r*	MPE	RMSE
Original	7.60	33	53	58	58.09	63	83	--	--	---
10%	7.56	29.2	53.04	57.93	57.98	62.73	87.46	0.64	-0.110	6.353
20%	7.55	31.02	53.17	58.05	58.03	62.80	84.35	0.78	0.060	4.98
30%	7.701	29.97	53.08	58.01	58.06	63.06	85.35	0.84	-0.026	4.304
40%	7.66	28.79	53.17	58.13	58.11	62.9	87.27	0.88	0.023	3.732
50%	7.61	33.81	53.17	58.20	58.12	62.93	88.36	0.908	0.032	3.254
60%	7.69	34.89	53.21	58.13	58.11	62.83	85.73	0.93	0.015	2.82
70%	7.65	32.38	53.3	58.15	58.11	62.87	83.74	0.94	0.016	2.516
80%	7.62	31.47	53.22	58.2	58.1	62.79	83.96	0.96	0.012	2.118
90%	7.62	33.2	53.18	58.28	58.1	62.71	84.53	0.97	0.006	1.608

Table 9: Comparison of summary statistics for conditional simulations. Mean, standard deviation, minimum, 25th, 50th, 75th quantiles, maximum, correlation, mean prediction error and root mean square.

Samples Selected	Minimum	25th	50th	Mean	75th	Maximum
10%	-26.98	-3.79	-0.05	-0.11	3.68	27.18
20%	-21.22	-2.60	0.05	0.06	2.63	21.69
30%	-20.81	-2.06	-0.03	-0.02	1.94	25.76
40%	-18.2	-1.59	-0.01	0.02	1.60	22.08
50%	-17.02	-1.27	0.02	0.03	1.32	19.07
60%	-17.31	-1.09	-0.01	0.01	1.15	22.84
70%	-14.09	-1.01	-0.01	0.01	1.01	20.9
80%	-13.06	-0.91	-0.002	0.01	0.914	18.21
90%	-14.25	-0.78	0.009	0.006	0.833	13.56

Table 10: Summary statistics of the prediction errors caused by conditional simulations

From the summary statistics and cross-validation measurements, we are convinced that as the sample size increases the difference between the simulated realizations and the original image significantly decreases. The experiments reported here reveal that conditional simulation can be one means of performing lossy compression of satellite images.

4.6. Comparison based on compression ratio and speed

The compression ratio is defined as the compressed output size divided by the original uncompressed input size. A compression algorithm that compresses a 20MB file to 5MB has a compression ratio of 5/20 = 0.4, often represented as a ratio 1:4. A smaller number signifies better compression is achieved. The compression ratio results obtained in the paper (Kelley & Tamassia, 2014) reveal that squeeze can achieve high compression ratios compared to standard compressions. In this study, the squeeze cipher applied on the study area does not offer a significant compression ratio as shown in the figure 1.25 below; it shows that in some cases the compressed file even can grow larger than the original.

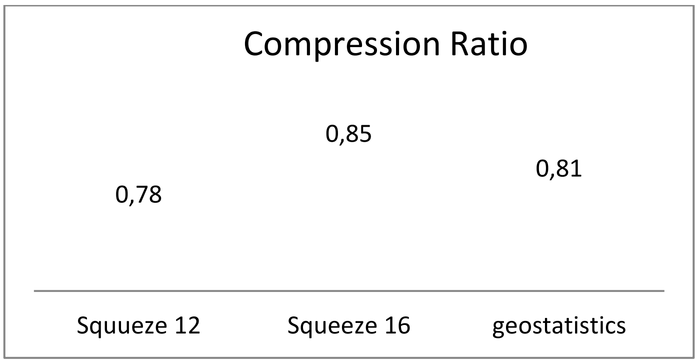


Figure 1.25: Compression ratio for the squeeze and geostatistics scheme

On the other hand, the geostatistical method is tunable, and the compression ratio depends on the size of the samples selected. The variogram model parameters together with the samples selected are encrypted using encryption standard (AES) CTR mode and will be transferred to the receiver at the other end. In this paper, we have sampled the study area on a constant basis from 10% to 80%. Hence, the compression ratio achieved by applying encryption on the 80% samples is 0.81 and as the previous sections show the image reconstructed does not lose too many details. The low compression ratio attributed to the squeeze cipher can be further studied by comparing it with a standard encryption performed on top of the LZW compression. For instance, LZW followed by AES gives a slightly better compression ratio than the squeeze scheme. However, be aware that though the geostatistical scheme achieves high compression ratios, in extreme cases it can cause substantial biases of the compressed image.

Further, in each case we recorded the elapsed compression and decompression CPU times. We consider the time taken for the compression and decompression separately. In the squeeze case the compression and decompression time the code prints out the time elapsed for compression and decompression. In the geostatistical scheme as it is a sequential compress then encrypt method we measure the compression/decompression and encryption/decryption time independently. However, note that this factor depends on the performance of the computers used and with the development of high-speed computers we get subtle values. The algorithms were tested on a 2.4 GHz core i7 computer with 8 GB RAM.

In R the proc.time command defines the time elapsed by the running R process and computes the real and CPU time in seconds. The real elapsed time that took to select 80% percent of the samples and writing a file that contains those samples is 0.14 sec. Then we encrypt this file using AES cipher. To measure the elapsed time when encrypting using OpenSSL we enter $time openssl enc -aes-128-ctr -salt -in 80.csv -out 80.csv.enc at the shell command and pass the password. We do these consecutively and on the average it took 0.008 s. To decrypt the file we pass the command $ time openssl enc -d -salt -aes-128-ctr -salt -in 80.csv.enc -out 80.csv.new and the password. On a similar approach the time it took to decrypt is 0.008 s, note that in AES CTR mode both encryption and decryption are fast as they can be parallelized. In Appendix C, we present the commands and their corresponding outputs. The figure below summarizes the time comparison between the two schemes.

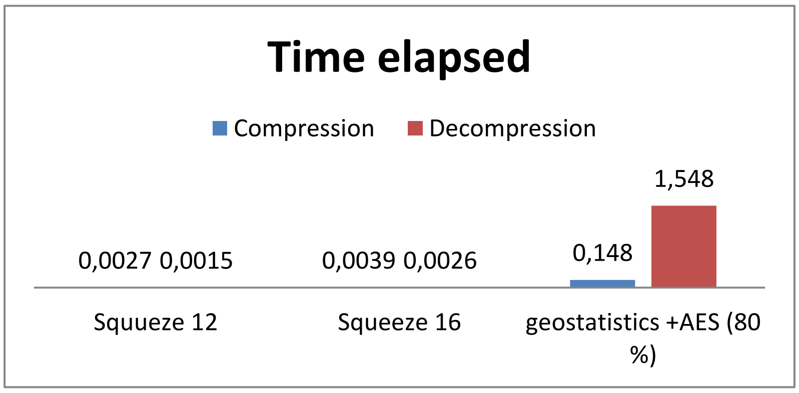


Figure 1.26: Time elapsed for the squeeze and geostatistics scheme

5. DISCUSSION

This study analyzed compression and decompression, encryption and decryption of satellite images. Compression and decompression are linked as the original satellite image remains unaltered after their successive application. Similarly, encryption and decryption are related as the orignal image can be recovered from encrypted file. The quality of a satellite image is to be maintained while reducing its volume. In this thesis, the squeeze compression and encryption, in brief, the squeeze scheme, is applied that was already available to texts. It is compared with image compression and encryption by geostatistical simulations, in short, the geostatistical scheme.

The study has shown that the squeeze scheme maintains an excellent quality of the image and that no loss of information occurs after recovering the image. The image reconstructed with the squeeze scheme is identical to the original image on a pixel-by-pixel basis. This can only be achieved for small compression ratios (0.8) because squeezing is initially formulated to reduce the one-dimensional correlation in texts. Adapting this algorithm to two-dimensional images, for this reason, causes several difficulties.

1. The squeeze scheme reads the image data by joining the rows of the image in a horizontal line and searching for repetitive strings along that line. Satellite images pixels, however, have a high spatial correlation between neighbor pixels. Thus, reading the file along a horizontal line by placing each row next to another row forces the algorithm to create dictionaries with a large lookup time and additional CPU cycles.
2. In addition, the squeeze scheme misses repetitive information that cannot be joined along a horizontal line. For this reason, the compression ratio is significantly low as compared to other standard image compression algorithms. Nevertheless, the results obtained by Kelley & Tamassia (2014) assure that due to the random permutation of the dictionary entries the squeeze cipher provides safety.

The geostatistical scheme is based on the assumption (in the field of image compression) that reads "small numerical errors in decoded values lead to small visual errors that are unrecognizable" (Salomon & Motta, 2010). This is the main idea when an image is reconstructed from a smaller data set. This however is not always true. The current study has shown that spatial prediction methods can produce results that are closer to the reference image. Notably, when the amount of conditioning data increases the simulated images will be unidentifiable from the original by the naked eye.

Overall, for large conditioning data sets the Gaussian conditional simulation methods provide smaller RMSE and MRE values and higher accuracy when assessed by means of the cross-validation methods. Conditional simulations, unlike kriging, however, are not standard methods for interpolations, and they rather preserve the general spatial variability of data.

The geostatistical scheme, for this reason, is a compression with loss of accuracy. Even though it offers high compression ratio, we might still choose the squeeze scheme if the quality of the reconstructed image is the most important issue to deal. We further note that a high compression ratio leads to a poor resulting image. The trade-off between compression ratio and the quality of the reconstructed images is an important factor worth considering when deciding upon an image compression schemes.

In geostatistical image compression and protection schemes, the image is first compressed. Then, the compressed image is entirely encrypted using a standard AES cipher. The AES encryption uses a password when encrypting the file and once the image file is encrypted it cannot be decrypted without using the correct password. In this scheme compression and encryption are completely disjoint processes. For big data like a satellite image the compression method used requires a considerable size of samples to recover a seemingly image, this will in turn increase the amount of data to be encrypted. This approach demands high computation and might not be desirable especially in reserved communications such as real-time networking and mobile communications which has limited computational power devices.

In principle, the senders of an image can apply any of the two schemes and thus protect the content of the image file upon sending. Both schemes compress an image to save space when transmitting and offer the best security currently available. The output file can be restored to the original image by the receiver who has the right password and the necessary application installed to run the spatial simulations.

In this paper, we have established a detailed comparison between the squeeze scheme and the geostatistical scheme. We managed to analyze the outcomes for one satellite image. The study further provided a toy example that gives the basics of the LZW compression and its modification towards squeeze ciphers. The comparison mainly focuses on the compression ratio that determines the quality of the restored image.

In the future, we also need to consider different kinds of satellite images such as multispectral images, hyperspectral images and test the reliability of the squeeze, as well as the geostatistical scheme. In addition, different geostatistical simulation methods like turning bands, simulated annealing should be established and performed to support the claim that spatial simulations gave alternative approach for image compression.
For users of big data such as satellite images choices have to be made on the level of compression ratio, type of compression, security requirements we entail for our applications. The security standards for both schemes are safe to be used for practical use and, in theory, it is hard to assume it will be possible to crack them. The main difference between the two schemes lies in the compression ratio and quality of the reconstructed image. The squeeze cipher compression depends on the nature of the data, and the compression ratio is not tunable to a level that we want but yet it is a reversible and secure scheme.

The geostatistical scheme however is a lossy compression technique that is tunable to the level required by the user and is safe as it uses the standard encryption algorithm. The primary weakness lies in its inability to replicate the reference image.
Image compression and encryption are a growing field. The study shows that currently available methods can further be implemented and analyzed for different types of images. In addition, it opens door to use object base analysis in compressing and encrypting images.

6. CONCLUSION AND RECOMMENDATIONS

This section presents the conclusions and recommendations through applying the methodologies explained in the previous chapter. In the first part we present the conclusions and answers to the research questions and in the second part recommendations are discussed.

6.1. Conclusions

The purpose of this thesis is to compare the squeeze scheme with the geostatistical scheme based encryption and encryption. We will address here the answers we found to the research questions.

- ✓ Should S1 be adapted to encrypt and compress satellite images?

The statistical tests showed that a modification of the LZW, the squeeze cipher, provides safety from attacks during communication for satellite images. It removes spatial redundancy without adding artifacts after decoding the image. Though the compression ratio found from the squeeze cipher is significantly lower than from the LZW algorithm, the findings of this study show that adjusting and using the squeeze cipher and perform secure compression instead of compression alone is preferable.

- ✓ How can we minimize data loss during image compression in geostatistical scheme?

From the results presented in Tables 9 and 10, we observe that the magnitude of the difference between the original image and restored image gradually decreases as we increase the number of samples and hence that the size of the samples is the primary factor to minimize the data loss in the geostatistical scheme. Thus, the data loss is reduced by adding the conditioning data and fit those with the appropriate variogram model. This study thus has shown that conditional simulation can be a suited routine for uncertainty analysis.

- ✓ What degree of compression gives an acceptable result in geostatistical scheme?

The statistical measures such as RMSE, ME, and correlation coefficient that are computed on a pixel basis for the different realizations show that we get higher accuracy if the conditioning data is significantly large. A comparable compression ratio to the squeeze scheme is obtained when we choose samples in the range of 75% to 80%. Thus, a compression ratio calculated from realizations generated from such datasets provides higher values. As we presented in Table 9, we can reasonably choose the size of conditioning data we want and reconstruct an image. For instance, if we choose 10% of the samples the reconstructed image will be noisier due to the high difference on a pixel level. We can increase the number of samples and continue the process until we are happy with the reconstructed image. Thus, it depends on the requirement of the particular user and task at hand.

- ✓ How to examine the efficiency of both squeeze scheme and geostatistical scheme?

The effectiveness of both schemes is explored on the basis of compression ratio and time elapsed. The results on Figure 1.25 and Figure 1.26 show that the squeeze scheme has achieved lowest compression ratio in relatively small time to encode and decode images (0.78 compression ratio in 0.0027 sec), whereas the geostatistical scheme is slow and is not reversible, for 80% of the samples we obtain (0.81 compression ratio in 0.148 sec).

- ✓ How to use additional information on the input image to improve the compression accuracy?

Unfortunately, we are unable to extend the study to include application to classified images and further study the object basis analysis. We recommend those alternative approaches for future study.

- ✓ Should image compression and encryption be done in one go? Alternatively, independently?

Compression alone is not sufficient as it can be accessed. But systems like squeeze cipher where security is embedded into the compression algorithm provide safety and thus can meet the demand of fast and secure transmission of data. But as Kelley & Tamassia (2014) pointed out combining compression and encryption in a naïve way has drawbacks. As the compression leak information about the length of the input.

6.2. Recommendations

This study works on a pixel basis analysis. One should note that neighboring pixels of satellite images are spatially auto-correlated. Hence, we can group these pixels based on some criteria and perform the operation at the object level. In general, object-based analysis starts with grouping pixels into meaningful objects through image segmentation techniques. Consequently, recently different image analysis like a satellite image classification has been on the object analysis. For the future, we recommend extending the squeeze algorithms to work at the object level. One way could be by integrating the algorithm to the simplest compression algorithms such as quadtree decomposition and run length encoding. One will recall that quadtree decomposition subdivides an image into homogeneous blocks. First, it divides a square image into four blocks and tests whether each block meets some criterion of homogeneity, a block that satisfies this criterion will not be divided further. However, if it does not, it will be split again into four blocks, and test each of these blocks and continue the process until each block satisfies the criterion. Previous works show that quadtrees play a vital role in compression of images and can work hand in hand with different compression techniques such as fractal compression. In contrast, run length encoding is a simple form of data compression in which consecutive elements are replaced by a counter showing how many times some data is repeated. If we use squeeze cipher as a post-processing step after run-length encoding may be a better compression as well fast communication could be achieved.

Further, if we can devise a better way of joining the rows of an image high compression ratio and small compression time could be achieved. Recall that, we form the squeeze dictionary by placing one row of the image next to the other, an alternative could be to join them in a reverse order i.e. joining the last elements of two rows of an image, such approaches could be further studied to improve the squeeze scheme.

About the geostatistical scheme, note that sequential simulation algorithms are based on a variogram model, and hence are limited to the reproduction of two-point statistics. Thus, they fail to reproduce complex structures. This triggers the concept and the possible need for object-based conditional simulations. In addition, in recent times, multiple-point statistics have been applied to the field of geology that is established to honor large amounts of local data. The variogram was the building block to read the spatial variability of the data set but in case of multiple point statistics all information is drawn from a training image. In addition, applying better sampling approaches, we can reduce the prediction errors. In the future, we recommend a better sampling strategy, object-based conditional simulations, multiple points statistics and analysis from the security point of view have to be considered during satellite image compression and decompression procedures.

LIST OF REFERENCES

Aldossari, M., Alfalou, A., & Brosseau, C. (2014). Simultaneous compression and encryption of closely resembling images: application to video sequences and polarimetric images. *Optics Express*, *22*(19), 22349–68. doi:10.1364/OE.22.022349

Allard, D., Froidevaux, R., & Biver, P. (2007). Conditional Simulation of Multi-Type Non Stationary Markov Object Models Respecting Specified Proportions. *Mathematical Geology*, *38*(8), 959–986. doi:10.1007/s11004-006-9057-5

Arpat, G. B. (2005). *Sequential simulation with patterns*. Stanford University. Retrieved from https://pangea.stanford.edu/ERE/pdf/pereports/PhD/Arpat05.pdf

Arpat, G. B., & Caers, J. (2007). Conditional Simulation with Patterns. *Mathematical Geology*, *39*(2), 177–203. doi:10.1007/s11004-006-9075-3

Blackstock, S. (2001). LZW and GIF explained. Retrieved October 15, 2014, from http://gingko.homeip.net/docs/file_formats/lzwgif.html

Cheng, H. (2000). Partial encryption of compressed images and videos. *IEEE Transactions on Signal Processing*, *48*(8), 2439–2451. doi:10.1109/78.852023

Cressie, N. A. C. (1993). *Statistics for spatial data*. J. Wiley. Retrieved from http://books.google.nl/books?id=4SdRAAAAMAAJ

Daemen, J., & Rijmen, V. (2002). *The Design of Rijndael: AES - The Advanced Encryption Standard* (p. 238). Springer Science & Business Media. Retrieved from http://books.google.com/books?hl=en&lr=&id=tfjd6icCUoYC&pgis=1

Delbari, M., Afrasiab, P., & Loiskandl, W. (2009). Using sequential Gaussian simulation to assess the field-scale spatial uncertainty of soil water content. *CATENA*, *79*(2), 163–169. doi:10.1016/j.catena.2009.08.001

Deutsch, C. V., & Journel, A. G. (1998). *GSLIB: geostatistical software library and user's guide* (2nd ed., p. 384). Oxford University Press, New York.

Emery, X. (2008). Statistical tests for validating geostatistical simulation algorithms. *Computers & Geosciences*, *34*(11), 1610–1620. doi:10.1016/j.cageo.2007.12.012

Emery, X., & Lantuéjoul, C. (2006). TBSIM: A computer program for conditional simulation of three-dimensional Gaussian random fields via the turning bands method. *Computers & Geosciences*, *32*(10), 1615–1628. doi:10.1016/j.cageo.2006.03.001

Evan, E. (1993). Spatial simulation: Environmental Applications. *Environmental Modelling with GIS, M.F. Goodchild et Al., Eds., Oxford University Press*, 423–437. Retrieved from http://epa.gov/esd/cmb/research/papers/ee104.pdf

Fischer, V., & Bernard, F. (2011). *Security Trends for FPGAS*. Dordrecht: Springer Netherlands. doi:10.1007/978-94-007-1338-3

Ghosh, J. K., & Singh, A. (2009). Fractal compression of satellite images. *Journal of the Indian Society of Remote Sensing*, *36*(4), 299–311. doi:10.1007/s12524-008-0031-1

Goldreich, O. (2007). *Foundations of Cryptography: Basic Tools* (p. 380). Cambridge University Press, New York. Retrieved from http://www.amazon.com/Foundations-Cryptography-Volume-Basic-Tools/dp/0521035368

Heron, S. (2009). Advanced Encryption Standard (AES). *Network Security*, *2009*(12), 8–12. doi:10.1016/S1353-4858(10)70006-4

Hu, Y.-C., & Chang, C.-C. (2000). A new lossless compression scheme based on Huffman coding scheme for image compression. *Signal Processing: Image Communication*, *16*(4), 367–372. doi:10.1016/S0923-5965(99)00064-8

J., B., Carson, J., Nelson, B., Nicol, D., & Banks, J. (2001). *Discrete-Event System Simulation.* Prentice Hall.

Jin, J. (2012). An image encryption based on elementary cellular automata. *Optics and Lasers in Engineering*, *50*(12), 1836–1843. doi:10.1016/j.optlaseng.2012.06.002

Journel, A. G. (2008). Geostatistics for Conditional Simulation of Ore Bodies. *Economic Geology*, *69*(5), 673–687. doi:10.2113/gsecongeo.69.5.673

Kak, A. (2014). AES: Lecture Notes on "Computer and Network Security." *Purdue University*. Retrieved January 24, 2015, from https://www.scribd.com/doc/251641555/Aes

Katz, J., & Lindell, Y. (2007). *Introduction to Modern Cryptography: Principles and Protocols* (p. 552). Chapman and Hall/CRC; 1 edition. Retrieved from http://www.amazon.com/Introduction-Modern-Cryptography-Principles-Protocols/dp/1584885513

Kelley, J., & Tamassia, R. (2014). Secure Compression: Theory & Practice. *Dept. of Computer Science, Brown University*. Retrieved from http://eprint.iacr.org/2014/113.pdf

Khan, M., & Shah, T. (2014). A Literature Review on Image Encryption Techniques. *3D Research*, *5*(4), 29. doi:10.1007/s13319-014-0029-0

LAKHANI, G. (2006). Reducing coding redundancy in LZW. *Information Sciences*, *176*(10), 1417–1434. doi:10.1016/j.ins.2005.03.007

Langiu, A. (2013). On parsing optimality for dictionary-based text compression—the Zip case. *Journal of Discrete Algorithms*, *20*, 65–70. doi:10.1016/j.jda.2013.04.001

Lantuéjoul, C. (2002). *Geostatistical Simulation: Models and Algorithms* (p. 256). Springer Science & Business Media. Retrieved from http://books.google.com/books?id=4FxPoZ4t898C&pgis=1

Menzes, A. J., Oorschoot, P. C. V., & Vanstone, S. (1996). *Handbook of Applied Cryptography* (p. 780). CRC Press; 1 edition. Retrieved from http://www.amazon.com/Handbook-Cryptography-Discrete-Mathematics-Applications/dp/0849385237

Muhaya, B., & T., F. (2011). Chaotic and AES cryptosystem for satellite imagery. *Telecommunication Systems*, *52*(2), 573–581. doi:10.1007/s11235-011-9462-z

Mukesh, S., & Smiley, G. (2012). Compression and Encryption: An Integrated Approach, *Vol.1 - Is*(Vol.1 - Issue 5 (July - 2012)). Retrieved from http://www.ijert.org/view.php?id=446&title=compression-and-encryption-an-integrated-approach

Munish, K., & Anshul, A. (2014). Research Publish Journals. *International Journal of Computer Science and Information Technology Research*, *2*(2), 77–81. Retrieved from http://www.researchpublish.com/journal/IJCSITR/Issue-2-April-2014-June-2014/45

Nandi, U., & Mandal, J. K. (2013). Modified Compression Techniques Based on Optimality of LZW Code (MOLZW). *Procedia Technology*, *10*, 949–956. doi:10.1016/j.protcy.2013.12.442

Nicolas, R., Alexandre, B., & Jianbing, W. (2009). *Applied Geostatistics with SGeMS A User's Guide* (p. 286). Cambridge University Press, New York.

Paar, C., Pelzel, J., & Preneel, B. (2011). *Understanding Cryptography: A Textbook for Students and Practitioners [Hardcover]* (p. 372). Springer; 1st ed. 2010 edition. Retrieved from http://www.amazon.com/Understanding-Cryptography-Textbook-Students-Practitioners/dp/3642041000/ref=pd_sim_sbs_b_1?ie=UTF8&refRID=1YF1YZMJBM45HCZ316D6

Panduranga, H. T., & NaveenKumar, S. K. (2013). Selective image encryption for Medical and Satellite Images. *International Journal of Engineering and Technology (IJET)*, *5*(1), 115–121.

Parziale, L., Barney, J., Cross, V., Johnston, W., Kienetz, E., Marins, E., … Redbooks, I. (2013). *Security for Linux on System z* (p. 348). IBM Redbooks. Retrieved from https://books.google.com/books?id=bFjAAgAAQBAJ&pgis=1

Pebesma, E. J. (2004). Multivariable geostatistics in S: the gstat package. *Computers & Geosciences*, *30*(7), 683–691. doi:10.1016/j.cageo.2004.03.012

PetroWiki. (2014). Geostatistical conditional simulation -. Retrieved September 28, 2014, from http://petrowiki.org/Geostatistical_conditional_simulation

Pyrcz, M. J., & Deutsch, C. V. (2014). *Geostatistical Reservoir Modeling* (p. 433). Oxford University Press. Retrieved from https://books.google.com/books?id=wNhBAgAAQBAJ&pgis=1

R.Gebbers, & Bruin, S. d. (2010). *Geostatistical Applications for Precision Agriculture* (p. 300). Springer Science & Business Media.

Remy, N., Boucher, A., & Wu, J. (2009). *Applied Geostatistics with SGeMS: A User's Guide* (p. 264). Cambridge University Press. Retrieved from https://books.google.com/books?id=AFlegl0OLacC&pgis=1

Robinson, T. P., & Metternicht, G. (2006). Testing the performance of spatial interpolation techniques for mapping soil properties. *Computers and Electronics in Agriculture*, *50*(2), 97–108. doi:10.1016/j.compag.2005.07.003

Seong, S.-W., & Mishra, P. (2006). A bitmask-based code compression technique for embedded systems. In *Proceedings of Conference on Computer-Aided Design* (pp. 251–254). New York (ACM).

Sha, L. (2008). *Encyclopedia of GIS*. (S. Shekhar & H. Xiong, Eds.) (pp. 472–475). Boston, MA: Springer US. doi:10.1007/978-0-387-35973-1

Sheldon, T. (2001). Compression Techniques (Linktionary term). Retrieved November 05, 2014, from http://www.linktionary.com/c/compression.html

Shine, J. A. (2001). Compression And Analysis of Very Large Imagery Data Sets Using Spatial Statistics. Retrieved from http://www.interfacesymposia.org/I01/I2001Proceedings/JShine/JShine-Paper.pdf

Socek, D., Magliveras, S., Ćulibrk, D., Marques, O., Kalva, H., & Furht, B. (2007). Digital Video Encryption Algorithms Based on Correlation-Preserving Permutations. *EURASIP Journal on Information Security*, *2007*(1), 1–15. doi:10.1155/2007/52965

Syed, A. R. S. (2003). Geostatistics - Frequently Asked Questions. Retrieved from http://www.ai-geostats.org/pub/AI_GEOSTATS/AI_GEOSTATSFAQ/FAQ_Geostatistics_01.pdf

TechTarget. (2014). WhatIs.com. Retrieved October 19, 2014, from http://whatis.techtarget.com/definition/initialization-vector-IV

ur Rehman, A., Liao, X., Kulsoom, A., & Abbas, S. A. (2014). Selective encryption for gray images based on chaos and DNA complementary rules. *Multimedia Tools and Applications*. doi:10.1007/s11042-013-1828-7

Usama, M., Khan, M. K., Alghathbar, K., & Lee, C. (2010). Chaos-based secure satellite imagery cryptosystem. *Computers & Mathematics with Applications*, *60*(2), 326–337. doi:10.1016/j.camwa.2009.12.033

Van Tilborg, H. C. A., & Jajodia, S. (Eds.). (2011). *Encyclopedia of Cryptography and Security*. Boston, MA: Springer US. doi:10.1007/978-1-4419-5906-5

Vanloocke, S. (2012). *Factorial Kriging of Soil Sensor Images using ISATIS*. Universiteit Gent. Retrieved from http://lib.ugent.be/fulltxt/RUG01/001/892/544/RUG01-001892544_2012_0001_AC.pdf

Wadi, S. M., & Zainal, N. (2013). Rapid Encryption Method based on AES Algorithm for Grey Scale HD Image Encryption. *Procedia Technology*, *11*, 51–56. doi:10.1016/j.protcy.2013.12.161

Wadi, S. M., & Zainal, N. (2014). High Definition Image Encryption Algorithm Based on AES Modification. *Wireless Personal Communications*, 1–19. doi:10.1007/s11277-014-1888-7

Wang, J.-F., Stein, A., Gao, B.-B., & Ge, Y. (2012). A review of spatial sampling. *Spatial Statistics*, *2*, 1–14. doi:10.1016/j.spasta.2012.08.001

Welch, T. A. (1984). A Technique for High Performance Data Compression [Terry A. Welch]. *IEEE Computer*, *17*(6), 8–19. Retrieved from http://dret.net/biblio/reference/wel84

Westlund, H. B. (2002). NIST reports measurable success of Advanced Encryption Standard. (News Briefs). - Free Online Library. *Journal of Research of the National Institute of Standards and Technology*. . Retrieved from http://www.thefreelibrary.com/NIST+reports+measurable+success+of+Advanced+Encryption+Standard....-a090984479

Xiang, T., Qu, J., & Xiao, D. (2014). Joint SPIHT compression and selective encryption. *Applied Soft Computing*, *21*, 159–170. doi:10.1016/j.asoc.2014.03.009

Xie, D., & Kuo, C.-C. J. (2005). Secure Lempel-Ziv compression with embedded encryption. In E. J. Delp III & P. W. Wong (Eds.), *Electronic Imaging 2005* (pp. 318–327). International Society for Optics and Photonics. doi:10.1117/12.590665

Yuen, C.-H., & Wong, K.-W. (2011). A chaos-based joint image compression and encryption scheme using DCT and SHA-1. *Applied Soft Computing*, *11*(8), 5092–5098. doi:10.1016/j.asoc.2011.05.050

Zhang, H., Lan, Y., Lacey, R. E., Huang, Y., Hoffmann, W. C., Martin, D., & Bora, G. C. (2009). Analysis of variograms with various sample sizes from a multispectral image. *International Journal of Agricultural and Biological Engineering*, *2*(4), 62–69. doi:10.3965/ijabe.v2i4.201

Zhou, J., Au, O. C., Fan, X., & Wong, P. H.-W. (2008). Secure LEMPEL-ZIV-WELCH (LZW) algorithm with random dictionary insertion and permutation. In *Multimedia and Expo, 2008 IEEE International Conference* (pp. 245 – 248). Hannover,GERMANY: IEEE. doi:10.1109/ICME.2008.4607417

Zhou, N., Zhang, A., Wu, J., Pei, D., & Yang, Y. (2014). Novel hybrid image compression–encryption algorithm based on compressive sensing. *Optik - International Journal for Light and Electron Optics*. doi:10.1016/j.ijleo.2014.06.054

Zhu, H., Zhao, C., & Zhang, X. (2013). A novel image encryption–compression scheme using hyper-chaos and Chinese remainder theorem. *Signal Processing: Image Communication*, *28*(6), 670–680. doi:10.1016/j.image.2013.02.004

Ziv, J., & Lempel, A. (1978). Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, *24*(5), 530–536. doi:10.1109/TIT.1978.1055934

APPENDIX A

I. **Squeeze Cipher Encryption algorithm** (Kelley & Tamassia, 2014)

```
Input: key $k_{prp}$, input character stream $I$

Output: encrypted output stream $O$

 InitSqueeze($k_{prp}$, $iv$, $G$, $prefix$, $c$, $T$)

while there is more input in $I$ do ----------------------▻ Start compressing and encrypting stream $I$

Read next character $c$

if $prefix \,||\, c$ in $T$ then

$prefix \leftarrow prefix \,||\, c$

else

Output index $i$ of prefix in $T$

RandomSwap($i, T$) ------------------------------------------- ▻ swap prefix with a random entry in $T$

RandomInsert($prefix \,||\, c, T$) ------------▻ insert $prefix \,||\, c$ randomly into empty entry in  $T$

 $prefix \leftarrow c$

if $prefix \neq \Lambda$ then ---------------------------------------------- ▻ Make sure we get any irregular input

Output index $i$ of prefix in $T$
```

II. **Squeeze Random Insert and InitSqueeze Functions**

```
function InitSqueeze ($k_{prp}$, $iv$, $G$, $p$, $c$, $T$)

Generate fresh $iv$

Compute $f_{k_{prp}}(iv) = seed$

Initialize $G$ with seed

$p \longleftarrow \Lambda, c \longleftarrow \Lambda$

Initialize table $T$

for each single character string $s$ do

RandomInsert ($s, T$)
```

```
**function** RandomInsert ($s, T$)
    Choose a random index $r$ in $T$
    While $T[r] \neq \Lambda$ do
        Choose a new random index $r$ in $T$
    $T[r] \longleftarrow s$
**function** RandomSwap ($i, T$)
    Choose a random index $r$ in $T$
    Swap $T[i]$ and $T[r]$
```

III. **Squeeze Cipher Decryption Algorithm** (Kelley & Tamassia, 2014)

```
**Input:** key $k_{prp}$, input stream of indices $I$
**Output:** output character stream $O$
Read $iv$ from $I$ --------------------------------------▷ Initialize PRG $G$, dictionary $T$, and variables $prev$ and $c$
InitializeSqueeze($k_{prp}$, $iv$, $G, T, prev$, $c, m$)
**while** there is more input in $I$ **do**----------------------------▷ Start decompressing and decrypting stream $I$
    Read next index $i$
    Compute the pseudo-random index $r$ of the new entry---------------▷ i.e., where the new $prev \,||\, c$ will be
    **if** $T[i]$ is undefined and ($i \neq r$ or $prev = \Lambda$) **then**
        Output $\perp$ and fail-------------------------------------------------------------------▷ decoding failed: invalid index
    ------------------------------------------------------------▷Skip these two tests on the first iteration
    **if** $T[i]$ is defined and $i \neq r$ and $prev = \Lambda$ then-----------------------▷ $i$ is valid and not part of a collision
        $c$ ←head ($T[i]$)
        RandomInsert ($prev \,||\, c, T$)
    **else if** $i = r$ and $prev \neq \Lambda$ then---------------------▷ particular case: original input was $prev \,||\, prev \,||\, c$
        $c$ ←head($prev$)
        RandomInsert($prev \,||\, c, T$)---------------------▷ We're decoding $prev \,||\, c$ and it will be in position $r = i$
```

```
Output $T[i]$ -----------------------------------------------------------------▻ Finally: output $T[i]$ and update $prev$
$prev \leftarrow T[i]$
RandomSwap$(i, T)$----------------------------------------------------------- ▻ Move $prev$ to a random position
```

APPENDIX B

The squeeze file generates in our case an ouput 100.tif.sec-12, depending the size of the dictionary used.

Step1: read the gdal info of the tiff using the following code by : gdalinfo 100.tif

Step 2: gdal_translate -of ENVI 100.tif 100.envi

Step 3: Create a file 100.tif.sec-12.hdr where the haders of these from 100.hdr file, where the samples and lines of this file are modified to the size of the encrypted ouput.

```
ENVI
description = {
100.tif.sec-12}
samples = 100
lines  = 98
bands  = 1
header offset = 0
file type = ENVI Standard
data type = 1
interleave = bsq
byte order = 0
map info = {UTM, 1, 1, 425985, 1279215, 15, 15, 37, North,WGS-84}
coordinate system string = {PROJCS["WGS_1984_UTM_Zone_37N",GEOGCS["GCS_WGS_1984",DATUM["D_WGS_1984",SPHEROID["WGS_1984",6378137,298.257223563]],PRIMEM["Greenwich",0],UNIT["Degree",0.017453292519943295]],PROJECTION["Transverse_Mercator"],PARAMETER["latitude_of_origin",0],PARAMETER["central_meridian",39],PARAMETER["scale_factor",0.9996],PARAMETER["false_easting",500000],PARAMETER["false_northing",0],UNIT["meters",1]]}
```

```
band names = {

Layer_1}
```

Step 4: gdalinfo 100.tif.sec-12. It outputs:

```
Driver: ENVI/ENVI .hdr Labelled

Files: 100.tif.sec-12

100.tif.sec-12.hdr

Size is 100, 98

Coordinate System is:

PROJCS["WGS_1984_UTM_Zone_37N",

GEOGCS["GCS_WGS_1984",

DATUM["WGS_1984",

SPHEROID["WGS_84",6378137,298.257223563]],

PRIMEM["Greenwich",0],

UNIT["Degree",0.017453292519943295]],

PROJECTION["Transverse_Mercator"],

PARAMETER["latitude_of_origin",0],

PARAMETER["central_meridian",39],

PARAMETER["scale_factor",0.9996],

PARAMETER["false_easting",500000],

PARAMETER["false_northing",0],

UNIT["meters",1]]

Origin = (425985.000000000000000,1279215.000000000000000)

Pixel Size = (15.000000000000000,-15.000000000000000)

Metadata:
```

```
Band_1=Layer_1

Image Structure Metadata:

INTERLEAVE=BAND

Corner Coordinates:

Upper Left  (  425985.000, 1279215.000) ( 38d19'16.15"E, 11d34'15.95"N)

Lower Left  (  425985.000, 1277745.000) ( 38d19'16.27"E, 11d33'28.09"N)

Upper Right (  427485.000, 1279215.000) ( 38d20' 5.68"E, 11d34'16.06"N)

Lower Right (  427485.000, 1277745.000) ( 38d20' 5.79"E, 11d33'28.21"N)

Center      (  426735.000, 1278480.000) ( 38d19'40.97"E, 11d33'52.08"N)

Band 1 Block=100x1 Type=Byte, ColorInterp=Undefined

Description = Layer_1
```

Step 5: gdal_translate -of GTiff 100.tif.sec-12 100.tif.sec-12.encrypted.tif

```
Input file size is 100, 98

0...10...20...30...40...50...60...70...80...90...100 - done.
```

APPENDIX C

a) AES CTR ENCRYPTION TIMING

```
getch@getch-ubuntu:~$ time openssl enc -aes-128-ctr -salt -in 80.csv -out 80.csv.enc

real    0m17.627s

user    0m0.004s

sys    0m0.004s

getch@getch-ubuntu:~$ time openssl enc -aes-128-ctr -salt -in 80.csv -out 80.csv.enc

real    0m7.742s

user    0m0.008s
```

```
sys    0m0.000s

getch@getch-ubuntu:~$ time openssl enc -aes-128-ctr -salt -in 80.csv -out 80.csv.enc

real    0m7.885s

user    0m0.008s

sys    0m0.000s
```

B) AES CTR DECRYPTION TIMING

```
getch@getch-ubuntu:~$ time openssl enc -d -salt -aes-128-ctr -salt -in 80.csv.enc -out 80.csv.new

real    0m5.462s

user    0m0.008s

sys    0m0.000s

getch@getch-ubuntu:~$ time openssl enc -d -salt -aes-128-ctr -salt -in 80.csv.enc -out 80.csv.new

real    0m5.801s

user    0m0.008s

sys    0m0.000s

getch@getch-ubuntu:~$ time openssl enc -d -salt -aes-128-ctr -salt -in 80.csv.enc -out 80.csv.new

real    0m4.216s

user    0m0.004s

sys    0m0.004s
```