DOCTOR OF PHILOSOPHY

Cryptanalysis Using Pattern Recognition Tools

Sharif, Suhaila Omer

Award date:
2013

Awarding institution:
Bangor University

Link to publication

Cryptanalysis Using Pattern Recognition Tools

A thesis submitted in candidature for the degree of

Doctor of Philosophy

by

Suhaila Omer Sharif

PRIFYSGOL
BANGOR
UNIVERSITY

School of Computer Science
Bangor University
United Kingdom
February 2013

Abstract

For cryptanalysis, an important task is to identify the encryption algorithm that was used to encrypt a plain-text file. The main objective of this dissertation is to find the best classification algorithm that can identify the encryption method for block and stream cipher algorithms.

This work provides a comparison of classification of encryption output between different types of block and stream cipher algorithms, with an evaluation between ECB and CBC modes using 8-bit and 16-bit codes. It provides results for different numbers of keys for all the encryption algorithms (block and stream cipher algorithms) that were analysed and determines the accuracy in each case.

We have created an encryption dataset that was used for the experimental evaluation. Different block and stream cipher algorithms were used to encrypt the source dataset which were a random sampling of text file data taken from the Internet in 2010 that included various types of data such as reports, papers, news, text from websites and journals. These samples ranged in sizes from 100 bytes to 10000 bytes. An initial analysis of the encrypted text shows that the data is random in nature. The Frequency Test shows a uniform distribution for the encrypted text. The Chi-square test also indicates the distribution of character codes is uniform. A compression test using the PPM text compression algorithm also shows that the encrypted text is uncompressible and therefore is random in nature. These tests show that the encrypted data is therefore difficult to classify.

The block and stream cipher algorithms used to encrypt the data used 8-bit and 16-bit codes. The study included two groups of block cipher algorithms: the first group considered the following block cipher algorithms: DES (64-bit), IDEA (128-bit), AES (128, 192, 256-bit) and RC2 (42, 84, 128-bit). The second group included another seven block cipher algorithms: RC2, RC6, Blowfish, Twofish, XTA, CAST and DESede (3DES), all with the same key size (128-bit). As well, the following stream cipher algorithms were investigated: Grain 128-bit, HC 128-bit, RC4 128-bit, VMPC 128-bit and Salsa20 128-bit.

The results from the classification experiment show that Pattern Recognition techniques are useful tools for cryptanalysis as a means of identifying the type of

encryption algorithm used to encrypt the data. As well, the result shows that increasing the number of encryption keys will result in reducing the classification accuracy. The results also show that it is possible to achieve an accuracy above 40% with some classifiers when each file is encrypted with different numbers of keys using block ciphers. It was also clear that increasing the number of files used also improves accuracy. The RoFo classifier had the best performance when identifying the encryption method for ciphered data, while IBL's performance was the worst. Moreover, the performance of the classifiers improved significantly when identification of four different algorithms was considered. It was noted that the three versions of AES (128, 192 and 256-bit) were not distinguishable within AES. Further, RC2 (128-bit) does not match the other versions of the same encoding RC2 (42, 84-bit).

For stream cipher algorithms, the results show that it is more difficult to classify encrypted output compared to block cipher algorithms. This is due to the bit based streaming approach adopted by the algorithms and the randomly distributed characters that are consequently produced in the encrypted output.

Acknowledgement

First of all, I please to record my sincerest thankfulness to an excellent supervisor Dr.Sa'ad Mansoor and my committee thesis Prof.Ludmila Kuncheva for being one of the best Supervisor and Professors; I have been ever familiar. Both supply me long term of study of inestimable support to my research while allowing me the freedom to prepare this thesis what it is. Apart from being an excellent Supervisor Dr.Sa'ad has as also been the best advisor I could have had during my stay at Bangor. Prof. Ludmila Kuncheva and Dr.Sa'ad has been a friend, a supervisor and an excellent teacher. Both have been beside me at every stage of my studies at Bangor University. With her and his help, I have been able to publish papers in various highly-respected journals and conference proceedings. Both also helped me in progressing towards my analytical skills when proving our theoretical results. Thanks, Dr. Sa'ad and Prof. Ludmila Kuncheva for your kindness. I want to thank Professor Nigel John for his advice. I want to thank staffs in School of Computer Sciences and colleagues who provided me with technical assistance and friendship during my student days. I would like to give my sincere gratitude to Olga for her help. I would also like to express my thanks to all my best friends (Ali, Shilan, Azida, Suhaib and Ibrahim) true friendship and unconditional support. Of course, I cannot forget my other colleagues: Nadim, Jess, Dr. Frank. I would like to express my heartfelt gratitude to my parents for providing me with a good education from the earliest days in life, and for education me to work hard and understand the value of that which is excellent. Furthermore, warm thanks to all my sisters and brothers. Many thanks to the School of Computer Science at Bangor University for helping. I would like to also express my thanks to Dlshad Othman Fathulla Deputy Minister of the Ministry of Finance in Kurdistan Region. I would like to thanks Dr. Tehan Bill for developing my thesis and giving helpful advice.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Cryptography and Network Security are both two important methodological approaches that protect data transmission and networks over the unsecured channel. With the rising popularity of using computer applications, specific security threats have appeared more and more frequently and therefore information security has become an extremely relevant and all important issue to be solved. Security is obviously a primary anxiety for every user of e-commerce. Cryptography is an extremely useful tool which decreases risks associated with observation and modification of information, where lasting secrecy is not significant but momentary integrity is. Network security is a similar method for preventing session take-over, and both of these are good examples of reasonable uses of cryptography. It helps people in many areas, whether through e-mail, cellular phones, ATM machines, for business, entertainment or education.

Cryptanalysis is an important task for cryptography. Cryptanalysis attempts to identify the weaknesses in the algorithms used to encrypt code or the methods used to generate keys. In cryptanalysis, when only the cipher-text is available, there are initially two significant tasks, identification of the encryption method used and the encryption key identification. Statistical methods and Machine Learning (ML) based methods have been used to identify the encryption method from the encrypted file. The statistical methods use the frequency of occurrence of the alphabet in the encrypted file, while in machine learning based methods, the task of identification of the encryption method is considered as a pattern classification task. The classifiers are used to capture the underlying behaviour of each encryption from a number of cipher-texts. The main purpose of identification of encryption method from cipher-text alone is considered to be a difficult one. Very little research has been done in this area when considering block cipher ad stream cipher algorithms.

Motivation

Identification of the encryption algorithm used to generate a piece of cipher-text is a key variable that may be gleaned from Cryptanalysis. The motivation of this study is to test and evaluate a novel approach which uses Pattern Recognition to attempt to classify cipher-text according to their originating algorithm. In this study eight pattern-recognition classifiers were examined, namely: Naive Bayes, SVM, MLP, IBL, Bag, AdaBM1, RoFo and C4.5. The study focuses on classifying cypher-text output of eleven different block and five stream cipher algorithms.

Hypothesis

Encryption algorithms operating on random input text files produce output which is notoriously difficult to distinguish between generating algorithm and traditional statistical approaches often fail. The hypothesis of this study is that by using Pattern Recognition based classifiers it is possible to effectively identify the encryption method used to generate a given cipher-text, even where the input text was random data.

Thesis Overview

In this section an overview of the thesis is provided detailing the procedure taken to evaluate the hypothesis.

Chapter 2 starts by discussing encryption and decryption algorithms and moves on to discuss the theoretical background of the thesis providing an overview of different types of cryptography. The reasons for applying cryptographic techniques are explained and the importance of also discussed. A description of how encryption works follows. Then Feistel ciphers and Substitution-Permutation Networks (SPN) are addressed. Modern cryptography, which includes Symmetric and Asymmetric algorithms, is defined. During the study it became clear that Symmetric algorithms were mathematically easier to calculate and needed less operational time than Asymmetric. Finally there is an overview of block and stream cipher algorithms. At the end of the second chapter a number of cryptographic algorithms used to encrypt and decrypt data were discussed. Then follows a brief review of the possible types of attack on block ciphers, focusing on Ciphertext-Only Attacks, in order to identify the encryption mode. There are various types of attacks, including Differential and Linear Cryptanalysis attacks which are explained here.

The literature review regarding Pattern Recognition is discussed in Chapter3. It begins by introducing of area of Pattern Recognition. The background and the concepts of the classification types are highlighted. First, Pattern Recognition techniques are introduced and an explanation of Multidimensional Scaling (MDS) method (used to find the similarities and dissimilarities between the algorithm) follows. The Statistical method and Machine Learning method (ML) are examined, chosen because they are common methods, and an explanation of their use in identification of the encryption mode and classification is given. An explanation of the use of WEKA tools follows along with a discussion and description of finding the accuracy of the eight classification types used in this study.

Creating and analysing the datasets is discussed in Chapter 4. This chapter begins by creating an encryption dataset to be used for the experimental evaluation and then analyses the created dataset to learn more about the encrypted data. This chapter shows that the encrypted data is random in nature and therefore difficult to classify.

Chapter 5, considers the classification as well as the evaluation of the classifiers' performances, both of which are critical problems in Pattern Recognition and Machine Learning. An explanation of the confusion matrix output and the success at identifying the encryption method using block ciphers in ECB and CBC modes is provided. The result shows that the Pattern Recognition techniques are a useful tool for identification of encryption algorithms for block ciphers. Further, it was discovered that RoFo is the most accurate classifier.

Chapter 6 deals with stream cipher algorithms using the same techniques as was applied to block ciphers in the previous chapter and also compares between block and stream cipher algorithms. It was discovered that there is only a slight difference in accuracy when implementing 8-bit or 16-bit codes. However different classifiers gave different accuracies in the two key lengths that were experimented with and it was found that block ciphers are more easy to identify than stream ciphers.

The final chapter concludes the work presented in this thesis with a summary and conclusion, a review of the thesis and directions for future work.

Contributions

The main contribution in this study is twofold: first, the creation of a dataset of encrypted files that can be used for evaluation of classification accuracy; and second, analysis of the encrypted text files. This study aims to find the best classification algorithm and identify the encryption method for block and stream cipher algorithms. This work provides a comparison of classification of encryption between different types of block and stream cipher algorithms, with an evaluation between ECB and CBC modes using 8-bit and 16-bit codes. It provides results for different numbers of keys for all the encryption algorithms (block and stream ciphers) that were analysed and determines the accuracy in each case.

As a result of this work, the following publications have been produced:

- Sharif, S.O. Mansoor, S.P., 'Performance Analysis of Stream and Block Cipher Algorithms ', Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference,1, V1-522 , 20-22 Aug, 2010.
- Sharif, S.O., Kuncheva, L.I., Mansoor, S.P., 'Classifying Encryption Algorithms Using Pattern Recognition Techniques ', Information Theory and Information Security (ICITIS), 2010 IEEE International Conference , 1168 - 1172 , 17-19 Dec, 2010.
- Sharif, S.O Manssor, S.P., 'Performance Evaluation of Classifiers used for Identification of Encryption Algorithms',
  Proc. of Int. Conf. on Advances in Information and Communication Technologies 2011, ACEEE, 172-175, 02.ICT.2011.2.60.

Chapter 2

Cryptography and Cryptanalysis

This chapter discusses the theoretical background of the thesis. It provides an overview of different types of cryptography and cryptanalysis. The chapter starts by discussing encryption algorithms in Section 2.1. Section 2.2 addresses the importance of cryptographies. Then Section 2.3 shows how encryption works while Section 2.4 addresses the Feistel ciphers and Substitution-Permutation Networks (SPN) are discussed. Section 2.5 addressed modern cryptography, which includes Symmetric and Asymmetric algorithms then Section2.6 is an overview of block cipher algorithms and Section 2.7 an overview of stream cipher algorithms. Section 2.8 introduces different types of Symmetric cryptography modes, while Section 2.9 explains block cipher cryptanalysis. Section 2.10 addresses Differential Cryptanalysis and Section 2.11 addresses Linear Cryptanalysis. Finally, the chapter concludes with a summary.

2.1 Introduction

Today, digital communication systems, particularly those linked to the internet, carry large amounts of sensitive data. Communication of such sensitive data must be both secure and secret. Common examples of such sensitive data include credit card details, bank account details, trade secrets and confidential e-mails. Less common examples include military and other confidential information. Cryptography is the science of keeping data secure. Today, it involves the scrambling of data, be it text, visual or audio files, so as to make the data unintelligible. This is done through encryption. The aim of encryption is to render the encrypted data safe from unauthorised parties, known as 'attackers'. The inverse of encryption is decryption, which decodes the data so as to render it intelligible to authorised parties. Technically, one can say that encryption involves algorithmically transforming plain-text(P) into cipher-text(C), thereby making it unintelligible to attackers. It then uses a key to decrypt the C back to P, thereby making it intelligible to authorised parties. For many years, it was thought that security in cryptographic algorithms was related to the difficulty of the mathematical operations underlying the encoding process [12]. In other words, the more complicated the coding process, the more difficult it was to break the code. This, as explained below, is not strictly true. Nonetheless, the core point is simple: the fundamental function in cryptography is to allow users to securely send and receive information over an insecure channel [13] [14] [15] [16].

Cryptography now takes three basic forms: Symmetric (secret key), Asymmetric (public key) encryptions and Hash functions. With Symmetric algorithms, encryption and decryption both apply the same key meaning that they share the same key. To guarantee the privacy of a Symmetric algorithm encrypted communication, the shared key must be kept secret from others. Notice a potential problem with this type of encryption method is that the key is needed to be shared between two or more parties prior to establishing the secure communication channel. Asymmetric algorithms separate encryption and decryption operations so that both sides are able to set up secure communications without exchanging keys in advance [17]. The hash function is the basis of a digital signature and message authentication for protecting the integrity of a data because of its collision-free and one-way properties [18] [19]. As an input, it takes a variable-sized message and produces a small fixed-sized chain as output. Whereas message authentication is the specific application of the security strength and hash function of the message authentication depending on the strength of the cryptography of the fundamental hash functions. Message authentication is a method that guarantees that delivered messages are from the alleged source and have not been modified [20].

2.2 The Importance of Cryptography

Simon Singh [21] [22] has highlighted the importance of cryptography in history. He cites, as just one example, the case of Queen Mary, imprisoned by the English and subject of the so-called Babington Plot of 1586: a plan to rescue Mary from prison. The plot was foiled because the plotters used a code to communicate with Mary, and thought the code to be unbreakable, which it was not. Elizabeth's agents soon broke it, which led to the unfortunate Mary being beheaded and the other plotters arrested and executed. A more successful use of encryption, also cited by Singh, involved US military intelligence during WWII. The US military employed Navajo Native Americans to communicate military intelligence by radio. The Native Americans used only their own language (they were

Navajo/English bilinguals) to do this. Because the Japanese did not understood Navajo and did not have the tools to help, they failed to "break" the code. The Navajo thus made an important contribution to the war in the Pacific. A less successful use of encryption, yet again reported by Singh, concerned Germany's use of the Enigma machine to encrypt its military intelligence during WWII. The Germans thought the code was so sophisticated that it was unbreakable. However, British Intelligence, with the help of Alan Turing and an early computer, broke the code. The code-breaking vastly facilitated the Allies' victory in WWII.

Thus cryptography is vitally important. Good cryptography helps win wars. Bad cryptography results in loss of wars as well as life. Today of course, cryptography is of more mundane importance. Banking, commerce and industry are increasingly conducted electronically. Recent scandals in the British press highlight the fact that what many consider private information can easily become public if such information is broadcast electronically. This prejudices personal identity (so-called identity fraud), the security of bank accounts and industrial secrets. Thus today cryptography is not only a matter for governments and the military; it is vital to everyone's security.

2.3 How Encryption Works

The basic principle of encryption is easy. One takes an intelligible string of information (e.g.*The Babington Plot failed*) and renders it unintelligible by algorithmically manipulating it. Thus, for instance, one could encrypt 'The Babington Plot failed' by simply printing successive letters of the alphabet instead of the original letters.

Thus: *Uif Cbcjohupo Qmpu gbjmfe.*

Such methods, however, are insecure. This is because the codes can be broken by simple statistical analysis. Thus, for instance, in English the letter *e* is the most common in printed text, and the letter *z* is among the most uncommon. From this, and guessing that common three-word constructions are likely to include 'the' as well as 'and', one is enabled to rapidly break the code. This method of cryptanalysis was developed in the ninth century by the Arab scholar al-Kindi (c.AD 801 873), although it took some time for it to become known in Europe (even-though it was used to outwit the Babington plotters).

A more effective method of encryption involves transforming each letter (or, in the case of digital information, each bit) in the plain-text in a different way. Thus, for instance, if one again takes *'The Babington Plot failed'*, one could first translate the text to the numbers representing each letter of the alphabet.

This would yield $20-8-5-2-1-2-9-14-7-20-15-14-16-12-15-20-6-1-9-12-5-4$. One could then take an irrational number, say the square root of two $(1.4142135623731...)$ and multiply each successive number in the P by each successive digit in the irrational number. This would render the *'The'* in *'The Babington Plot failed'* as 20325 making such a cipher much more difficult to break. A more sophisticated variant on this again is to take a huge number that has not only two, but huge factors. These factors are primes. The product of the primes can then be used to encrypt the message in the same way as described above.

The advantage of using the product of two enormous primes is that there is, as yet, no way to find the unique factors of a number other than through trial and error. Even the fastest of today's computers cannot, in practical terms, "break" codes constructed in this way, because to do so would take them too long. Thus, although in theory brute force may break a code, in practice brute force is often insufficient. Of course, there are countless other ways of encrypting material. One may, for instance, take an agreed page of a known book and simply write numbers, each relating to a successive letter on the page (if one decided to use a successive page for each successive message, this would be, in effect, a one-time pad, and as such unbreakable). One could also, if one wished to be fiendish, further encode such a cipher by multiplying it by the product of two enormous primes. How much one wishes to encrypt is thus in part a function of the importance of secrecy, and in part a function of the hassle involved in the encryption and decryption.

Two more points are relevant. First, the complexity of the encryption does not necessarily translate to difficulty of code-breaking. The Enigma machine, for instance, had a very complex form of encryption, yet its codes were broken. Conversely, the use of two large primes is relatively simple, but the codes so produced are, in practice, unbreakable.

Second, times change. The Enigma machine's codes would have been unbreakable in Elizabethan England, for Elizabeth's agents lacked 20th century mathematics and computers. Whether today's unbreakable codes will remain unbreakable is a subject of debate. Quantum computers (currently a topic of research and development), if developed, should be able to perform millions of calculations simultaneously. This would render them vastly more powerful than today's computers, and would render all currently used ciphers breakable, simply by brute force. Cryptographers then, have to have an eye for the future.

There are also different categories of encryption algorithms; plain-text may be processed either as block or stream ciphers. Algorithms may be classified by the type of operations used for transforming plain-text to cipher-text algorithms today are varied. These is an algorithm for prime factorisation [23] [24].

2.4 Feistel and Substitution-Permutation Network(SPN)

Feistel ciphers and Substitution-Permutation networks SPN are the two primary structures in block cipher algorithms. A Feistel cipher is a structure used in the building of block ciphers, labelled after the German-born physicist and cryptographer Horst Feistel. The structure has one large advantage: encryption and decryption are effectually equal, with the latter being simply the reverse of the former. This is efficient in terms of computer programming. Figure 2.1 shows Feistel encryption. In a Feistel cipher, the (N-bit) plain-text is split into (N/2)-bit parts. Each block is divided into two halves; (L_i is the first called the Left half most and R_i is the second called the Right half most) and the two half blocks pass into a number of rounds. In each round an initial permutation of the R_i is entered into the output, then the L_i will be complete, afterwards the R_i is passed via a keyed function applying an m bit key and lastly the output is combined with the L_i with an XOR operation. The basic function of a Feistel cipher is as follows [25] [26]:

$$L_i = R_{i-1}$$
$$R_i = L_i \oplus f(R_{i-1}, K_i)$$

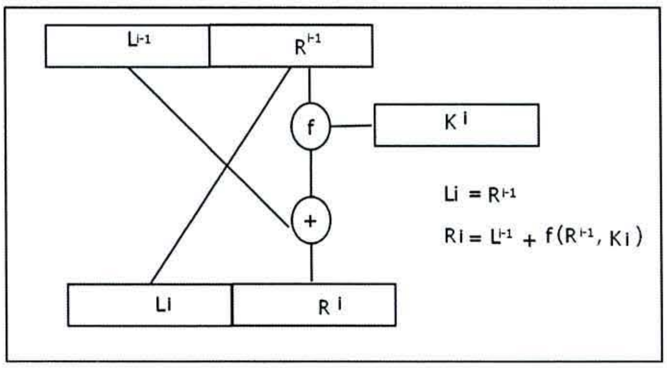


Fig. 2.1: Figure of Feistel cipher.

SPNs use a chain of mathematical operations. The effect of this is to take each block of plain-text and use it as an input. This input is then altered by many "layers" of substitutions. This is done in Substitution (S) and Permutation (P) boxes, which together produce the cipher-text. Common transformations include simple XOR bitwise rotation operators. Decryption then involves merely reversing the substitutions and permutations within the cipher-text. In Feistel, the structure has an advantage of the same algorithm between encryption and decryption, and the characteristic of the SPN structure is that it has a diverse algorithm between encryption and decryption [27].

2.5 Modern Cryptography

2.5.1 Asymmetric Key Encryption

Asymmetric encryption includes a public key in combination with a private key. A public key is an encryption algorithm that everyone is aware of private key is identifiable only by a private individual. The mechanism for the encryption is as follows: say Alice wants to send a coded message to Bob. Alice first encodes her message using the public key, then she re-encrypts the already encrypted message with her private key. She then sends it to Bob. Upon receipt of the message, Bob further encrypts it with his private key and returns it to Alice. Alice then removes her private key from the triple encrypted message, rendering it only double encrypted (with the public key and Bobs private key). She then sends it back to Bob, who can then remove both his private key and the public key. Thus asymmetric encryption allows for codes to be more secure than symmetric encryption does [17].

An important advantage of Asymmetric ciphers over Symmetric cipher is that no secret channel is needed for the exchange of the public key. The receiver needs only to be assured of the authenticity of the public key. The Asymmetric cipher is applied in order to encrypt a session key, and the encrypted session key is then applied to encrypt the real message. Due to the high speed of Symmetric ciphers, the key-exchange benefits from using Asymmetric ciphers, which work at a lower speed and are therefore more precise.

2.5.2 Symmetric Key Encryption

As a basis for information security Symmetric ciphers have long been utilised. While they are mainly developed for data confidentiality, their flexibility lets them be used in various cryptographic systems; these include hash functions, pseudo random number generators and message authentication protocols [25].

Although Symmetric encryption is not as secure as Asymmetric encryption, it has the advantage of speed. Symmetric encryption methods are approximately 1000 times faster. Because of this, they require less computational processing power. Figure 2.2 shows the Symmetric encryption algorithm.

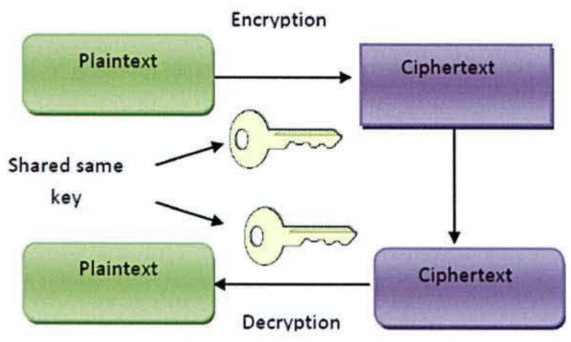


Fig. 2.2: Symmetric Key Encryption.

Given that Symmetric systems, as is witnessed by hash functions, can be very difficult to break, their relative insecurity when contrasted with Asymmetric systems is more than compensated for by their greater ease of use. There are two categories of algorithm in Symmetric algorithms: block and stream. Block ciphers typically use 64 or 128-bit at a time. A message longer than the block size is encrypted splitting the message into blocks and encrypting each block separately. Generally, block ciphers use several rounds of simple cryptographic operations. In addition, the cipher key is expanded to a number of sub-keys by a key schedule, and sub-keys are mixed with data blocks in different rounds, characteristically with bitwise XOR operations. These procedures lead to high performance, with the result that block ciphers are now used extensively. Cryptography is applied to modify readable text (identified as Plain-text) into an unreadable secret format (known as cipher-text) using a method called encryption [28] [29] [30].

The process of the two methods is identical except for the amount of data each encrypts at each step. First, a block cipher uses the most modern encryption methods, second, there are two categorise of classes: Substitution or Transposition ciphers. They differ according to what chunks of the message are handled by the encryption techniques. *Substitution* ciphers map each element in the plain-text onto another element. *Transposition* is a technique of encryption by which the positions held by units of plain-text are shifted according to a regular system. Consequently, the cipher-text constitutes a permutation of the plain-text. A Transposition cipher does not modify the characters in the plain-text while it creates the cipher-text, it simply re-arranges them. It uses some type of permutation function within the text to produce a re-arrangement that is able to be reversed if the secret is known to the permutation [23] [24]. The effectiveness of the Symmetric key depends on the length of the keys used. For the strongest algorithm, encryption using a longer key is more difficult to break than one using a smaller key.

2.6 A Brief Overview of Block ciphers

Effective cryptography involves compromise. For example, if the block size is small it will be relatively easy for attackers to decrypt the message but if the block size is too large, it might be inconvenient to use [31]. If a message is longer than the block size (128-bit), it must break the message into a block. Each block is encrypted individually. Because of the dangers of attack, the US National Institute of Standards and Technology (NIST) has standardised five modes of operation: ECB, CBC, CFB, OFB and CTR. These are explained more in Section 2.8 [32].

2.6.1 Data Encryption Standard (DES)

The Data Encryption Standard (DES) was adapted as the US federal standard in 1976. It is a Symmetric block cipher algorithm which employs the same key for encryption and decryption [9] [33]. DES has 64 binary digits (0s or 1s); the effective key of 56-bit is randomly produced; and the other 8-bit are ignored. Figure 2.3 shows the structure of the DES algorithm [23] [34] [35] [36].

DES has three mechanisms: the Initial Permutation IP, the Round F-function and the Inverse of Initial permutation IP^{-1}. It uses the Feistel cipher construction with 16 rounds of processing. The first stage of block encryption starts with the Initial Permutation IP, subsequently goes to 16 rounds, and ultimately to the final permutation(IP^{-1}) [37]. The DES round function applies sixteen 48-bit keys to the rightmost 32-bit R_{i-1} to generate a 32-bit output[23]. However, the round function is made up of four operations: Expansion (P-box), XOR operation, Substitution (S-box) and P-box Permutation. When the DES round is applied 16 times, it uses a different 48-bit key. When the block is less than 64-bit, it must be padded in order to be capable of the request [23] [38] [39]. Normally, a block is compiled of bits numbered from left to right, that is, the left most bit of a block is bit one and the right most is zero [10] [40] [41]. Figure 2.4 shows a block diagram of F-function in DES algorithm.

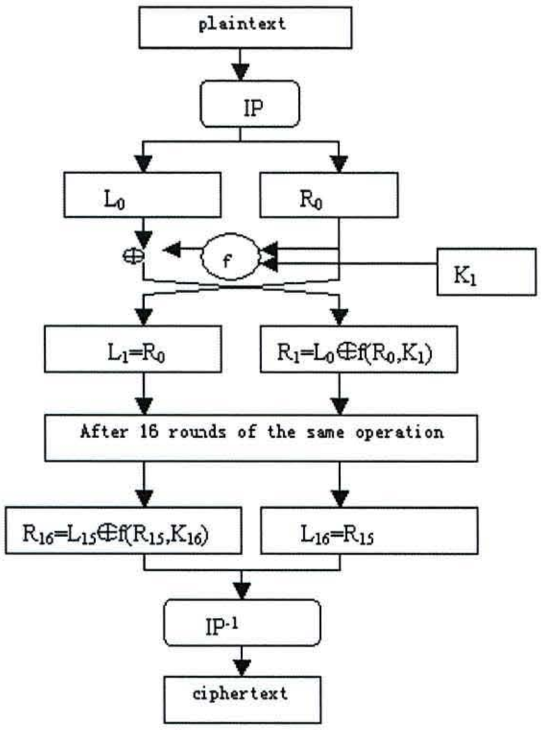

plaintext
IP
L_0
R_0
f
K_1
$L_1=R_0$
$R_1=L_0\oplus f(R_0,K_1)$
After 16 rounds of the same operation
$R_{16}=L_{15}\oplus f(R_{15},K_{16})$
$L_{16}=R_{15}$
IP^{-1}
ciphertext

Fig. 2.3: The DES algorithm [1].

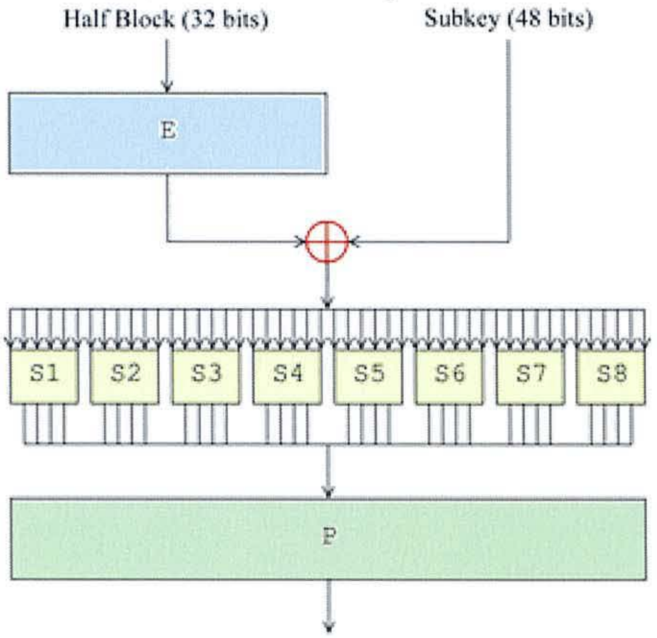


Fig. 2.4: The F-function of DES [2].

Initially [42] [43], a block of the 64-bit permutation input IP plain-text will be divided into two halves, the halves are called Li (Left half) and Ri (Right half), with each half consisting of 32-bit. P-box is clearly a permutation and nothing else; it has a one to one mapping of its input to its output providing a 32 bit output from a 32-bit input. Table 2.1 shows the IP and Table 2.2 the IP^{-1}. These permutations are key-less straight permutations, which are the inverse of each other. For example, in the IP, the 58^{th} bit in the input becomes the first bit in the output and so on with bit-7 as its final bit[23] [33] [44]. However, the number of block sizes are fewer than 64-bit, which must be padded as appropriate for the application. The algorithm has output of bit-40 of the pre-result block as its first bit, bit-8 as its second bit, and so on, until bit 25 of the pre-result block is the last bit of the output [9].

Tab. 2.1: Initial Permutation [9].

58	50	42	34	26	18	10	02
60	52	44	36	28	20	12	04
62	52	46	38	30	22	14	06
64	54	48	40	32	24	16	08
57	56	41	33	25	17	09	01
59	49	43	35	27	19	11	03
61	53	45	37	29	21	13	05
63	55	47	39	31	23	15	07

Tab. 2.2: Final Permutation IP^{-1} [9].

40	08	48	16	56	24	64	32
39	07	47	15	55	23	63	31
38	06	46	14	54	22	62	30
37	05	45	13	53	21	61	29
36	04	44	12	52	20	60	28
35	03	43	11	51	19	59	27
34	02	42	10	50	18	58	26
33	01	41	09	49	17	57	25

Having produced the 32-bit output, as above, this function could be generated with four operations: Expand P-box, XOR, substation S-box or straight P-box. A round of the DES can be viewed as [10] [45].

$$L_i = R_{i-1} \quad \text{where } i = 1...16$$

$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i) \quad \text{where } i = 1...16$$

Afterwards the 64-bit are divided into two halves, the Ri is approved during the F-function transformation by the 48-bit sub-key to produce an exclusive-XORed

Tab. 2.3: Expand P-Box [9].

(a) Primitive function

16	07	21	21
29	12	28	17
01	15	23	26
05	18	31	10
02	08	24	14
32	27	03	09
19	13	30	06
22	11	04	25

(b) Bit-selection E(P-Box).

32	01	02	03	04	05
04	05	06	07	08	09
08	09	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	01

with Li [23] [45]. In addition, the DES in explanation operation used the Table 2.3(a) to identify this P-box. First, it is required to expand R_{i-1} (32-bit) to 48-bit keys $K_1, .., K_{16}$, which were derived from the supplied 56-bit key by making two copies of half it bits. Subsequently 48-bit are XORed with the round key K_i, excluding $E(R_{i-1})$ with K[i] [40]. Some of the input goes to more than one output. Table 2.3 (b) shows the expanded P-box.

The XOR is the second operation in the DES F function. The E(P-box) is XORed with the K_i round key into 48-bit, which is then split into eight 6-bit blocks and are used as inputs for the defined S-boxes. [28]. A round key must be used in this operation; the Ri and Ki are both 48-bit in length. Note that the S-Box B[1] consists of bits 1-6, B[2] consists of bits 7-12, and so on with bits 43-48 being B[6] [33] [44] [46]. DES applies eight S-boxes, each one with a 6-bit input data and a 4-bit output data [23]. The final operation of DES is a straight permutation with a 32-bit input and a 32-bit output. Table 2.4 shows the straight permutation table. This operation will follow the same universal rule as the previous permutation [38] [23]. The Substantiation S-box is where the cipher function obtained its security.

The S-box is a set of 8-bit with two dimensional arrays consisting of four rows and sixteen columns. Each box consists 4-bit in length. The result from the XOR operation is eight 6-bit segments. Figure 2.4 shows that the left half which consists of the 6-bit are B[1], and the right half consists of the 6-bit are B[8]. A 32-bit is produced which is then passed by permutation(P-box) [30] [28] [34].

The to being generated can be inserted directly or be the consequence of the hashing of another operation. For this aim, there is no standard hash function algorithm. Initially, the key must be reduced from a 64-bit to a 56-bit. Therefore, bit 1 of the (PC1) is bit 57 of the original key and bit 2 is bit 49 and so on[23]. The following Table 2.5 establishes permuted choice 1(PC-1). The 56-bit are divided into 2 parts. Each part consists of 28 bit, one is called C[0], the other D[0]. Subsequently, on calculation the 16 iteration sub-keys will start with i (the number of the round) which equals 1 to 16. One or two circular left shifts must be made on both C[i-1] and D[i-1] to obtain C[i] and D[i]. The number of per-iteration is set in the sub-key rotation shown in the Table 2.6 below [46]. Then the chain of the C[i] and D[i] are permuted, as shown below. This will produce the K[i]; the length of the key is 48 bit i.e. C16 is equivalent to C0, and D16 is equivalent to D0 [10] [45].

Tab. 2.4: Straight Permutation [10].

16	07	20	21	29	12	28	17
01	15	23	26	05	18	31	10
02	08	24	14	32	27	03	09
19	13	30	06	22	11	04	25

Tab. 2.5: Permutation Choice(PC1)[10].

57	49	41	33	25	17	09
01	58	50	42	34	26	18
10	02	59	51	43	35	27
19	11	03	60	52	44	36
63	55	47	39	31	23	15
07	62	54	46	38	30	22
14	06	61	53	45	37	29
21	13	05	28	20	12	04

Tab. 2.6: Rotation Sub-key [10].

Round Number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Number of bits to rotate	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

2.6.2 Advanced Encryption Standard

Advanced Encryption Standard (AES) algorithms have been extensively accepted and used for sensitive data security, such as in many password-protected documents and wireless communications, for example in wireless sensor networks [47]. Vincent Rijndael and Joan Daemen developed the Rijndael algorithm, which was chosen by the U.S. National Institute of Standards and Technology (NIST) as the candidate for the AES. It was shown to be the most important algorithm and therefore it replaced DES because it was no longer held to be secure. AES is a Symmetric block cipher having data lengths and variable key lengths of 128, 192, and 256-bit encryption keys. Among all the software products AES has become one of the most popular and extensively applied common block ciphers. It achieves encryption and decryption using 128-bit. The block cipher could be expressed as a row matrix of 16 bytes. The 128-bit input message is separated into a 4-by-4 matrix of 8-bit, known as a state, and in order to encrypt the data it uses a number of rounds of operations. The number of rounds is determined by the size of the key (10 rounds for 128, 12 rounds for 192, and 14 rounds for 256), Table 2.7 shows the number of rounds and data length [48] [49] [50] [51] [52].

AES employs one of three different cipher key strengths as encryption keys. Each encryption key size permits the algorithm to perform somewhat differently. Consequently, the increasing key sizes not only proffer a larger number of bits with which to rearrange the data, they also increase the difficulty of the cipher algorithm, rendering the cipher more secure. In the encryption part, initially the data block to be encrypted is split into an array of bytes, referred to as a state matrix, shown in Figure 2.5. AES is based on a round function and achieves different combinations of the algorithm by repeating the round function at different times.

AES has four steps: Byte Substitutions, Shift Row, Mix Column, and Add Round Key. The final round of the algorithm is similar to the standard round, except that it does not include a Mix Column operation[51] [53] [11]. A round of transformations consists of the repeated application of Nr (the number of the round) and depends on the block and key lengths. An encryption with Rijndale [51] consists of an initial key addition (Add Round Key), then Nr-1 applications of the transformation round and finally one application of the final round. Table 2.7 shows the number of round and data length [11], while Figure 2.6 shows the encryption process.

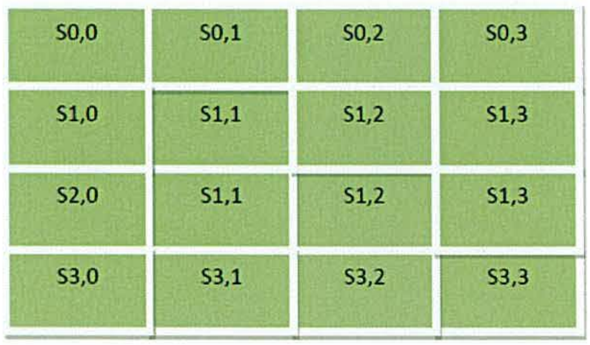


Fig. 2.5: State matrix.

Each round begins with a *Sub-bytes* which is a non-linear byte substation that works independently. The state is treated as a 4-by-4 matrix of bytes; all bytes of

Tab. 2.7: Number of rounds and data length.

Number of rounds(Nr)	128-bit	192-bit	256-bit
128-bit Key	10	12	14
192-bit Key	14	12	14
256-bit Key	14	14	14

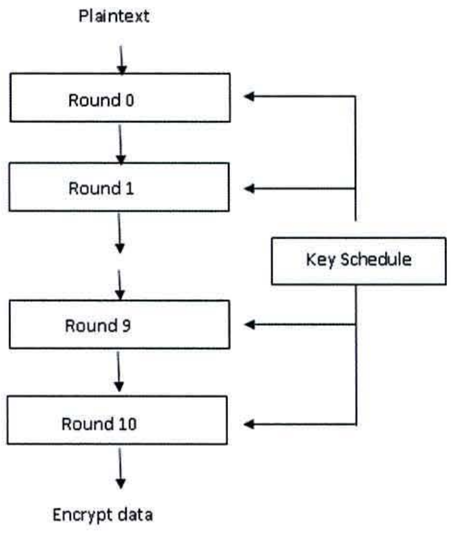


Fig. 2.6: AES Encryption Process [3].

the state uses a substation table (S-box). Figure 2.7 shows the Sub-Byte operation [23] [11].

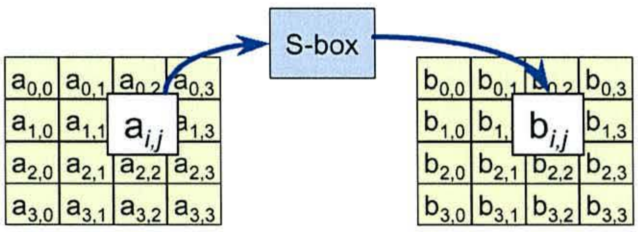

S-box
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{i,j}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
$b_{0,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$
$b_{1,0}$ $b_{1,1}$ $b_{i,j}$ $b_{1,3}$
$b_{2,0}$ $b_{2,1}$ $b_{2,2}$ $b_{2,3}$
$b_{3,0}$ $b_{3,1}$ $b_{3,2}$ $b_{3,3}$

Fig. 2.7: Sub-byte Transformation [3].

Shift-rows transformation: this is a transportation step that permutes the bytes such that every row of the state is shifted to the left. The number of the shift is a function of the number of the row (0, 1, 2, or 3) of the state matrix. The first row remains without shifting, but each byte in the second row is shifted once to the left. Respectively the third and fourth rows are shifted by offsets of two and then three. Figure 2.8 shows the shift operation [23] [11].

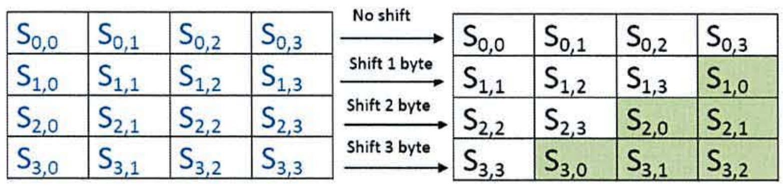

No shift
Shift 1 byte
Shift 2 byte
Shift 3 byte
$S_{0,0}$ $S_{0,1}$ $S_{0,2}$ $S_{0,3}$
$S_{1,0}$ $S_{1,1}$ $S_{1,2}$ $S_{1,3}$
$S_{2,0}$ $S_{2,1}$ $S_{2,2}$ $S_{2,3}$
$S_{3,0}$ $S_{3,1}$ $S_{3,2}$ $S_{3,3}$
$S_{0,0}$ $S_{0,1}$ $S_{0,2}$ $S_{0,3}$
$S_{1,1}$ $S_{1,2}$ $S_{1,3}$ $S_{1,0}$
$S_{2,2}$ $S_{2,3}$ $S_{2,0}$ $S_{2,1}$
$S_{3,3}$ $S_{3,0}$ $S_{3,1}$ $S_{3,2}$

Fig. 2.8: Shift-rows Transformation.

Mix-Column Transformation: the mixing operation changes the contents of each byte by obtaining four bytes at a time and then combining them to rebuild new bytes. The combination method starts by multiplying each byte with a different constant and then combines them in order show that each new byte is different. At this point, matrix multiplication is complete and shows that the new column is the product of the two old columns. *Add-round key transformation*: this is similar to mix-column transformation, also using one column. Every single byte of the state is mixed with a sub-key, which is obtained by applying the key schedule. Each sub key is identical in size to the state. By combining each byte, the sub-key is added to the state with the equivalent byte of the sub-key applying the XOR operation. The operation is a matrix multiplication [23] [11].

Add-Round Key Transformation: this is similar to Mix-Column transformation, also using one column. Here, every single byte of the state is mixed with the sub key, a sub key is derived from applying the key schedule. However, each sub-key is identical to the state. By combining each byte, the sub-key is added to the state with the equivalent byte of the sub-key applying *XOR* operation, which is a matrix multiplication[23] [11].

Key schedule: the sub keys are obtained from the cipher key, meaning that the key schedule with each sub key is needed for the number of words of key data, for example, a 128-bit key schedule on the block of four 32-bit words. Round Keys involve two components: the Key Expansion and the Round Key[23] [11].

Key Expansion: an AES algorithm is applied to a key expansion in order to generate a round key for every round. If the number of rounds is Nr (Number of rounds), + 1 128-bit round keys from one single cipher key. The key-expansion routine generates a total of Nb (Nr+1) words, the algorithm needs an initial set of Nb words, and Nb words of key data are required for every Nr. The function of key expansion takes the user key, which is 16 bytes long, and uses the round constant called "matrix-rcon "with the substitution table S-box to produce 176-byte long key schedule w, which will be utilized during the encryption and decryption methods. In addition, the key-expansion routine generates round keys word-by-word; a word is an array of four bytes. After the routine 4-bytes N^{r+1} words are generated, which are called "$w_0, w_1, w_2 ... w_4(Nr+1) - 1$ ". The consequence key schedule comprises of a linear array of 4-by-4 words represent as (wi), where i in the range $0 \leq i < Nb(N_{r+1})$.

In other words, there are 44 words in the AES 128-bit (10 rounds), 52 words in AES 192-bit (12 rounds), and in the AES 256-bit (14 rounds) there are 60 words. Each round is complete with four words. The relationship between rounds and rows is shown in Table 2.8 [23] [11].

Tab. 2.8: Shows the relationship between rounds and rows [11].

Round	Words			
Per-round	W_0	W_1	W_2	W_3
1	W_4	W_5	W_6	W_7
2	W_8	W_9	W_{10}	W_{11}
...	...	...	...	...
Nr	W_{4Nr}	W_{4Nr+1}	W_{4Nr+2}	W_{4Nr+3}

2.6.3 Triple Data Encryption Standard

Triple data encryption standard (3DES) is an improvement of DES. It has a 64-bit block size, with a 192-bit key length (3×64). The encryption technique is like that of the original DES. However, the blocks are used three times to raise the standard of the encryption and the safe time average, without needing a complete new block cipher. Figure 2.9 shows the structure of 3DES algorithm (Encryption, Decryption).

In order to gain higher security, the three keys should be separate. Significantly, this is equivalent to using a length of 168-bit key for encryption. It has an identical process of encryption as a regular DES algorithm. 3DES employs a key bundle, which means that it takes three 56-bit DES keys, which are K1, K2 and K3. The input data is encrypted with the $K1$, decrypted with the $K2$, and finally encrypted again with the $K3$ (encrypt, decrypt, encrypt). DES runs three times faster than 3DES, however, if applied properly 3DES is more secure. The procedure for encrypting data is the same as for decrypting but it is implemented in reverse.

In DES, the input key is 64-bit long, whereas the definite key applied by DES is only 56-bit long. Therefore the true key length of 3DES is 168-bit because each of the three keys holds 8 parity bits that are not applied throughout the encryption method [54] [55] [56] [57] [58].

The following function defines the 3DES algorithm [59]:

$$C = DESk_3DES{-}1k2DESk_1(p)_1$$

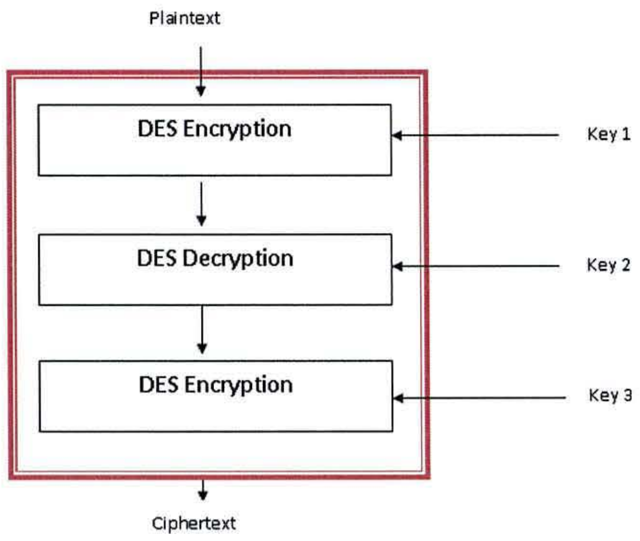


Fig. 2.9: Encryption and Decryption of 3DES.

Further, 3DES can work with one, two or three 56-bit keys which means the plaintext, in outcome, is and encrypted three times with three keys. Following is a number of modes options of 3DES algorithm [60] [61]

- Option 1 DES-EEE3: This means DES is encrypted with three different keys.
- Option 2 DES-EDE3: This means three DES operations with three different keys. Moreover, the data are encrypted, decrypted, and encrypted.
- Option 3 DES-EEE2 and DES-EDE2: Which the similar to the preceding options except that the first and third operations in DES-EEE2 are applied with a similar key.

2.6.4 International Data Encryption Algorithm (IDEA)

In November 2000, International Data Encryption Algorithm (IDEA) was submitted as a block cipher to the NESSIE Project within the Information Societies

Technology (IST) programme of the European Commission (EC). IDEA is a Symmetric algorithm. It is employed to generate 52 subkeys, each with 16-bit keys. Two of the subkeys are applied during each round proper, and the other four are applied before each round and after the final round. It comprises eight rounds. IDEA is a 64-bit block cipher using a 128-bit secret key. In IDEA the plain-text is split into four quarters, each 16-bit long, including three operations, which are used to mix two 16-bit values to create a 16-bit outcome. The three operations are: addition, XOR, and multiplication [62]. First, addition is normal addition which carries modulo 65,536 which is (2^{16+1}). Second, multiplication, as applied in IDEA, needs a number of explanations. Generally, multiplication by zero always generates zero, and it is not invertible. However, multiplication modulo n is also not invertible when it is multiplied by a number that is not relatively prime to n. Using multiplication in IDEA, it is essential that it is always invertible [63] [29].

The decryption follows the exact same pattern, this time in reverse order. The cipher relies on combining operations from three categories. The three operations using a vector of length 16-bit are as follows: First bitwise addition modulo 2 (**XOR**) of two 16-bit sub blocks; second addition of integers modulo 2^{16}; and third multiplication modulo p and multiplication $modulo\, p = 2^{16} + 1$. Annotation of the value 0 is never used; 2^{16} is represented by all the zero vectors [64] [65].

2.6.5 Twofish

Twofish is a block cipher with a 128-bit block and a variable key length up to 256-bit. It consists of a 16-round Feistel structure with a bijiective F-function, which creates a 4 key dependant on 8-by-8-bit S-boxes, with a fixed 4-by-4 maximum over $GF(2^8)$ distance [66]. However, the only non-Feistel elements are the 1-bit rotation. To generate a pure Feistel structure, the rotations can be moved into the F-function but this needs an extra rotation of the words immediately before the output-whitening step. See Figure 2.10 for an overview of the cipher construction. Whitening is an XORed key material method conducted on block ciphers before the first round and after the last round. Rivest independently invented DES-X which was applied by Merkle in Khufu and Khafre [67] [4].

Twofish algorithms use 128-bit of the sub-key and are then XORed before the first round, and another 128-bit after the last round. These sub-keys are calculated in the same way as the round sub-keys, however, they are not applied anywhere else in the cipher [67].

In the algorithm, the 128-bit (plain-text) is separated into four 32-bit words. In the input whitening operation these are XORed with four key words with 16 rounds. In each round, inside the g-function, one round is rotated by eight bits. The two words on the left side are applied as inputs to two g-functions. However, the g-function is made up of four byte-wide key-dependent S-boxes, which is followed by a linear mixing procedure based on a Maximum Distance Separable Matrix (MDS). Subsequently, the results of the two g-functions are mixed applying a Pseudo-Hadamard Transform (PHT) in which two keywords are added. Then these two outcomes are XORed with the words on the right (one of which is first rotated left by 1 bit; the other is rotated right). For the next round, the left and right halves are then exchanged. Finally, after all the rounds are swapped, the last round is reversed, and the four words are XORed with four other key words in order to create the cipher-text. Formally, the 16 bytes of plain-text, that are known as $p_0, ..., p_{15}$, are first separated into 4 words ($P_0, ..., P_3$) of 32-bit. Each word uses the little-ending convention as follows [67] [68] [4]:

where:

P represents the number of words and *p* represent the Plain-text.

$$P_i = \sum_{j=0}^{3} p(4i + j).2^{8j} \quad \text{where } i = 1...3.$$

Afterwards, during the input whitening operation, the four words are XORed with another 4 words to expand the key:

$$R_{0,i} = P_i \oplus K_i \quad \text{where } i = 1...3.$$

In each round in the 16 rounds, the first two words were applied as input to the F-function, when it is applied as input it takes the round number. Then the third word is XORed with the first output of F-function and then rotated left by one bit. The fourth word is rotated left again by one bit, then XORed with the second output word of F-function. Finally, the two halves are swapped [68] [69] [70].

$$(F_{r,0}, F_{r,1}) = F(R_{r,0}, R_{r,1}, r)$$

$$R_{r+,0} = ROR(R_{r,2} \oplus F_{r,0}, 1)$$

$$R_{r+1,1} = ROL(R_{r,3}, 1) \oplus F_{r,1}$$

$$R_{r+1,2} = R_{r,0}$$

$$R_{r+1,3} = R_{r,1}$$

Where:

For $r = 0, ..., 15$

ROR and ROL are functions which rotate their first 32-bit words left or right by the number of bits that are indicated by their second 32-bit words, then the output whitening operation undoes the 'change' of the last round, and then XORes the words with four words of the expanded key. Finally, the cipher-text written as 16 bytes $c_0, ..., c_{15}$ and then use the little-ending conversion as used for the Plain-text.

$$C_i = R_{16,(i+2)}mod_4 \oplus K_{i+4} \quad \text{where } i = 1...3.$$

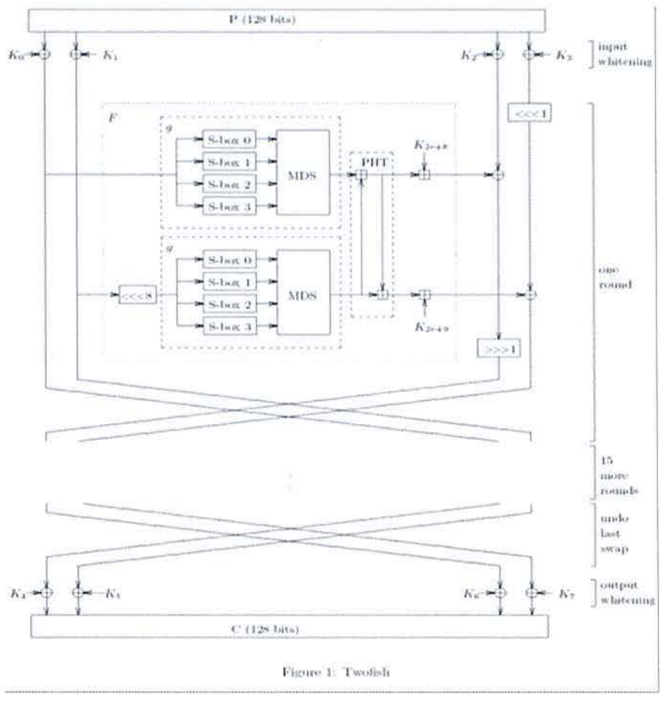


Fig. 2.10: An overview of the cipher construction [4].

2.6.6 Blowfish

Bruce Schneier developed Blowfish specifically for use in performance-constrained environments such as embedded systems, and to replace DES, which is now in the public domain. It was initially developed in 1993, and until now has not been decoded. The cryptographic community deemed it 'reasonably secure'. It is a Symmetric algorithm, that is, it applies the same secure key between the sender and the receiver as a block cipher. The block length for Blowfish is 64-bit; variable-length keys can be up to 448-bit. Figure 2.11 shows the algorithm.

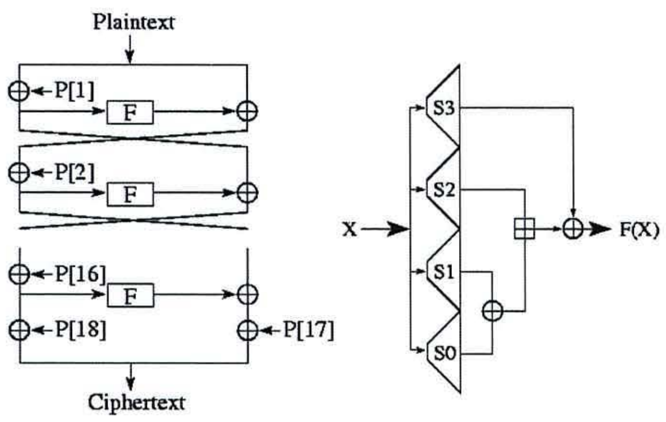


Fig. 2.11: The Blowfish algorithm [5].

There are two components of Blowfish: a data encryption and a key-expansion. The key-expansion transforms a 448-bit key into the number of sub-keys totalling 4168 bytes. Data encryption works through a 16 round network. Each round comprises a key-dependent substitution and a key-dependent permutation. All functions are XORed with additional 32-bit words. The communications link and automatic file encryption does not require the key to be changed during the function. On 32-bit microprocessors with longer data caches (e.g., Pentium and Power-PC) Blowfish is faster than DES [71] [72] [73] [74].

2.6.7 RC2

RC2 is a block cipher developed by Ronald Rivest in 1989. It was published as an Internet Draft during 1997 [75], RC stands for "Rivest Cipher ". It has a 64-bit block cipher with variable key length starting from 40-bit to 128-bit. The 40-bit key is relatively weak, since the encryption key is small. RC2 is susceptible to key attacks which use 234 selected plain-texts. The algorithm has only one P-box, which is used for key expansion to load into memory. The RC2 algorithm is quickest for small files [6]. Note that RC1 was only designed on paper but was never implemented in practice and that RC3 was observed to be breakable during development [76].

2.6.8 RC5

RC5 is a Symmetric block cipher suitable for hardware and software. It is developed by Ronald Rivets in 1994 [77]. The same secure key is applied onto encryption and decryption data. Thus RC5 converts plain-text data blocks of 16, 32, and 64-bit into cipher-text in similar blocks of the same length and then employs a selectable key length (0, 1,...,255) byte. It contains a number of modular-like additions and XORs. The universal construction is a Feistel network. RC5 is a set of rotations identified as rounds r. These have values in the range of 0, 1, ...,255, with a changeable block size and a key length. This allows for flexibility in performance characteristics and security levels [77]. The size of the block of plain-text is twice the size of a word, one block contains two words. RC5-$w/r/b$, is a version of RC5 where w represents the word size bits; r represents the non-negative numbers of rounds, and b represents the length of the encryption key in bytes. The RC5 algorithm can be expressed as RC5-$w/r/b$.

where:

- w = Word size in bit, which RC5 encrypts as 2-words blocks;
- r = Number of rounds, allowable value (0, 1,..., 255);
- b = Number of 8-bit bytes (Octal) in the secret key (K).

RC5 comprises three parts: Key expansion, Encryption and Decryption. Figure 2.12 shows RC5 encryption algorithm. RC5 has been widely used in Wireless Transport Layer Security. It is also used for smart cards and other machines with restricted memory. The encryption and decryption algorithms are exceptionally straightforward [78] [77] [79]. To decrypt, RC5 reverses the encryption algorithm. Most processors efficiently support all 43 of RC5s operations.

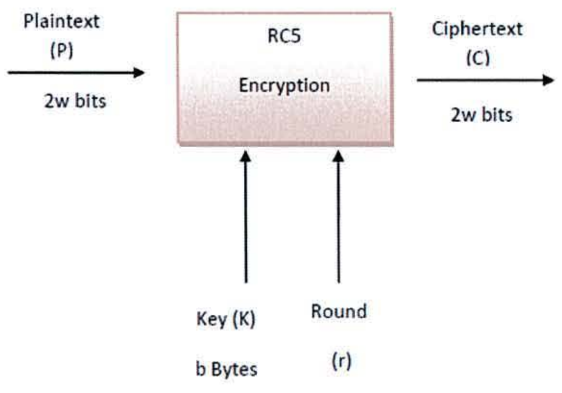


Fig. 2.12: RC5 Encryption Algorithm.

The key-expansion algorithm increases the user key(K) to fill the extended key table S; S starts from 1 to 25; therefore, S resembles an array of $t = 2(r+1)$ random binary words concluding with K. Key expansion uses two word-size magic constants: P_w, and Q_w [77] [80].

2.6.9 RC6

RC6 is a Symmetric block cipher. It was developed to match the requirements of AES. It has a simple structure and was one of five finalists that could compete with AES. The RC6 algorithm consists of two Feistel networks, in which data are mixed through data-dependent rotations [81]. RC6-$w/r/b$ (as explained in RC5), is a version of RC6 that functions on units of four w-bit words employing six basic functions. The base-two algorithm of w can be represented by lg w [28] [50] [28] [82]. Figure 2.13 shows the encryption of RC6 algorithm.

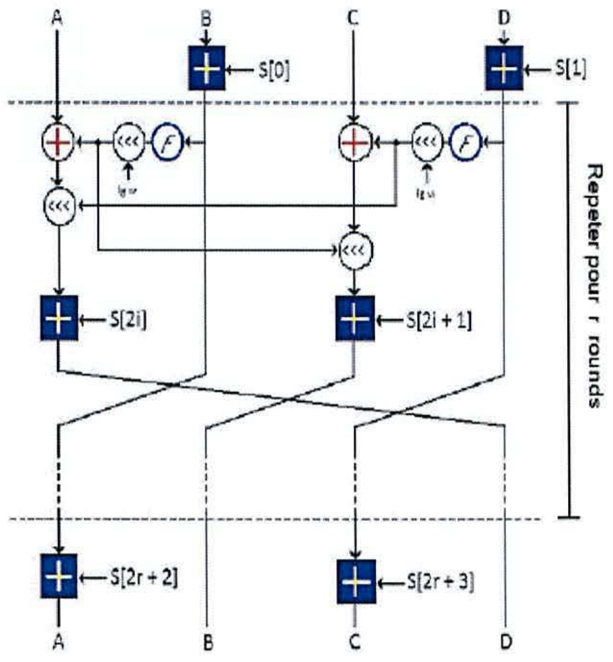


Fig. 2.13: RC6 Encryption Algorithm [6].

2.6.10 Serpent

The Substitution-Permutation Network (SPN) employs a block cipher that ensures that an SNP satisfies the cryptographic property of completeness: all output bits are a function of all input bits. Serpent is a symmetric block cipher. The Serpent algorithm consists of a 32-round (SPN) operating on four 32-bit words, therefore giving a 128-bit sub-key block size. It also requires another 128-bit sub-key which needs to be XORed with the block. However, the sub-keys are created by the key generation method. Each value employed in the cipher are symbolized as bit streams (which is a sequence of bits). Then the index of the bits are calculated from 0 to 31-bit word, 0 to 127-bit in 128 bit blocks, 0 to 255-bit in 256-bit keys and so on. It encrypts a plain-text with 128-bit to a cipher-text with 128-bit in a 32 rounds governed by 33 128-bit sub-keys $K_0, ..., K_{32}$. The key lengths are 128, 192 or 256 bit; the short keys (128 and 192) are mapped to full-length keys of 256 bit. This type of mapping is intended to map each short key to a full length key. The Serpent algorithm involves three key parts [83] [84] [85]:

- Initial Permutation (IP);
- 32 rounds, each of them created upon a sub-key addition;
- Final Permutation (FP).

The SPN is employed in the design of block cipher to ensure that an SNP satisfies important cryptographic properties: completeness if the all output bit is a function of the all input bit. Nevertheless, decryption is not like encryption because the inverse of the S-boxes is required in reverse order; there is also an inverse linear transformation and reverse order of the sub-keys. The IP and FP do not have any cryptographic significance [86] [87] [88] [89].

2.6.11 CAST-128

Carlisle Adams and Stafford developed CAST-128, otherwise known as CAST5 [90]. Its specification was published as RFC2144. It was accepted by the CSE 46 (Communications Security Establishment). The Government of Canada also selected it as one of the algorithms accepted by the ISO (the International Organization for Standardization) and the IEC (the International Electro-technical Commission) for the specialised system for worldwide standardisation [91]. It is a Symmetric block cipher with 64-bit block, 12- or 16-rounds and a key size of between 40 and 128-bit. When the key size is longer than 80, the full 16 rounds are implemented. When the number of an input key is fewer than 128-bit, zero bytes are padded on the right end. There are three different categories of 32-bit F-functions. Implementation requires 32-bit processors, however, in the development of compact hardware, the longer S-boxes and the three F-functions are problematic [92] [7]. It is a fast cipher with no known weaknesses.

The widely used e-mail PGP (Pretty Good Privacy) employs CAST5, which is similar to DES but with a more powerful round function. As with DES applications, it is presumed that hackers are not able to access the S-box structure [93].

Figure 2.14 shows encryption and decryption. Table 2.9 shows the F-function applied in each round [94] [95].

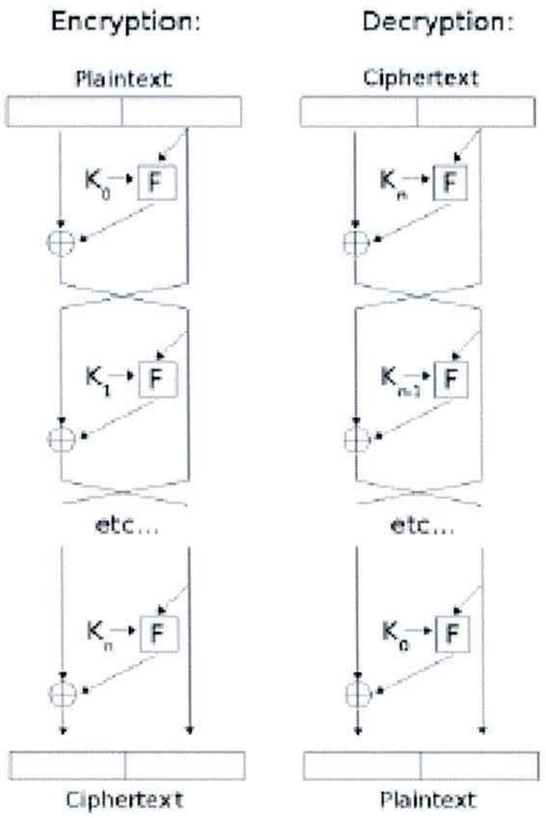


Fig. 2.14: Encryption and Decryption of CAST-128 [7].

Tab. 2.9: Types of F-function applied in each round.

Types of	Rounds				
(1)	1	4	7	10	13
(2)	2	5	8	11	14
(3)	3	6	9	12	15

2.6.12 CAST-256

CAST-256 (alternatively CAST6) is a Symmetric block cipher. It has a 128-bit block size and a 256-bit primary key, which is applied to the key schedule scheme of the algorithm. Suitable key sizes are 128, 160, 192, 224 or 256-bit. It comprises of 48 rounds or 12 quad-rounds of mixing, supplying "confusion" and "diffusion" data and key bits. There are two sets of sub-keys employed per round: K_{ri}, which is a 5-bit sub-key and is employed as a rotation key for i rounds, and a 32-bit sub-key, K_{mi}, which is employed as a masking key for i rounds. For encryption data, there are a total of 48 rounds. Types of F-function are applied in each round. The fundamental security of the algorithm is the round function. CAST-6 uses three different 32-bit round functions. The round function of the cipher is entirely dependent on the CAST-5 round function [96] [97]. The following are round functions:

1- Round Function(f1)

$$I = (K_{mi} + D) \lll K_{ri}$$

$$O = ((S1[I_a] \oplus S_2[I_b]) - S_3[I_c]) + S_4[I_d]$$

2- Round Function(f2)

$$I = ((K_{mi} \oplus D) \lll K_{ri})$$

$$O = ((S_1[I_a] - S_2[I_b]) - S_3[I_c]) \oplus S_4[I_d]$$

3- Round Function(f3)

$$I = ((K_{mi} - D) \lll K_{ri})$$

$$O = ((S_1[I_a] + S_2[I_b]) \oplus S_3[I_c]) - S_4[I_d]$$

Where:

D = 32-bit data input of the round function

$I_a \& I_b$ = the most important byte to the last important byte of I

S_i= the i^{th} called the Substitution box (S-box)

O = the 32-bit output of the round function

Each S-box is a non-linear mapping of an 8-bit input to a 32-bit output, where "+" is addition and "?" is subtraction *modulo* 2^{32}.

Furthermore, $\oplus$ is a bitwise exclusive-OR operation. Ultimately, "u « v" is the rotation of u to the left by the value shown by v [96] [98] [99].

- The plain-text is presented in four 32-bit input registers: A, B, C, and D.
- A 32-bit masking key with 5-bit rotation keys is used in each round, and the output of the 48 rounds incorporates the four 32-bit registers (A, B, C, and D) as the cipher-text.
- Decryption is the same as encryption, save that it is applied in reverse order.

2.7 A Brief Overview of Stream Ciphers

Stream ciphers are Symmetric encryption algorithms in which each plain-text digit is encrypted one at a time with the corresponding digit of the key stream, to provide a number of the cipher-text streams. The stream cipher is sometimes termed State ciphers because the encryption of a bit depends on its most recent state [77]. Stream ciphers comprise two parts: First a key stream generator, and second a mixing function. The key stream generator is the important part in stream ciphers; the mixing function is simply an XOR operation in the cipher's encryption algorithm. The output cipher is identical to the original plain-text if the key stream generator creates sequences of zeros. The major advantage of stream ciphers is that they are faster and more suitable for streaming applications. Their major disadvantage is that they are not suitable for some computing architectures [24] [100].

2.7.1 RC4

RC4 was designed by Ron Rivest in 1987. The details were published in 1996. It uses a public-key encryption method [101], which is applied to many applications, including wireless networks. RC stands for 'Rivest Cipher' or 'Ronis Code'. The algorithm applies a variable length key (1 to 256) and 50 bits to initialize a 256-bit state table. RC4 is a changeable key-size stream cipher that was designed for security and efficiency. A similar method is applied for both encryption and decryption.

One of the weaknesses of RC4 stream ciphers is the key size. RC4 is moderately uncomplicated and effective, and is applied to, among other things; Wired Equivalent Privacy(WEP), Temporal Key Integrity Protocol(TKIP), Transport Layer Security (TLS) and its predecessor Secure Sockets Layer (SSL) protocols. Fluhrer, Mantin and Shamir suggest RC4 is not at all secure in the WEP mode of operation. The majority of the observed weaknesses appear to be related to the Key-Scheduling Algorithm (KSA) which is part of RC4 [102]. In order for RC4 encryption to be broken, it requires guessing the Brute-force search, either of the whole key space or the internal state of the cipher.

RC4 encryption includes two elements: The Key-Scheduling Algorithm(KSA) and a Pseudo-Random Generation algorithm (PRGA). A variable key length is used by the KSA in order to initialize a state table S, which is an arrangement of all the 'N = 2n' possible n bit words. This permutation is employed by the $PRGA$ to create a Pseudo-Random Key-Stream. Random permutation is the basis of the RC4 algorithm. Key length K[ ], goes from 1 to 256 bytes (2048-bit) and is applied to initialise a 256-bit state vector S [ ], with elements S [0] and S [256]. S[ ] contains a permutation at all times of all 8-bit numbers in the range 0 to 255. A byte K is produced from S[ ] by choosing one of eight 255-byte entries symmetrically, for both encryption and decryption. With the generation of each K value, the entries in S [ ] are permuted again [103] [104] [105] [106] [107].

2.7.2 Grain-128

The stream cipher Grain-128 was designed in April 2007 by the researchers Hell, Johansson and Meier. Grain-128 is an alternative to the earlier Grain, with a 128-bit linear Feedback-Shift Register (LFSR) and a 128 bit Non-Linear Feedback Shift Register (NLFSR). Grain-128 is a binary additive contemporaneous with an internal state of 256-bit. The design is mainly implemented in hardware environments. Grain-128 gives a higher level of security than many other ciphers intended for use in hardware applications.

Grain ciphers comprise three main blocks: a k-bit LFSR, a k-bit NLFSR and a Non-Linear Filtering Function, where k = 80 or 128. It has an internal state size of 160, split into NLFSR and LFSR, each of size 80. The design applies a Boolean g function of 11 variables as the feedback function of the NLFSR and another Boolean g function of 5 variables of the LFSR and the NLFSR; this enables the filtering of the contents of the five fixed bits of the internal states of LFSR and the NLFSR. The use of NLFSR is recent in modern cryptography. Grain-128 uses the k-bit key K and the l-bit to determine the value of the initial vector (IV). The cipher output is a l-bit key stream sequence $z_t = 0, ..., L-1$. The state of Grain-128 consists of a 128-bit LFSR and a 128-bit NFSR.

The LFSR consists of 128-bit and its generating polynomial is *f(x)* = 0 while the NLFSR consists of 128-bit storage elements denoted as bits. Each bit $i \in 0, 1, ..., n-1$ has a related state variable xi that represents the current value of the bit i and a feedback function denoted as *fi* where *fi* $: 0, 1^n \rightarrow 0, 1$ *fi* determines how the value of i is updated. Therefore, for any $i \in 0, 1, ..., n-1,$ *fi* depends on $x_{(i+1)modn}$ and a subset of variables from the set $x_0, x_1, ..., x_i$. Moreover, $NLFSR$ is an efficient set of standards of status variables $(x_0, x_1, ..., x_{n-1})$. $LFSR$ is denoted as $s_i, s_{i+1}, ..., s_{i+k-1}$ and the $NFSR$ is denoted as $b_i, b_{i+1}, ..., b_{i+k-1}$ [108],[109]. Note that for more information about Grain-128, function *g* and *h*, and key generate, one may refer to [110]. For a detailed description of the Grain-128 stream cipher please refer to "A New Version of Grain-128 with Authentication "[111].

2.7.3 Salsa20

Salsa20 was designed in 2005 and presented to the eSTREAM and ECRYPT Stream Cipher Project. Salsa20 comprises a family of 256-bit stream ciphers [112]. Salsa20 has consistently progressed to the third round of eSTREAM. Salsa20 is faster than AES and was recommended by the designer for typical cryptographic application. It uses a stream cipher algorithm that operates in counter mode. It produces a sequence of key stream blocks, Z, that are XORed with the plain-textto create the cipher text.

Salsa20 is designed to provide high security. The core of Salsa20 is a hash function. The major function in the Salsa20 core is a quarter-round function. The functions are defined as a 4-word sequence. A quarter-round (y) is a 4-word sequence [113] [114] [115].

2.7.4 HC-128

HC Stream ciphers comprise two types: HC-128 and HC-256, with key sizes of 128 and 256-bit respectively. HC-128 is a 32-bit word based on Stream cipher. The most important part of HC-128 is the two secret tables, each of which includes 512 32-bit factors. The algorithm employs two different non-linear functions: an initialization stage and a key stream generation stage. The secret and IV keys are expanded and inserted into the two tables in the initialization stage. All table entries are replaced once the algorithm is performed and output is created. At every step of the key-stream generation phase, one table element is updated applying a non-linear function. After 1024 operations, the two tables are completely updated. In each operation,the output creates one 32-bit element.

HC-128 has a total size of 4096 bytes in the two secret tables [116]. HC-128 consists of two internal state arrays, P and Q, each one including 512 32-bit words. Thus, the key-stream is created in blocks of 512 words. Inside a block, one of the arrays is updated and the key-stream word is displayed by XORing the updated entry with the total words from the other array. Following all blocks of 512 key-stream words, the roles of the two arrays are reversed [117] [118].

At each step, the algorithm applies shift and rotation operators to get one 32-bit output. HC-128 has a total size of 4096 bytes. The IV has two major uses: it provides randomized encryption, and it helps in synchronising communication between sender and receiver [119] [120].

2.7.5 HC-256

HC-256 is a software stream cipher which was published in 2004. The key-stream Si is produced from a 256-bit secret key K and a 256-bit IV. It is a simple, very secure, software-efficient stream cipher and is freely accessible. It comprises two secret tables, P and Q, both of which have 1024 32-bit elements. The two tables can be used for the same purpose as an S-Box.

One element in each step is updated and one 32-bit output is created. HC-256 is word-orientated, with 32-bit in each word, which uses a 256-bit key with a 256-bit IV. HC-256 (and HC-128) and has not yet suffered serious attack. The HC-256 cipher applies a 256-bit key K and a 256-bit IV. The HC-256-bit comprises of two tables, each with 1024 32-bit factors. Every table updates one factor with a Non-Linear Feedback Function. Each of the 2048 steps in all the factors in the two tables are updated. At every step, the HC 256-bit produces one 32-bit output employing the 32-bit mapping, similar to that used in Blowfish. Before the output is produced Linear masking is applied

HC-256 initialization process includes expanding the secret and IV keys and inserting them into the two secret tables, P and Q, while running the cipher 4096 stage without creating output, where $\|$ is denoted as concatenation and each K_i and IV_i denoted as a 32-bit number, the K and IV are expanded into an array $W_i(0 \leq)i \leq 2559)$ [121] [120] [118].

2.7.6 VMPC

In the 2004, Fast Software Encryption workshop in Delhi, India, Bartosz Zoltak presented Variably Modified Permutation Composition (VMPC), which is a combination of elementary operations on integers and permutations. The concept behind VMPC revolves around a transformation of permutation P into permutation Q. VMPC generates a stream of 8-bit words(values). They are internal 8-bit variable(s) and a swap operation of some elements of the internal P. The Key-Scheduling Algorithm (KSA) of the VMPC transforms a cryptographic key, of a length from 120-bit to 256-bit (as well as an IV)) into a 256-element permutation$P$. The main idea of the cipher is based on an internal 256-element permutation. The VMPC cipher, together with its KSA, were designed in specific to eliminate some of the known weaknesses feature of the alleged RC4 key stream generator. The KSA of the VMPC cipher transforms a cryptographic key of length from 128-bit to 512-bit (and an (IV)) into a 256-element internal permutation (P) [122] [123]. For a detailed description of the VMPC stream cipher function, please refer to "VMPC One-Way Function "by Bartosz Zoltak

`http://www.VMPCfunction.com`

2.8 Symmetric Cryptography Modes

In Symmetric key algorithms, a block cipher works on blocks of fixed length, frequently 64 or 128-bit, but the message might be of any length. Thus, encrypting the same plain-text under the same key always produces the same output, such as described in the CBC mode below. A number of methods of operation have been applied, which permit block ciphers to supply data confidentiality of arbitrary length. Symmetric cryptography algorithms comprise of different modes. Different modes have different methods: Some relate more to the efficiency or fault tolerance, while others may relate to security levels. In addition, a block cipher mode is one that includes the use of a Symmetric key block cipher, to supply information services such as authentication or confidentiality.

Commonly, block ciphers apply ciphering modes, which combine the blocks with certain essential operations and feedback. Mode operations applied must be computationally rapid and capable, and must raise the security of the cipher-text[124]. An Initialization Vector (IV) is applied in different modes to randomise the encryption and to produce cipher-text, even if the same plain-text is encrypted multiple times the re-keying process is not slower. In CBC and CFB modes IV is never reused with the same key. If an IV is reused it would lead to a lack of information of the first block of plain-text or any other common prefix, which is shared via the two messages. The following are the five most popular modes of operations today, which will be discussed in this chapter:

- Electronic Codebook Mode (ECB);
- Cipher-Block Chaining Mode (CBC);
- Cipher Feedback Mode (CFB);
- Output Feedback Mode (OFB);
- Counter Mode (CTR).

2.8.1 Electronic Codebook Mode (ECB)

The main feature of the ECB mode is confidentiality, and it is the simplest one. In this mode the block cipher is applied directly to the algorithm. The plain-text is split into blocks of equal length and each block is encrypted individually with the same key. In encryption, the forward cipher function is applied directly and independently to every block of the plain-text. The consequent sequence of output blocks is the cipher-text. In decryption, the inverse cipher function is used directly and independently to every block of the cipher-text. The consequent sequence of output blocks is the Plain-text.

plain-text blocks:

$$M = M_1 M_2 ... M_t$$

$$C_j = E_K(M_j)$$

cipher-text blocks:

$$C = C_1, C_2, ..., C_t$$

$$M_j = D_K(C_j)$$

The advantage of this mode is that processing is independent, which means that if we have a multiple encryption process this mode can encrypt and decrypt separate blocks. There is no error propagation between blocks, meaning that if block one is received with an error in it, the receiver can decrypt block two with no problem.

A disadvantage of this mode is that when plain-texts are encrypted they produce identical cipher-texts. At present, ECB mode does not provide any integrity validation, that is modifications, intentional or not, to the cipher-text[14] [124]. This is a typical disadvantage in the operation of ECB mode that becomes obvious when dealing with lengthy bits of Plain-text[125] [126].

2.8.2 Cipher-Block Chaining Mode (CBC)

The second operation mode is called CBC. Here, before being encrypted, each plain-text is XORed with the previous cipher-text block. Therefore, each cipher-text is dependent on every block up to that phase. This means that in order to discover the plain-text of a specific block, the cipher-text, the key, and the cipher-text for the previous block must be known.

In CBC Mode, only when the IVs are the same, can encryption blocks be encrypted into the same cipher-text. Usually, IVs are chosen randomly by the dispatcher and sent to the recipient along with or before the encrypted message, hence, chaining dependency is high. Propagative error is also caused by this dependency: a small change in bit error in cipher-text block C_j will affect the ability to decipher the blocks C_j and C_{j+1}. Through its chaining dependencies, the CBC mode cannot be parallelised as it is [127] [128]:

plain-text blocks:

$$C_i = E_k(C_{i-1} \oplus P_i)C_0 = IV$$

Decryption:

cipher-text blocks

$$C = C_0C_1...C_t$$

$$P_i = C_{i-1} \oplus D_k(C_i)$$

$$M = M_1M_2...M_t$$

2.8.3 Cipher Feedback Mode (CFB)

One of the most common problems with both the ECB and CBC modes is that encryption and decryption cannot start until an entire block of 64-bit of plain-text data exists. In general, the CFB Mode operates on K-bits, which means each one of the applications produces K-bits randomly when XORing the plain-text. The most significant feature of the CFB mode is that if message m is selected as the same as the character size, this type of mode is self synchronizing, which means that if one or more m-bit characters are lost between the communicators re-synchronisation automatically occurs after 64-bit. This is important in communications environments [125] [129]. For example, using an 8-bit CFB allows one 8-bit piece of Plain-text, meaning that a single character is encrypted without needing to wait for a whole block of data to be produced [125].

2.8.4 Output Feedback mode (OFB)

The CBC and CFB modes of operation display a disadvantage in that error time corresponds to the cipher's block size. The OFB mode is similar to the CFB mode, apart from one difference, that is that in OFB mode each bit in the cipher-text is dependent on the previous one-bit or bits. This feature avoids error broadcast, which means if an error has occurred in transmission, it does not influence the bit that follows [23] [125].

2.9 Cryptanalysis

Cryptanalysis is the study of techniques, which obtains the meaning of encrypted plain-text, without access to the confidential data that is normally necessary. It is one of the more challenging research areas in the discipline of security [130]. Cryptanalysis is used to measure the security of cryptography algorithms. Characteristically, this involves finding the key that is used for decrypting the original message.

Cryptanalysis characteristically involves learning how resistant a cipher is for distinguishing attacks and to recover the key. Various research has proven that there are a number of effective attacks on block ciphers. Different attacks have been designed to manipulate the vulnerabilities of the cipher structure. Block ciphers are a significant class of cryptographic algorithms, repeatedly applied for the efficient encryption of long strings of data [131].

Differential and Linear cryptanalysis are used to break block ciphers by using statistical attacks. Differential cryptanalysis is an effective method for the analysis of block ciphers. It has been applied with success, for example on DES, and RC5 [132] [133] [134].

The following are four possible sorts of attacks that need to be considered when estimating cipher security:

- Ciphertext-Only Attacks;
- Known-Plaintext Attacks;
- Chosen-Plaintext Attacks;
- Chosen-Piphertext Attacks.

2.9.1 Ciphertext-Only Attacks

This is the simplest of the four attacks and each cipher should repel this type of attack. With Ciphertext-Only Attacks the cryptanalyst has no idea of the plain-text and can only work from the cipher-text. The attacker has some or all cipher-text and attempts to encrypt the text in order to determine the original plain-text. However, Ciphertext-Only Attacks are the main practical attacks on cryptographic systems. Ciphertext-Only is the most difficult of the cryptanalysis because the attacker knows little about the structure of the plain-text[135] [136].

2.9.2 Known-Plaintext Attacks

In this type of attack, the cryptanalyst is familiar with the cipher-text and its respective plain-text, and must find the key or vice versa. Certain encryption designs are enormously susceptible to Known-Plaintext Attacks. When Plain-texts are known, for instance, the header of an encrypted file can be guessed easily by studying the file layout, therefore, it is considered fairly easy to break. Known-Plaintext Attacks in modern ciphers are extremely common because most data and binary files transmitted over networks have some fixed segments, such as frequently used syntax elements and leading headers. When an attacker gets temporary access to the encryption and decryption, Chosen-Plaintext and Chosen-Ciphertext Attacks are possible [137].

2.9.3 Chosen-Plaintext Attacks

A Chosen-Plaintext attack is also known as Differential Cryptanalysis. With this type of attack, the cryptanalyst chooses the plain-text and puts it into a "machine" that gives the encrypted cipher-text without the key. A plain-text can have effects on encryption. Surly encryption is used to affect the plain-text and not the other way around. The cryptanalyst's challenge is to deduce the key by comparing the complete cipher-text with the unique plain-text[138].

2.9.4 Chosen-Ciphertext Attacks

In this attack, the cryptanalyst selects the cipher-text then sends it to the victim and is given in return the corresponding plain-text. The cryptanalyst has a "machine" that does the exact opposite of a Chosen-Ciphertext attack by decrypting Chosen Ciphertext with the secret key. However, certain ciphers that are resistant to Chosen-plain-text attacks can fail a Chosen-Ciphertext attack, for example, if there is a flaw in the decryption process when there is no flaw in the encryption process [138].

2.10 Differential Cryptanalysis

Differential Cryptanalysis is a significant Chosen-Plaintext Attack on block ciphers, which is one of the strongest attacks on private key ciphers [139]. It has been used with success against many block-ciphers, for example on DES. Differential Cryptanalysis uses an extremely large amount of Chosen-Plaintext in order to sift out the correct key. In 1990, Biham and Shamir developed it to attack DES algorithms [140] [66]. This cryptanalysis technique is executed in the two stages of "design" and "execution".

In the design stage, a cryptanalyst looks for the weaknesses in the cipher algorithms and uses them to discover an apposite probability differential characteristic. In the execution stage the cryptanalyst collects sufficient cipher-text pairs with suitable differential characteristics and attempts to pinpoint the effective bits of the key by way of a counting scheme. The basic security threats to block ciphers are known as Differential Cryptanalysis and Linear Cryptanalysis. Differential attacks are Chosen-Plaintext attacks that employ the statistical relation between the input difference and output difference of any two plain-text and cipher-text pairs [141].

The idea of Differential Cryptanalysis is to analyse pairs of plain-texts instead of single plain-texts. An attacker selects the unlikeness P between plain-texts (P; P*) and examines the propagation (avalanche) of the exchanges in the encryption technique. During the attack, the attacker examines and then analyses the cipher-texts pairs (C; C*), which show difference C, predicted by attacker analysis [142].

Differential Cryptanalysis is an approach that analyses the influence of specific differences in pairs of plain-text on the distinctions of the consequent pairs. These distinctions can be applied to assign characters of the possible keys and to locate the most probable key [143] [144]. It focuses on statistical analysis between the input difference and output difference of any two plain-text and cipher-text pairs.

In [145], an unusual AES incident with an uncomplicated and integrated countermeasure against the Differential Power Analysis (DPA) was submitted. Almost all the algorithms implanted in smart cards have been designed to resist high level Linear, Differential and high-order Differential attacks, while nothing has been done to ensure that they are resistant to DPA attacks. Cryptanalysis of Feistel ciphers is difficult because of their high non-linearity and autocorrelation. In addition, substitution ciphers are easily broken due to their weak encryption operation [146].

2.11 Linear Cryptanalysis

Linear Cryptanalysis is one of the most significant attacks against block ciphers [147]. Linear Cryptanalysis uses the correlation between the input and the output bits of the round. An m-round differential trail involves a chaining of variation in propagations and these are known as "differential". An m-round linear trail involves a chaining of m-round transformation correlations, which are known as the "linear" [85] [148].

The Japanese cryptographer Mitsuru Matsui designed Linear Cryptanalysis that uses sentence linear relationships between plain-text, cipher-text and key bits that display information about the key [149]. Linear cryptanalysis is more contemporary than Differential cryptanalysis. This kind of attack attempts to observe a Linear approximation to illustrate the cipher transformation. As with Differential attack, the benefit of Linear cryptanalysis is that it is a Known-Plaintext attack as opposed to a Chosen-Plaintext attack [150].

The concept is to approximate the operation of a portion of the cipher with an expression that is Linear where the Linearity refers to a mod-2 bit-wise operation ($XORoperation$) denoted by " $\oplus$ "). One expression of such can be seen below [151]:

$X_{i1} \oplus X_{i2} \oplus ... \oplus X_{in} \oplus Y_{j1} \oplus Y_{j2} \oplus ... \oplus Y_{jn} = 0$

where X_i describes the i-th bit of the input $X = [X_1, X_2, ...]$ and Y_j describes the j-th bit of the output $Y = [Y_1, Y_2, ...]$. In general this formula stands for the exclusive-OR "sum" of u input bits and v output bits. The approximation in linear cryptanalysis is to identify expressions of the form above which have sometimes a strong or low probability of occurrence.

The initial measure in a Linear cryptanalysis attack comprises of in sentence Linear approximations of the cipher with biases as high as possible. The difficulty of searching such approximations is not insignificant because of the considerable cardinality of the set of candidates [152].

Cryptanalysis systems have had a notable influence on the perceived security of various ciphers. For example, Data Encryption Standard DES can theoretically be analysed by Differential cryptanalysis using a select plain-text approach. This special cryptanalysis was founded on linear approximation of non-linear S-boxes within the bounds of the algorithm. This type of attack is then distributed for various further block ciphers [153] [154].

Summary

Cryptography involves the translation of data that is comprehensible to unwanted viewers to being incomprehensible to only desired viewers. It usually involves changing data in such a way that it is incomprehensible to everyone, and becomes comprehensible only after being manipulated by a mathematical device. Cryptography has long been important in preserving military and state secrets but in recent decades has assumed added importance because of the need to preserve financial and personal details from unwanted scrutiny, and because of the increasing power of computers both to create and break ciphers.

Encryption has two components: First, the algorithm for doing the transformation; second, a secret part of information that identifies the particular transformation (called a *key*). There are two main types of cipher: Symmetric and Asymmetric cipher algorithms. The former relies on the principle that sender and receiver of encrypted data know the key to deciphering the encrypted data. Because of this, there are security issues. First, an innocent party may reveal the key to an attacker; second, the ciphers are relatively easy to crack, unless very carefully constructed (hash functions, however, can make them difficult to crack).

The latter relies on the principle that neither sender nor receiver knows the other's keys, but they both know of another, common key, used in the encryption. This renders Asymmetric ciphers more secure than Symmetric ones. Symmetric ciphers, however, are easier to use. Ciphers can be block or stream. In block cipher, data are encrypted several parts at a time; in stream cipher, data are encrypted one datum at a time. There are many types of block and of stream ciphers. Block ciphers, for example, include Feistel ciphers and SPNs.

The many forms of block cipher vary in their performance. Old ciphers, such as the original DES developed in 1970, are now relatively insecure, and have been supplanted by AES and 3DES. Other secure ciphers (at time of writing) include Blowfish and RC6. RC2, by contrast, appears insecure, which is fast only for small files. CAST5, however, appears fast and secure.

Stream ciphers are those that process data one bit (or byte) at a time. As with block ciphers their performance varies and depends on the devices upon which

they are intended to be used. RC4 appears insecure, especially for WEP; Salsa20, by contrast, appears better than AES. Nonetheless, their is room for niche applications. Grain-128, for instance, is suitable for use with small computers with little memory and restricted power consumption.

In ECB mode, the cipher function of each block of the plain-text is applied directly and independently. The consequent sequence of output blocks is the cipher-text. The CBC mode is the combining and chaining of the plain-text blocks with the former plain-text blocks. The ECB mode was observed to have the highest confidentiality and is the simplest amongst the four researched. The CBC mode is a sequence used to encrypt a single unit or block with a cipher key applied to the whole block. The CFB mode, encrypts a single, set number of bits of plain-text, which is encrypted again and transferred directly in order to obtain the cipher-text. The OFB mode has some similarities to the CFB mode, that is, it allows encryption of differing block sizes. However, the keys are different and the output is the encryption block rather than the cipher-text. The CBC mode needs an IV which is unpredictable by the adversary, particularly if this adversary could mount a chosen plain-text attack. In CRT mode IVs need to give identical key stream. However the same is valid for OFB mode.

This chapter also considered Deferential Cryptanalysis and Linear Cryptanalysis. It defined and discussed the different types of cryptanalysis. The four types of cryptanalysis were explained. Ciphertext-Only Attack, which only allows an attacker to look at a certain number of cipher-texts; Known-Plaintext Attacks, which are attacks where an attacker can look at a number of plain-texts and the corresponding cipher-texts; Chosen-Plaintext Attacks, where an attacker can purposely select a number of plain-texts and look at the corresponding cipher-texts and Chosen-Ciphertext Attack, where an attacker can purposely select a number of cipher-texts and look at the corresponding plain-texts.

Chapter 3

Pattern Recognition

This chapter presents the background and the fundamentals of Pattern Recognition. The background and the concepts of the classification types are highlighted. Section 3.1 starts by introducing Pattern Recognition techniques. Section 3.2 explains the Multidimensional Scaling(MDS) method. While Section 3.3 addresses the statistical method, Section 3.4 explains the use of the Machine Learning method(ML). Section 3.5 explains the use of WEKA tools. Then Section 3.6 discusses classification while Section 3.6.1 explains in detail the types of Classification which have used in our study.

3.1 Introduction

In 1960s, Pattern Recognition developed significantly as a field of study. Among the different structures where Pattern Recognition has been customarily created, the Statistical approach has been studied comprehensively and widely used in practice.

PR was specifically used in interdisciplinary subjects, such as computer science, psychology and physiology among others. Automatic Recognition, Characterization, Classification, and Clustering of Patterns are significant problems in engineering and scientific teaching such as biology, psychology, medicine, marketing, computer vision etc. Pattern Recognition is a scientific concept the main purpose being to classify objects into a number of classes. It is dependent on an application of supervised or unsupervised classification. These applications can be images, types of measurements or signal waveforms that require classifying. Furthermore, it is an important part in most machine intelligence systems built for making decisions. A. Jain et al. [155] quote Watanabe who determines a pattern "as the opposite of a chaos; it is an entity, vaguely defined, that could be given a name". Some examples are: fingerprints, handwritten cursive word, human face or speech signals.

Theodoridis and Koutroumbas [156] state that: "Patten Recognition is the scientific discipline whose goal is the classification of objects into a number of categories or classes. Depending on the application, these objects can be images or signal waveforms or any type of measurements that need to be classified."

In Pattern Recognition, if the training patterns seem to form clusters PR can use classifiers which apply distance function for classification. If each class corresponds to a single prototype called the "cluster centre", minimum distance classifiers build a new pattern [157]. The use of clustering is to group together data points which are close to each other.

Among the different frameworks in which it has been traditionally formulated, the statistical approach has been intensively studied and applied in practice. In general, neural network techniques and methods imported from statistical learning theory have been receiving increasing notice. The design of a recognition system needs alert attention to the following subjects: pattern representation, definition of pattern classes, sensing environment, feature extraction and selection, cluster analysis, classifier and learning, selection of training and test samples, and performance evaluation [158].

Pattern Recognition is comprised of the following two operations [156] [159] [157]:

- Supervised Classification (for example discriminant analysis) is used when an input pattern is distinguished as a member of a predetermined class. A set of data, each compiled of measurements on a set of variables is labelled to show the class type. In most cases, the main PR problem is one of discriminating between different populations, such as discriminating between three different types of humans: (a) Tall and Thin, (b) Tall and Fat and (c) Short and Thin.
- Unsupervised Classification (for example clustering) is used when a pattern is allocated to a previously unfamiliar class. The data and features within the data need to be identified so that one group can be distinguishing from another.

Basically, the design of Pattern Recognition system consists of the following aspects:

- Pre-processing;
- Data Acquisition;

- Decision Making;
- Data Representation.

However, the best-known types of PR techniques: Statistical Classification, Template matching, Syntactic or Structural Matching, and Neural Networks. Template Matching is one of the easiest and first approaches to PR. Matching is a generic performance in PR that is used to identify the similarity between two entities such as points, curves, or shapes of the identical kind. In template matching, a template (typically, a 2D shape) or a prototype of the pattern to be recognised is present.

LM is a common method in PR which is used to identify the similarity between two objects, while, Statistical Classification is illustrated in terms of the measurements or *d* features and is observed as an idea in *d*-dimensional spaces. Syntactic or Structural PR is more relevant in taking on classifying aspects where a pattern is considered as being compiled of uncomplicated sub patterns that are themselves constituted from even easier sub patterns. Neural Networks is one of the successful methods used to estimate the function without requiring a mathematical description of how the output functionally depends on the input [155].

The recognition system consists of two methods: training and testing. Figure 3.1 shows the recognition system. In the training method, the feature extraction locates the suitable feature for representing the input patterns and classifiers which are trained to partition the feature space. Although, in the testing method, the trained classifier allocates the input pattern to one of the pattern classes being contemplated, based on the measured features [155] [160].

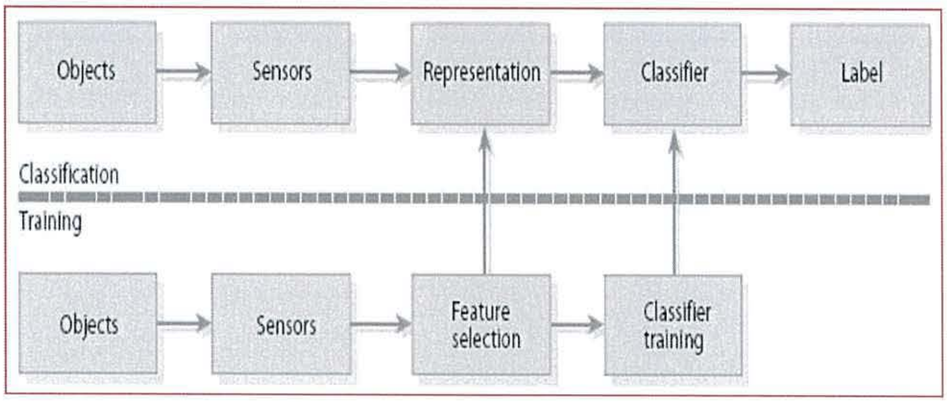


Fig. 3.1: Recognition System [8].

3.2 Multidimensional Scaling (MDS)

Nowadays, Multidimensional Scaling (MDS) has become more and more popular as a technique for both exploratory data analysis and multivariate analysis. MDS is a technique applied in order to extract a set of independent variables from a Proximity matrix, input proximities may be either similarities or dissimilarities. It creates a graphical representation of a square item-by-item Proximity matrix or Distance matrix. Applications of MDSs are established in a broad range of areas, including pattern analysis, data pre-processing visualization, cybernetics, localisation and scale development.

MDS represents a set of techniques for interpreting Similarity or Dissimilarity data. When the MDS shows a low number, it points to a strong similarity between two items, and when it shows a high number, it points to a strong dissimilarity. Similar items are represented by points that are close to each other, dissimilar items by points that are far away from each other. In general, it is applied to obtain High-dimensional data or Proximity data and then the data is reduced to an additional interpretable form, frequently, but not always represented in one (1D), two (2D) or three (3D) dimensions [161].

In 2D, the input data are contained in an item x item matrix of Proximities, which can be calculated to find either Similarity or Dissimilarity data. Proximities might be gained directly from experiments or judgements, or can be derived applying a suitable Distance Matrix from item x dimension data. The Proximities p_{ij} are mapped into distances of an m-dimension MDS configuration *X*, which means the mapping is given by a representation function $f(p_{ij})$ that determines how the proximities should be linked to the distances $d_{ij}(X)$. Assume that measures of similarity or dissimilarity, which use the general terms of proximity, p_{ij}, are given for the pairs *(i,j)* of *n* objects. Such data may be inter-correlations of, for example, test items, ratings of similarity of political candidates, or trade indices for a set of countries [162] [163].

No matter the purpose, the data should be represented in 2D or 3D when it can be plotted and inspected visually. The main reason for doing this is that one wants a graphical display of the structure of the data, particularly in a display where the information is essential, or for smoothing data [164] [165] [166] [167] .

It is possible to use Euclidean Distance and Non-Euclidean Distance to find the similarity and dissimilarity in data. Euclidean Distance can be calculated between the items, applying the variables as dimensions. In this case, we use high-dimensional Euclidean Distances as dissimilarities and we can apply MDS to reconstruct these distances in a low dimensional space. A weakness of MDS scaling is that it does not provide a clear mapping function, therefore it is not possible to position a different pattern in a map. The Euclidean Distance of points *i* and *j* in a 2D configuration *X* is computed by the following method [168]:

$$X_i = X_{ii}, X_{i2}, ..., X_{in}. \tag{3.1}$$

Or

$$X_j = X_{j2}, X_{ji2}, ..., X_{jn}. \tag{3.2}$$

where

dij denotes the dissimilarity distance, and where the distance between two points can be identified as:

$$d_{ij}(X) = \sqrt{((x_{i1} - x_{j1})^2 + ((x_{i2} - x_{j2})^2}.$$

3.3 Statistical Methods

The growth in the use of computer simulation methods for multifaceted modelling, consistent physical or built-up processes have led to the call for statistical methods that can be employed to recognise such systems. Statistics can be described as a body of analytical and computational techniques that calculate input data. The Statistical method can be used to build on scientific research in order to make it as capable and creative as possible. Numerous engineers and scientists have insufficient experience in experimental design and in the appropriate choice of statistical analyses for data that is experimentally acquired. John L. Gill (author) declares: "Statistical analysis too often has meant the manipulation of ambiguous data by means of dubious methods to solve a problem that has not been defined" [169] [170].

In the Statistical method, every pattern is identified in terms of *d* features or measurements and are seen as a point in a *d*-dimensional area. The aim is to select those features that permit Pattern Vectors belonging to diverse groups to compact regions in a *d*-dimensional feature-area. To determine the efficacy of the feature-set the data need to be separated to find a good pattern from different classes. In the Statistical Decision Theoretic Method, the decision boundaries are determined by the probability distributions of the patterns that belong to all classes, which must be learned or specified [155].

3.4 Machine Learning Based Methods (ML)

Today, Machine Learning algorithms have provided significant core function to many application domains such as computational linguistics and computational biology [171]. Machine Learning (ML) is a method of programming computers in order to be able to recognise patterns by using example data or past experience. The purpose of ML is to generate classifying expressions well enough to be understood easily by the human. PR, therefore, is the ability to be predictive and descriptive, that is gaining knowledge from data. The goal is to gain new knowledge or skills and organise the knowledge structure, so that it can make improvements on its own performance. The machine is man-made, its performance is completely stipulated by the designer and its ability cannot exceed the designer in any case. This estimation is correct when we understand that machines do not have been studying ability. Figure 3.2 shows the basic structure of ML.

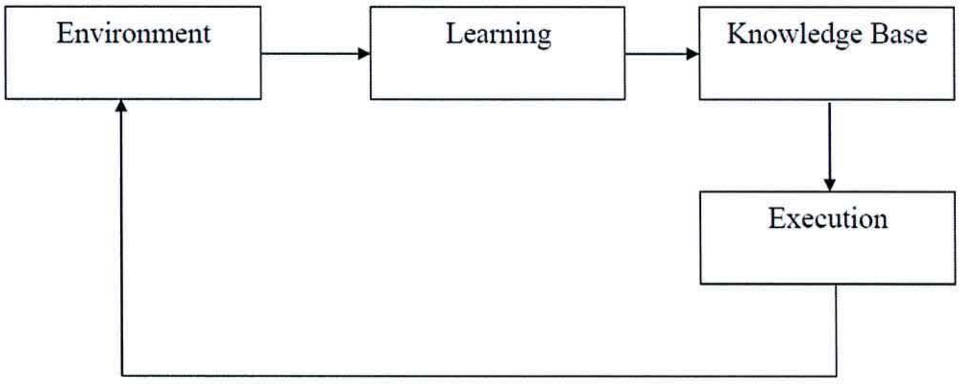


Fig. 3.2: Basic structure of learning system.

One might ask: "Why should machines have to learn?". Understanding learning in machines would further users' knowledge on how humans and animals learn. There are also significant engineering motives defined as the following [172] [173]:

- Certain tasks cannot be properly defined other than by example, that is, those that might be capable of assigning input and output pairs but not specifying a concise connection between desired inputs and outputs.
- Potentially, hidden among a massive pile of data, there are significant connections and correlations. Machine Learning methods can be applied to extract these connections (data mining)

In [174] Morgan Kaufmann state that: " The input to a machine learning scheme is a set of instances. The instances are the things are to be classified, associated, or clustered. Each instance is an individual, independent example of the concept to be learned. In addition, each one is characterized by the values of a set predetermined attributes. Each data set is represented as a matrix of instances versus attributes, which in dataset terms is a single relation".

What is an attribute? Every individual, independent instance that supplies the input to ML is characterized by its value on a fixed, predefined set of attributes or features. The instances are the rows of the tables that are shown in the simulation for different block and stream cipher algorithms, and the attributes are the columns.

3.5 WEKA Data Mining Tools

The full name of (WEKA) is "Waikato Environment for Knowledge Analysis". It is a Java class library which is implemented in numerous state-of-the-art Data Mining and Machine Learning. WEKA can be downloaded from Waikato University in New Zealand using the following URL:

`http://sourceforge.net/projects/weka/files/weka-3-7-windows-jre/3.7.5/weka-3-7-5jre.exe/download`

The new version number is (3.7.0). WEKA is freely available on the Internet and is accompanied by a new text on Data Mining. It has become a commonly applied tool for Data Mining research, it has achieved widespread acceptance within the world of academia and business, and is a free and open-source software [175] [176] [177].

WEKA is applied in ML and Data Mining used for teaching both technical internals and applications of Learning Algorithms and Machine, and as an inquiry tool for empirically evaluating new methods and development.

In [178], the authors states that the purpose of the programme is to create an up-to-date facility for improving methods of ML and looking into their implementation in main areas of the New Zealand economy. The particular aim was to produce a workbench for ML, identify the factors that contribute to the successful implementation in agriculture, and generate new systems of ML and methods of assessing their efficacy.

The workbench consists of techniques for regression, clustering, classification, association rule mining and attribute selection. Data Mining analyses and computational paradigms that enable computers to detect a structure in databases, perform forecast and prediction and comprehensively enhance their performance through interaction with data [179]. While a common work platform of Data Mining, WEKA comprises 10 Java packages Classifiers, Associations, Core, Filters, Evaluation, Visualization etc. The package of Core is the core of the WEKA system, comprising a number of key types such as instances, attribute, etc. In total WEKA has more than 1 million lines of code. The main Java packages to achieve the association rules algorithm contain two packages: (a) Associations and (b) Core are the, which include 190 Java source files [180] [181].

The main features of WEKA [176]:

1. Data preprocessing: A native file format is Attribute-Relation File Format (ARFF). WEKA can supports other formats such as instance Comma Separated Value (CSV), Matlab American Standard Code for Information Interchange (ASCII) files, and database connectivity through Java Database Connectivity (JDBC).

2. Classification: WEKA includes more than 100 classification methods. Classifiers are split into Bayesian methods (Naive Bayes, Bayesian nets, etc.), Rule-based methods (decision tables, OneR, RIPPER), Lazy methods (nearest neighbor and variants), tree learners (C4.5, Naive Bayes trees, M5), Miscellaneous and Function-based Learners methods (linear regression, SVMs, Gaussian processes). Moreover, WEKA comprises meta-classifiers such as Bagging, Boosting and Stacking; Multiple Instance classifiers and Interfaces for classifiers are applied in Groovy and Jython.

3. Clustering: Unsupervised learning is backed up by a number of clustering methods, comprising EMbased mixture models, k-means and different hierarchical clustering algorithms.

4. Selection of Attributes: The set of attributes is an essential basic for classification performance.

5. Data visualization: Data be able to inspect visually by plotting attribute values against the class, or against other values of the attribute. Because of detecting outliers and study classifier characteristics and decision boundaries, classifier output can be able to examine the training data, for specific methods, there are exact tools for visualization, such as a tree observer for a method that produces classification trees, a Bayes network viewer for automatic layout.

3.6 Classifiers

A classifier is a classification model which is assigned an unclassified instance to a predefined set of classes. The classifier is a function of *f(X)* in which the domain comprises of the training, X represents training samples $X_i = x_1, x_2, x_3, ..., x_n$ and Y classes represent the range, and the range can be one of Y classes. Furthermore, the range is called Target Attribute [182]. Data Mining can be defined as technology that enables data to analysis, explore and visualize of extremely large databases. Classification is one of the significant types of Data Mining that is a predicting modelling technique. In various real-world problems, classification techniques were used with respect to application domain and also for different research aspects. The aim of classification is to predict accurately the target class for each case in the data. The errand of classification can be done by using a number of methods and using different kinds of Classifier Algorithms. It is applied to set of data instances into proper class i.e. [183] [184] [185]. Figure 3.3 shows sample WEKA output for one of the classifier algorithm called Naive Bayesian. The classification accuracy figures that were used for the experimental in Chapter 5 and 6 is shown in the middle of the figure. The figure summary correct and incorrect instances with number of instances. The correctly classified instances are 119 which means 99.1667% and incorrectly classified instances is 1, which means 0.8333%. As well, shows that the Naive Bayes have been used 120 instances and 256 attributes. Furthermore, confusion matrix included that was used for our experimental.

```
=== Run information ===
Scheme:     weka.classifiers.bayes.NaiveBayes
Relation:   SuhailaENC
Instances:  120
Attributes: 257

        [list of attributes omitted]

Test mode:   10-fold cross-validation
=== Classifier model (full training set) ===

Naive Bayes Classifier

 Class

 Attribute        1     2     3     4

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances      119          99.1667 \%
Incorrectly Classified Instances      1           0.8333 \%
Total Number of Instances           120


=== Confusion Matrix ===

 a   b   c   d   <-- classified as

 30  0   0   0 |  a = 1

 0   29  1   0 |  b = 2

 0   0   30  0 |  c = 3

 0   0   0  30 |  d = 4
```

Fig. 3.3: Sample output information from WEKA for the Naive Bayesian classifier.

3.6.1 Classifier Methods

In our experiments detailed in later chapter, eight classifiers have been used. They are listed below. The following ML algorithms all used supervised algorithms.

Naive Bayesian Classifier (NB)

Naive Bayes (NB) is the simplest form of Bayesian Networks. It is one of the most computationally straightforward and efficient Bayesian Classifier methods. It has been used as an effective classifier for many years. The Bayesian technique is a generally applied supervised Classification Algorithm. It is a type of Pattern Recognition technique based on Bayes theorem that known has prior probability and conditional probability [186] [187] [188].

This type of classification is based on the probability density function which describes the mapping between attributes and classification attributes in the Classification Method. In addition, it is a easy structure that has the classification node as the parent node of each other nodes. Notice that no other connections are permitted in a Naïve Bayes structure. It has two advantages over other classifiers. Firstly, Naive Bayes is easy to construct, as the structure is given a priority hence no structure learning procedure is need Secondly, the classification method is particularly effective. Those types of advantages are due of that assumption, as explained in the equation below, all the features are independent of each other [189] [190].

The Naive Bayes assumption is that all the features are conditionally individual specified the class label, as the following assumption. This assumption is identifying class-conditional independence. It is performed merely of the calculations involved, it is consider to be naïve [191] [192]. The Naïve Bayes learning method is specifically appropriate for classification rather than any other difficult learning methods such as Support Vector Machine and Decision Trees. Because of its simplicity and good practicality, it is very important to enhance the performance of the Naïve Bayes [193].

A classifier is an f function that maps feature vectors $x \in X$ to output class labels $y \in 1, ..., C$ where X represents feature space. One of the Bayesian Classifier methods is Naive Bayes, which is recognised as a state-of-the-art Bayesian Classifier [194] [195] [196] [197].

It comprises two advantages over many other classifiers. (a) It is simple to construct, as the structure is granted a priori, therefore, no structured learning procedure is needed. (b) The classification method is particularly efficient. This is because of the hypothesis that each one of the features are independent of one other. It has surprisingly outperformed numerous sophisticated classifiers, in particular where the features are not powerfully correlated.

A classifier that assigns a class label to an example is constructed from a set of training examples with class labels. Presume that $A_1, A_2, ..., A_n$ are n attributes. For example E is symbolized by a vector $(a_1, a_2, ..., a_n)$ where a_i is the value of A_i. Suppose C represent the class variable that corresponds to the class, and c represents the value that C takes. Bayes Theorem can be applied to calculate the probability [198], [196], [197]:

Support Vector Machines Classifier (SVM)

In the 1990s, the Support Vector Machine (SVM) was developed by Vapink in theory and application. SVM is a method which is based on the principle of structural risk minimization and the Vapnik Chervaonenkis dimension (VC) theory [199] [200]. SVMs are supervised learning machines based on statistical learning hypothesis, which are able to be used for pattern recognition and regression.

These are functional methods for data classification based on statistical learning theory designed by V.N.Vapnik, and have been efficiently executed to various Classification and Pattern Recognition problems, such as text classification and image recognition. In [201] the authors state: "SVM is one of the popular techniques for pattern recognition and is considered to be the state-of-the-art tool for linear and non-linear classification". It is capable of solving classification problems with high dimensional feature space and small training set size.

It is a typical binary-classifier, and it has been extended for the design of multi-class SVM classifiers. Computational difficulty and classification time for the SVM classifiers using non-linear kernels depend [202] [203].

The main idea behind SVMs is to map the primary data points from the input space to a high dimensional or even infinite-dimensional feature space such that the classification problem becomes simpler in the feature space.

Moreover, the greater distance of hyperplane to the nearest training data points, the less classification error. A separate hyperplane can be formulated as an *n-dimensional* feature space for SVMs and their extensions and alternatives, frequently called Kernel-Based techniques, which have been studied widely and applied to different pattern classifications and function approximation problems [189] [204]. Since their introduction, SVMs have been proved to be successful tools for the solution of a large range of classification problems.

The SVM method applies a primary source of information, that is Kernel matrix *K(i,j)*, since *K* is Mercer's Kernel and *i,j* denote data points in the sample [205] [206], [183]. These techniques classified by building an *n-dimensional* hyperplane which splits the data into two types: the training and test data points. However, to understand the basic of SVM classification, one requires four basic concepts: (a) the separating hyperplane,(b)the maximum-margin hyperplane, (c) the soft margin and (d) the kernel function. In addition, the Classifier method consists of Statistical method, Machine Learning, and Neural Network methods [207] [197]. For more details about SVM classifier see [208].

Neural Networks Classifier (MLP)

Artificial Neural Networks have been widely applied in different fields of science and engineering. A Neural Network, in general, includes a large figure of simple processing interconnections and elements. The easy processing elements are defined as neurons, and each neuron has numerous input signals and one output signal. MLP is a network, which are comprised of generally pair or more layers of neurons and of an extra input layer. The input layer is linked by some authors as an individual network layer whereas through others it is not [209] [210]. One or two optimised the output layers and of every hidden layer during the training method depends on the weight. Neural Networks are non-linear; it is self-organizing and adaptable. In Mathematics, a transfer function established in what manner a neuron will weight its response to incoming signals and produces operation, considered the most significant part [197] [211].

Neural Networks can be applied to solve highly non-linear control problems. Processing elements have a number of internal parameters called weights. Neural Networks are considered as enormously parallel computing systems comprising a number of modest processors with numerous interconnections. The principal variation between Neural Networks and other approaches to Pattern Recognition are that these networks have the capacity to learn complicated non linear-input and output relationships, and use serial training [212].

The learning process of MLP network is based on the data samples, composed of the N-dimensional input vector *x* and the M-dimensional desired output vector *d*, named destination. When processing the vector x as input the MLP produces the vector $y(x, w)$ as an output signal, where, x represents as the vector of adapted weights. In a step by step manner, the corrective adjustment is designed to create the output $Y_k(k = 1, 2, ..., M)$ to reply d_k [213].

In [214], the author shows that many of the benchmark and researchers have been published on Neural and Statistical Classifiers. One of the widest was the Statlog project [215] in which statistical techniques, Machine Learning, and Neural Networks were compared applying a great figure of dissimilar datasets. In [216], the researcher shown that the Neural Network is considered as a significant tool for classification. The current numerous study activities in neural classification have recognized that Neural Networks shown potential alternatives to different conventional classification methods. The following aspects are the advantages of the Neural Network Classification [216] [217]:

- Neural Networks are data driven self-adaptive techniques in that they can adjust themselves to data without any explicit specification of functional or distributional forms of the underlying model.
- They are general functional approximates in that neural networks can approach any function with arbitrary accuracy.
- The Neural Network relies heavily on having adequate data for training purposes.

Instance- Based Learner Classifier (IB1 or IBL)

In 1996, the Bagging classifier was developed by Breiman [218] [185]. IBL generates classification predictions using only specific instances. As the Nearest Neighbour Classification function merely assigns classifications according to the Nearest Neighbour policy. It can determine which instances in the instance space will be classified by each of the stored instances. IBL is similar to the Nearest Neighbour Algorithm except that it normalizes its 'attributes' ranges, processes instances incrementally, and has a simple policy for tolerating missing values. Euclidean distance applied in IBL algorithms provides ranked matches between training instances and provides test instances. Equation 3.6.1 represents the similarity used within IBL algorithms [195] [219]:

$$similarity(x, y) = -\sqrt{\sum_{i=1}^{n} f(x_i, y_i)}.$$

where n: is the Instances attributes
Numeric value attributes are represented by:

$f(x_i, y_i) = (x_i - y_i)^2.$

Boolean and symbolic attributes are represented by

$f(x_i, y_i) = (x_i \neq y_i).$

Bagging Classifier (Bag)

Bagging is a statistical combining and re-sampling method used to decrease the misclassification error of a base classifier, which is based on Bootstrapping and Aggregating techniques. The Bootstrap is designed to create new data using the information from the original data. Or it is a "bootstrap" ensemble technique that produces individuals for its ensemble by training every classifier on a random redistribution of the training set. Every classifier is trained with a set of n training samples, drawn randomly with replacement from the original training set of size n, wherever n is the size of the initial training set; numerous of the initial examples may be reiterated in the consequent training set whilst others may be left out. Each one separate classifier in the ensemble is resulted in with a various random sampling of the training set. Bagging is known to be a successful in raising the accuracy of prediction of the non-constant classifiers [220] [221].

For small data sets, a Learning Algorithm is not good enough because if small changes occur in the training data set it will create very divers classifiers. Normally, the Bagging classifier improves recognition for unstable classifiers because it efficiently averages over such discontinuities. There are no persuasive simulation studies or hypothetical sources with the view that bagging will assist all unstable classifiers [222] [223].

AdaBoost Classifier (AdaBM1)

In 1995, the AdaBoost algorithm developed by Freund and Schapire, resolved numerous of the practical problems of the earlier boosting algorithms. Boosting is a family of methods, the most well-known members being AdaBoost. Boosting method aims to promote a learning accuracy of the algorithm, with it an algorithm becomes stronger. Boosting is the public method and it is one of the Boosting methods in Data Mining methods for progressing towards the accuracy of prediction. The method produces a set or ensemble of classifiers from a given data set. Every classifier is made with a various training set acquired from the primary set applying re-sampling methods, and the latest output is come by selecting. The aim behind it is obtaining an extremely accurate classifier by mixing numerous weak classifiers. All weak classifiers are needed merely to be accurate; that is, to be better than random guessing. The purpose of these ensembles is to raise up the accuracy with respect to the base [224] [199].

The classifiers in the ensemble are added one by one so that every sub-sequent classifier is trained on data that have been 'difficult' for the old ensemble members. It is machine linear method that combines weak classifier in an iterative way to produce a final physically powerful classifier via the learning procedure. Diversity is an important property of a classifier ensemble. The achievement of AdaBM1 has been explained, between others, with its diversity creating capability [225] [197] [226] [227] [228].

The input set represent as $(x_1, y_1, \ldots, x_m, y_m)$, however, every x_i refers to instance X, and every label y_i is in label set is represent as Y, $x_i \in X$ and $y_i \in Y = -1, +1$. AdaBoost names a given base learning algorithm often in a sequence of rounds $t = 1, ..., T$. On the primary objects of the algorithm is to supply a set of weights through the training set [229].

The AdaBoost Learning Algorithm is applied to boost the classification implementation of an uncomplicated learning algorithm, moreover it is an aggressive mechanism for determining a small set of right classification functions [230]. In 2001, Viola and Jones developed a modified AdaBoost algorithm successfully and applied it to face detection [231].

Rotation Forest Classifier (RoFo)

Rodriguez et al. [232] suggested this classifier based on feature extraction. The base classifier is again a Decision Trees (DT). In RoFo classifier, selected DT because they are sensitive to rotation of the feature axes, therefore the name "Forest ". In RoFo, each tree is built on a bootstrap sample form the data rotated in random way. Bootstrap is a technique for assigning measures of accuracy to sample estimations. To produce the training data for a base classifier, the feature set is randomly divided into *K* subsets and Principal Component Analysis (PCA) is used to every subset, *K* is a parameter of the algorithm. Each main component is retained in order to keep the changing information in the data. Hence, *K* axis rotations for a base classifier that take place to form the new features .

The concept of the rotation approach is to stimulate simultaneously separate accuracy and diversity within the ensemble, for each base classifier diversity is raise during the feature extraction. Accuracy is attempted by keeping each standard components and applying the all data set to train every base classifier. A absolute valuable characteristic of the proposed technique is that RoFo could be applied in conjunction with practically whatever base classifiers that are applied in the creation ensemble method

C4.5 Classifier

The C4.5 classifier is an improvement on the ID3 classifier. One of the most well-known algorithms for building decision trees is the ID3 algorithm developed by Ross Quinlan in 1979. The decision tree in the ID3 classifier builds on symbolic data. The C4.5 algorithm is able to build decision trees from a set of training data, in the same way as is built in ID3. Each element in C4.5 stipulates values for a collection of attributes and for a class [233]. All attributes may have discrete or continuous values. In addition, the unfamiliar unique value denotes unspecified values [223]. A performance measure of a decision tree over a set of cases is the classification error. This is the percentage of the misclassified cases where the predicted class differs from the actual class. The training data is a set of already-sampled classifiers called S: $S = s_1, s_2, ...$. In every sample $s_i = x_1, x_2, ...$ which is a vector and $x_1, x_2, ...$ express attributes(features) of the sample. However, training data is improved with a vector represented as C: $C = c_1, c_1, ...$ where the class is named $c_1, c_2, ...$ in each sample which belongs to the vector. Every node of the C4.5 algorithm tree selects one attribute of the data that mainly divides its set of samples into sub-sets which is enhanced in one class or another. Its criteria is the normalised information which increases the difference in entropy. Dividing the data is the outcome of selecting an attribute. The attribute with the strongest normalized information increases the selected trees to create the decision [234]. The Decision Tree is applied to classify a class value to a case that is dependent on the values of the attributes of the case [235].

A performance measure of a decision tree over a set of cases is the classification error. It is the percentage of the misclassified cases whose predicted class differs from actual class. Moreover, a decision node identifies a test over one of the attributes that is called the attribute selected at the node. However, each one has possible outcomes of the test, resulting into a child node. A Decision Tree is applied to classify a case, i.e., to assign a class value to a case depending on the values of the attributes of the case. In general, a path from the root to leaf of the decision tree can be followed based on the attribute values of the case. The class defined at the leaf, is the class predicted by the decision tree. The primary aim of the algorithm is to find relationships between values of a target or dependent attributes, which have to be a qualitative attribute with two or further values and those of a number of sets of independent attributes (numeric or qualitative) [223] [236] [237]. For a detailed description of the C4.5, refer to `http://www.cis.temple.edu/~giorgio/cis587/readings/id3-c45.html`

3.7 Encryption Classification

" Encryption Classification" is defined here to mean the process of identifying the algorithm used to encrypt some Plain-text from the encrypted output file. Today, there has been very little work done on this problem. In [238], Dileep and Sekhar state that, statistical techniques and machine learning based techniques have been applied to identify the encryption method from the encrypted files using Support Vector Machines (SVM). They represented a cipher-text by a document vector with fixed length representation and varying length representative of words in cipher-texts. The size of the cipher-text generated from a plain-text of 500 ASCII characters is 4000 bits. For generating a document vector from a cipher text considered the common dictionary based technique and the class specific dictionary based technique using five block ciphers: Data Encryption Standards (DES) with Electronic Code Book (ECB) mode, DES with cipher Block Chaining (CBC) mode, Triple DES (3DES), Blowfish, Advanced Encryption Standard (AES) and RC5 algorithms. The authors argued that a better performance was obtained for the ?xed length word and for the class speci?c dictionary based technique. However, the performance of the identi?cation of encryption technique was poor for the cipher texts generated have been used keys that are differences from the keys used in generation of the training data.

Spillman et al.[239] state that a neural network based method has been used for cryptanalysis of a Feistel type block cipher. In the last few years, there has been growing interest in the use of machine learning classifiers for data mining, Cufoglu, et al. (2009) [240] authors the performance of classifiers used to identify user profiling. The results conclude that the Naive Bayes classifier produce the best performance over user related information. Cufoglu, et al. (2008) [195] found that the Naive Bayes and IBL classifiers have the highest classification accuracy with the lowest error rate. Also they obtained simulation results evaluate against the existing works of SVM, Decision Trees (DTs) and Neural Networks(NNs).

Summary

In this chapter Pattern Recognition Algorithms have been reviewed. Two common tasks of Pattern Recognition is unsupervised and supervised classification. With the diverse frameworks in which Pattern Recognition was traditionally formulated, the statistical approach has been most intensively studied and applied in practice. This chapter discusses the use of WEKA tools, Multidimensional Scaling (MDS) and Machine Learning (ML). MDS which has become more and more popular as a method for both exploratory Data Analysis and Multivariate and Classifier. The aim of Statistical Methods is to build decision boundaries in the feature area, that separate patterns belonging to diverse classes.

Several classification have been presented, such a Naive Bayes (NB), Support Vector Machines (SVM), Neural Networks (MLP), Instance-Based Learner (IBL), Bagging (Bag), AdaBoost (AdaBM1), and C4.5 classification algorithms. In addition, the advantages and disadvantage of some classifiers have been highlighted. Boosting has been designed to improve the accuracy of any given learning algorithm. In the Naive Bayes classifier, the consequence of an attribute value is given classes, which is independent on the values of the other attributes. Support Vector Machines is a typical binary-classifier. Most research shows that the SVM classifier has became a successful tool for the solution of a large range of classification problems. Furthermore, Neural Networks is a data driven self-adaptive. In Bagging classifiers, the classifiers are created independently from each other. IBL classifiers use a simple Euclidean Distance function to supply graded matches between training instance and give test Instances. The AdaBM1 classifier uses a more refined way of sampling the original training set, where the samples are selected according to the accuracy of the previously created classifiers.

Another useful classification approach is The most Rotation Forest (RoFo) is to ensure within the ensemble and encourage simultaneously separately accuracy and finally, the most important aim of the C4.5 algorithm is to discover the relationships between values of a target or dependent attributes.

There has been very little work done on using PR techniques to identify the algorithm used to encrypt some plain-text from the encrypted output file.

Chapter 4

Creating and Analysing the Datasets

The purpose of this chapter is to create an encryption dataset to be used for the experimental evaluation in Chapter 5 and 6. A secondary purpose is to analyse the created dataset to learn more about the encrypted data (for example to determine the randomness of the encryption output). Section 4.1 introduces related work. Section 4.2 defines a random number sequence and Section 4.3 addresses motivation and the aim in this study. Section 4.4 addresses methodology used in this study. Section 4.5 explains how the datasets were generated. Section 4.6 analyses the datasets. Finally, the chapter concludes with a summary.

4.1 Randomness in Cryptographic Systems

Generally, with the rapid increase of network communication and cryptography, the use of random numbers is becoming more and more significant in secure data communication. The security of cryptographic systems depends on the irreproducible digital key streams that are generated and made unpredictable through the use of random number generators. As well, random numbers are applied for authentication protocols and key management in cryptographic system and smart cards[241] [242].

The use of random numbers is critical to cryptographic systems. In the world of protection information, often we see such statements as "protected by authentication bit 2048 " or "guaranteed 128-bit AES", describing the strength of encryption algorithms deployed in the security solution. Algorithms such as AES, RC4 and ECC have a track record of being difficult to break.

Unfortunately, what we see very rarely is a statement about the strength of the random number generator applied by a security system. However, system designers are generally more concerned with the bit generation speed and power consumption, than with the randomness of the bits generate. However, in most, if not all, cryptographic systems the quality of the random number generator directly influences how hard it is to attack the system. There are number of Symmetric cipher algorithms like DES, RC2, and RC5 that use a randomly selected encryption key. An Asymmetric algorithm such as Diffie-Hellman, RSA and DSA uses randomly generated values while generating prime numbers [243].

In the past, different mathematical models have been developed to improve the quality of random number generators. Numerous theorists have attempted to define the term 'random', but at present there are still differences in the definitions that have been put forward. However, true random number generator (TRNGs) are characterized by their output not being able to be reproduced.

Hans Freudenthal, (1987) [244] states that "it may be taken for granted that any attempt at defining disorder in a formal way will lead to a contradiction that does not mean that the notion of disorder is contradictory."

Fahad et al., (2010) [245] insist "that not all algorithms need the highest-quality random numbers, so a good GPU RNG should provide a speed quality trade-off that can be tuned for fast low-quality or slower high-quality random numbers."

Daniel et al., (2009) [246] proposed a system of Fingerprint Extraction and Random Numbers in SRAM (FERNS) that harvests statistic identity and randomness from existing volatile CMOS memory without requiring any dedicated circuitry. The random numbers that result come from runtime physically random noise and manufacture-time physically random device threshold voltage mismatches.

A random number generator is widely applied in different fields, for instance, games, information and probability theories, pattern recognition, quantum mechanics, statistics and statistical mechanics etc., particularly in cryptography and anything that is concerned with unpredictability. Commonly, random real numbers are generated from a sequence of random binary numbers [247] [248].

There are two types of random number generation: using hardware and using software. The hardware type is a true random number generator, while the software type is a pseudo random number generator. A true random number generator adopts a random natural phenomena (e.g. atmospheric or thermal noise) to generate random numbers; these random numbers are better quality than those generated by software. However, it is clearly unrealistic to configure the hardware devices for each user who has to generate random numbers, and similarly unrealistic to configure the software because it mainly uses algorithm or mathematical models to generate random numbers.

In computer programmes [249], most random numbers used are pseudo-random, that means they are produced in a predictable fashion applying a mathematical formula for many purpose, which is usually acceptable. However, the data might not be random in the way expected when applied to dice rolls and lottery drawings, for example.

4.2 Definition of a Random Number Sequence

As stated above, the term 'random' has varying definitions:

Lambalgen, V.(1987) [244] stated that: "it may be taken for granted that any attempt at defining disorder in a formal way will lead to a contradiction. This does not mean that the notion of disorder is contradictory. It is so, however, as soon as I try to formalize it".

Donald Knuth was nicknamed as the "father" of analysis of algorithms. Yet it was he who expressed that random numbers do not exist. We must ask ourselves, is 1 a random number, or a sequence independent random numbers? [250].

Several options are available, for those interested in analysing their cryptographic Random Number Generator (RNG). Following are highlights of various statistical tests that available [251].

- In Donald Knuth's book, the Art of Computer Programming, Seminumerical Algorithms, Volume 2, he describes numerous empirical tests which include the: frequency, serial, gap, poker, coupon collector's, permutation, run, maximum-of-t, collision, birthday spacings, and serial correlation. For further information, visit [251].
- Crypt-XS has been developed by Information Security Research Centre at Queensland University of Technology in Australia, which is suite of statistical tests. Crypt-XS tests consists: frequency, binary derivative, change point, runs, sequence complexity and linear complexity. For further information see [251].
- The NIST Statistical Test Suite is the result of collaborations between the Statistical Engineering Division and Computer Security Division at NIST. Statistical tests in the package are: frequency, block frequency, cumulative sums, runs, long runs, Marsaglia's rank, spectral (based on the Discrete Fourier Transform), nonoverlapping template matchings, overlapping template matchings, Maurer's universal statistical, approximate entropy (based on the work of Pincus, Singer and Kalman), random excursions (due to

Baron and Rukhin), Lempel-Ziv complexity, linear complexity, and serial. For additional information visit [251].

A random number generator (RNG) is the first type of sequence generator. An RNG applies a non-determinism source (i.e., the entropy source), along with some processing function (i.e., the entropy distillation process) to generate randomness. The entropy source in general comprises of some physical quantity, like the noise in an electrical circuit, the timing of user processes, or the quantum effects in a semiconductor. A variety of combinations of these inputs may be applied [252].

Thus for cryptographic purposes, Rukhin et al. (2001) [253] argued that the output of RNGs needs to be unpredictable. But, some physical sources (for instance date/time vectors) are quite predictable. These problems can be mitigated by combining outputs from diverse kinds of sources to use as the inputs for an RNG. On the other hand, the resulting outputs from the RNG can still be deficient when evaluated by statistical tests.

Lan et al. (2010) [254] state that using cryptographic techniques becomes a critical issue in communications, due to the increasing wireless networks and mobile applications and, the protection of information during transmission via open communication channels. Certainly, the competence of a cryptographic system relies on the key size produced by its RNG.

4.3 Motivation and Aim

The basic motivation of the work described in this chapter is to create the dataset for the experimental evaluations and so that other researchers can benefit from the dataset as well. The purpose of this study is to :

- Create an encryption dataset using the Crypto++ library.
- Analyse the dataset using various tests to determine the nature of the data such as randomness.

The following tests have been chosen to analyse the datasets:

- Frequency Test [255].
- Chi-square Test [256].
- Compression test using the Prediction by Partial Matching(PPM) algorithm [257] [258].

4.4 Methodology

In this study, we have used the Crypto++ library to generate the datasets. This section describes how this was done.

4.4.1 JavaTM Cryptography Extension (JCE)

The Crypto++ library is an open-source library that supports a vast array of cryptographic schemes. The library also uses the algorithms defined by Java Cryptography Extension(JCE).

The API page provides more information on the methods used [259] [260]. Our approach uses the Java Cryptography Extension(JCE). The JCE is a set of packages that supplies a framework and implementations for encryption, key management, key generation, and Message Authentication Code (MAC) algorithms. Supported encryption algorithms includes symmetric, asymmetric, block, and stream ciphers. The tools also support secure streams and sealed objects. JCE was designed as an extension package that includes implementation for cryptographic services. It is an application programming interface (API) that includes Symmetric block encryption, Symmetric stream encryption, Asymmetric encryption; password based encryption, key agreement, and message authentication codes. With the Java 2 SDK, v1.5, release, the JCE provider called "SunJCE" comes pre-installed and registered [261] [262].

4.4.2 Cryptographic Algorithm Benchmarking Utility

A program called the Cryptographic Algorithm Benchmarking Utility was written to create the dataset. It is written in Java and makes use primarily of the "Java Cryptography Extension (JCE)" as defined in the on line documentation [263].

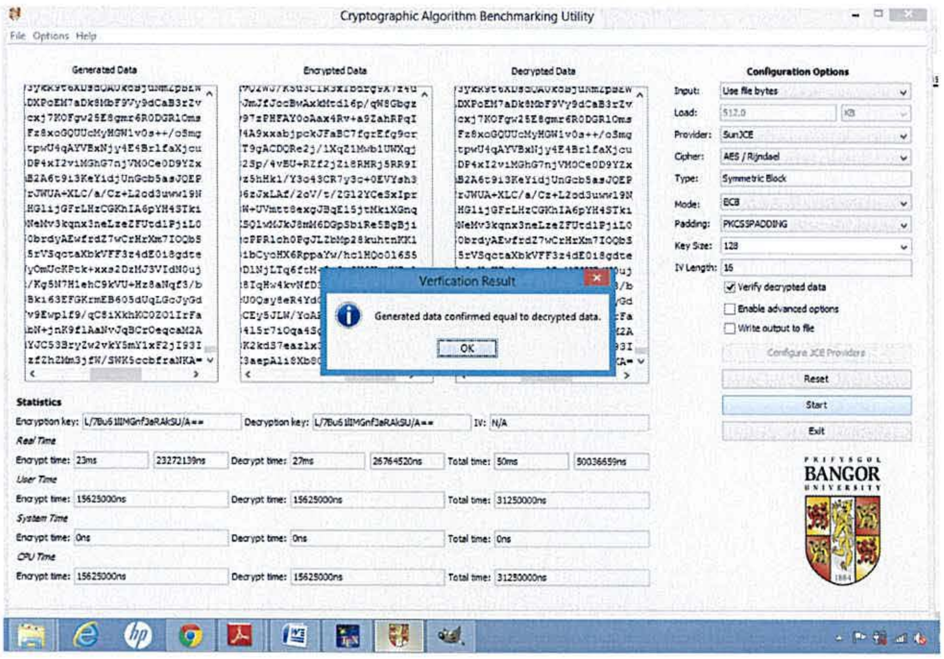
However, because this does not provide all the algorithms that were needed for our experiments, we also made use of a popular third-party library called Bouncy Castle [264].

The Graphical User Interface (GUI) for this research project was developed using the Java GUI F(see Figure 4.1). The Java code can be downloaded from: `http://pcwww.liv.ac.uk/~rwbutler/cabu.zip`.

When using the GUI, the user first uploads the data, then the library will convert it to ASCII code base 64 encoded data as shown in first panel in Figure 4.1 called

Generated data. Basically, a base 64 encoded data is a string of characters (which comprise only the $a - z$, $A - Z$, $0 - 9$, $+$ and $\%$, $\pounds$ characters) that is produced used when sending non-text data through a text-only transmission protocol. We have used base 64 encoding data because it takes a stream of characters and converts them into characters that belong to the universal ASCII code set. For more information about how base 64 does work, see [265].

The GUI allows the user to change the block and stream cipher between Symmetric and Asymmetric, and change the mode type for block ciphers. As mentioned in the literature review chapter, since some type of blocks require padding in encryption, the library also provides different padding such as PKCS5PADDNG or ISO01126PADDING. For DES and Blowfish block cipher algorithms in Electronic Code Book (ECB) and Block Chaining (CBC) mode, both are algorithms that require their input to be an exact multiple of the block size. For example, if the plain-text to be encrypted is not an exact multiple, then it needs to be padded before encrypting by adding a padding string. This is automatically done by the library. The receiving party for decrypting the data needs to know how to remove the padding in an unambiguous manner. The GUI also allows the user to save the output to a specific folder. This output is then used for the experimental evaluation.


Cryptographic Algorithm Benchmarking Utility
File
Options
Help
Configure JCE Providers
Reset
Start
Exit
PRIFYSGOL BANGOR UNIVERSITY

Generated Data

DXPoEM7aDk8MbF9Vy9dCaB3rZv
cxj7KOFgw25E8gmr6RODGR1Oms
Fz8xoGQUUcMyMGW1v0s++/o5mg
tpwU4qAYVBxNjy4E4BrlfaXjcu
DP4xI2viMGhG7njVM0Ce0D9YZx
B2A6t9i3KeYidjUnGcb5asJOEP
rJWUA+XLC/a/Cz+L2od3uww19N
HG1ijGFrLHzCGKhIA6pYH4STki
NeMv3kqnx3neLzeZFUtdiPjiL0
ObrdyAEwfrdZ7wCrHrXm7IOQb5
5rVSqctaXbkVFF3z4dE0i8gdte
yOmUcKPtk+xxs2DzMJ3VIdN0uj
/Kg5N7H1ehC9kVU+Hz8aNqf3/b
Bki63EFGKrmEB605dUqLGcJyGd
v9Ewplf9/qC8iXkhKC0Z01IrFa
bN+jnK9f1AaNvJqBCrOeqcaM2A
YJC53BryZw2vkY5mY1xF2jI93I
zfZhZMm3jfW/SWK5ccbfraNKA=

Encrypted Data

JmJfJocBwAxkMtdl6p/qN8Gbgz
97zPHFAYOoAax4Rv+a9ZahRPqI
4A9xxabjpckJFaBC7fgrEfg9cr
T9gACDQRe2j/1XqZ1Mwb1UWXqj
2Sp/4vBU+RZf2jZi8RHRj5RR9I
z5hHk1/Y3o43CR7y3c+0EVYsh3
6zJxLAf/2oV/t/ZG12YCeSxIpr
W+UVmtt8exgJBqE15jtMkiXGnq
SQ1wMJkJ8mM6DGpSbiReSBgBji
cPBR1ch0PgJLZbMp28kuhtnKKi
ibCyoHX6RppaYw/hc1HOo01655
D1NjLTq6ftM
8IqHw4kvNfD
U0Qsy8eR4Yd
CEy5JLW/YoA
415r7iOqa4S
K2kdS7eazlx
3aepAli8Xb8

Decrypted Data

DXPoEM7aDk8MbF9Vy9dCaB3rZv
cxj7KOFgw25E8gmr6RODGR1Oms
Fz8xoGQUUcMyMGW1v0s++/o5mg
tpwU4qAYVBxNjy4E4BrlfaXjcu
DP4xI2viMGhG7njVM0Ce0D9YZx
B2A6t9i3KeYidjUnGcb5asJOEP
rJWUA+XLC/a/Cz+L2od3uww19N
HG1ijGFrLHzCGKhIA6pYH4STki
NeMv3kqnx3neLzeZFUtdiPjiL0
ObrdyAEwfrdZ7wCrHrXm7IOQb5
5rVSqctaXbkVFF3z4dE0i8gdte

Verfication Result

Generated data confirmed equal to decrypted data.

OK

Configuration Options

Input: Use file bytes
Load: 512.0 KB
Provider: SunJCE
Cipher: AES / Rijndael
Type: Symmetric Block
Mode: ECB
Padding: PKCS5PADDING
Key Size: 128
IV Length: 16
[✓] Verify decrypted data
[ ] Enable advanced options
[ ] Write output to file

Statistics

Encryption key: L/7Bu61lIMGnf3eRAkSU/A== Decryption key: L/7Bu61lIMGnf3eRAkSU/A== IV: N/A

Real Time

Encrypt time: 23ms 23272139ns Decrypt time: 27ms 26764520ns Total time: 50ms 50036659ns

User Time

Encrypt time: 15625000ns Decrypt time: 15625000ns Total time: 31250000ns

System Time

Encrypt time: 0ns Decrypt time: 0ns Total time: 0ns

CPU Time

Encrypt time: 15625000ns Decrypt time: 15625000ns Total time: 31250000ns

Fig. 4.1: Graphical User Interface for the Cryptographic Algorithm Benchmarking Utility.

The JCE is extensible, meaning that new cryptographic providers can be plugged into the framework in order to provide new cryptographic implementations, Bouncy

Castle being one example which is not available out of the box. The program has been written so that new providers can easily be plugged in by providing a new .prov file in the lib directory. The files of most interest are the SunJCE.prov file (which contains the vast majority of cipher implementations provided by the Java JDK) and Bouncy Castle.prov.

Which providers are used by the application will depend upon which boxes have been selected via the Enable Advanced Options check box and the Configure JCE Providers button is clicked. Once the application has been installed, it is necessary to replace one of the standard Java policy files to allow the use of some of the greater key lengths provided by the application. Also there are a number of Sources Available Cryptographic Libraries available over [266]. For more information about the Crypto++ library 5.6.1, see [267].

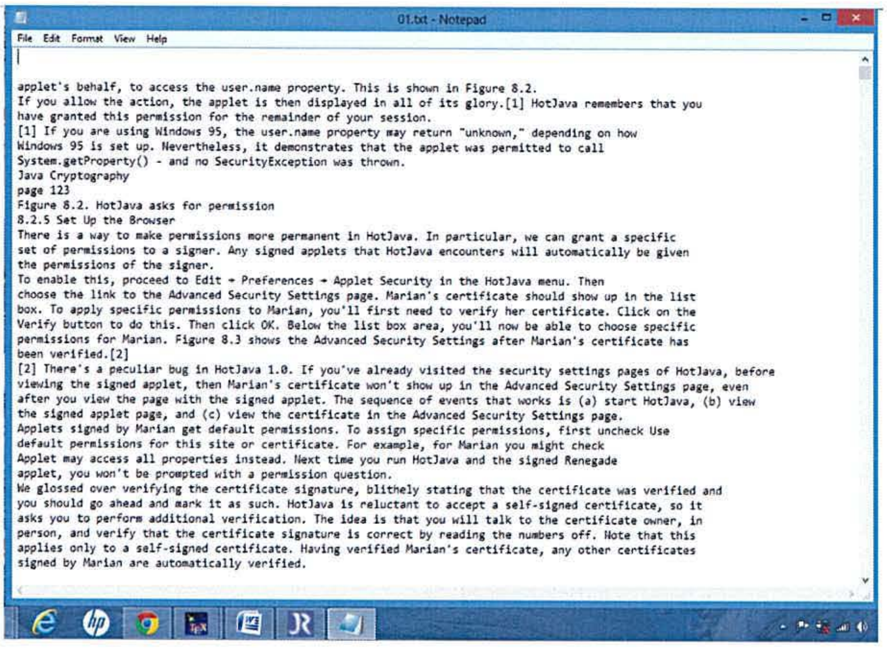
4.5 Generating the Datasets

This section explains our experimental setup. The experiment uses different algorithms of block and stream cipher, and used Crpto++ library with Bouncy Castle to encrypt and decrypt data, using 100 input files with different key sizes with text file sizes being 512KB.

A random sampling of text file data was taken from the Internet (2010) which included various types of data such as reports, papers, news, text from websites and journals. These samples ranged in sizes from 100 bytes to 10000 bytes. The files are included on the attached DVD at the end of the dissertation.

The Crypto++ library was used to encrypt the dataset. The block cipher algorithms with ECB and CBC modes were used to encrypt the data using different stream algorithms. In this experiment, the data files are divided into 8-bit and 16-bit blocks. The study included two groups of block cipher algorithms: The first group considered the following block cipher algorithms: DES (64-bit), IDEA (128-bit), AES (128, 192, 256-bit) and RC2 (42, 84, 128-bit). The second group included another seven block cipher algorithms: RC2, RC6, Blowfish, Twofish, XTA, CAST and DESede (3DES), all with the same key size (128-bit). As well, the stream cipher algorithms were used with five classes, which included five different stream ciphers: Grain 128-bit, HC 128-bit, RC4 128-bit, VMPC 128-bit and Salsa20 128-bit.

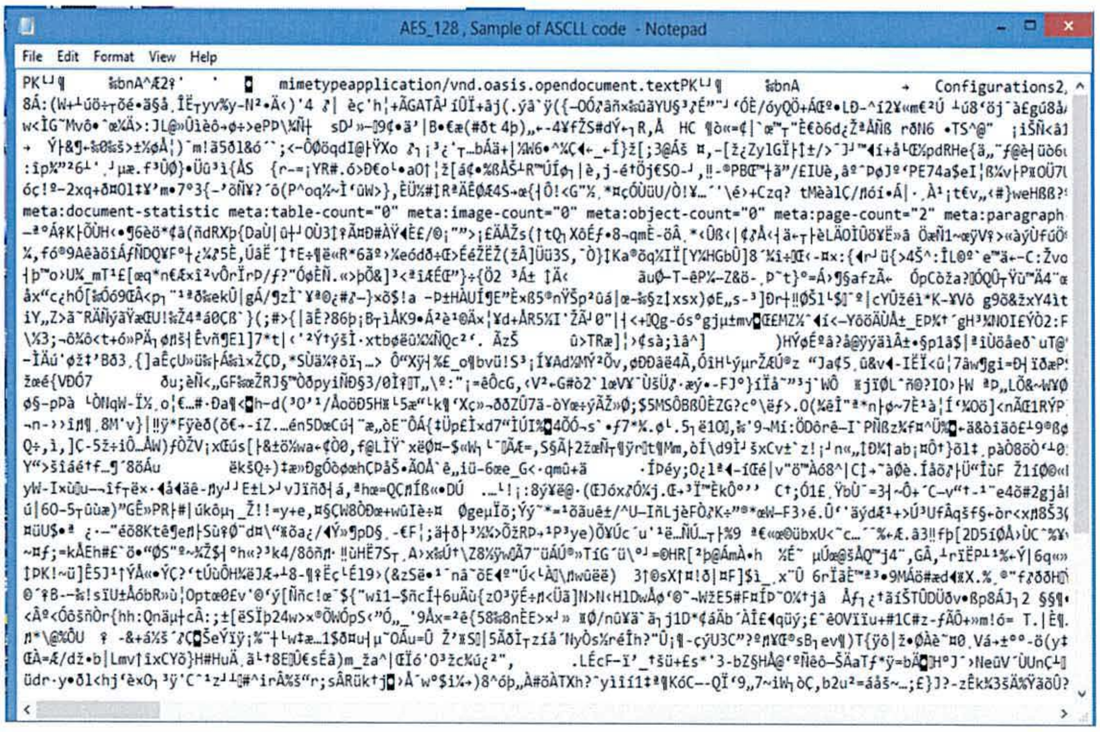
Figure 4.2 provides a sample of the AES 128-bit block cipher before encryption.



```
applet's behalf, to access the user.name property. This is shown in Figure 8.2.
If you allow the action, the applet is then displayed in all of its glory.[1] HotJava remembers that you
have granted this permission for the remainder of your session.
[1] If you are using Windows 95, the user.name property may return "unknown," depending on how
Windows 95 is set up. Nevertheless, it demonstrates that the applet was permitted to call
System.getProperty() - and no SecurityException was thrown.
Java Cryptography
page 123
Figure 8.2. HotJava asks for permission
8.2.5 Set Up the Browser
There is a way to make permissions more permanent in HotJava. In particular, we can grant a specific
set of permissions to a signer. Any signed applets that HotJava encounters will automatically be given
the permissions of the signer.
To enable this, proceed to Edit → Preferences → Applet Security in the HotJava menu. Then
choose the link to the Advanced Security Settings page. Marian's certificate should show up in the list
box. To apply specific permissions to Marian, you'll first need to verify her certificate. Click on the
Verify button to do this. Then click OK. Below the list box area, you'll now be able to choose specific
permissions for Marian. Figure 8.3 shows the Advanced Security Settings after Marian's certificate has
been verified.[2]
[2] There's a peculiar bug in HotJava 1.0. If you've already visited the security settings pages of HotJava, before
viewing the signed applet, then Marian's certificate won't show up in the Advanced Security Settings page, even
after you view the page with the signed applet. The sequence of events that works is (a) start HotJava, (b) view
the signed applet page, and (c) view the certificate in the Advanced Security Settings page.
Applets signed by Marian get default permissions. To assign specific permissions, first uncheck Use
default permissions for this site or certificate. For example, for Marian you might check
Applet may access all properties instead. Next time you run HotJava and the signed Renegade
applet, you won't be prompted with a permission question.
We glossed over verifying the certificate signature, blithely stating that the certificate was verified and
you should go ahead and mark it as such. HotJava is reluctant to accept a self-signed certificate, so it
asks you to perform additional verification. The idea is that you will talk to the certificate owner, in
person, and verify that the certificate signature is correct by reading the numbers off. Note that this
applies only to a self-signed certificate. Having verified Marian's certificate, any other certificates
signed by Marian are automatically verified.
```

Fig. 4.2: Sample of AES block cipher before being encrypted.

Figure 4.3 provides a sample result of ASCII output for AES 128-bit block cipher algorithm after being encrypted.



```
PK└┘¶      ‰bnA^Æ2‽'   '   ◘   mimetypeapplication/vnd.oasis.opendocument.textPK└┘¶      ‰bnA        →   Configurations2,
8Á:(W+┴úö÷┬õé•ä§å¸ÎË┬yv%y-N²•Ä‹)'4 ♪| èç'h¦+ÃGATÂ┘íÛÏ+âj(.ýâ`ÿ({-OÓ♪ãñx‰ûãYU§³♪É"┘‘ÓÈ/óyQÖ+ÁŒº•LÐ-^í2¥«m€²Ú ┴ú8‘öj¯â£gû8â
w<ÌG˜Mvô•¯œ¼Ä>:JL@»Ûièô÷ø÷>ePÞ\¼Ñ╪   sD┘»–▯9¢•ä'|B•€æ(#ðt 4þ)„←-4¥fŽS#dÝ←┐R,Å   HC ¶ò«=¢|`œ"┬"È€ò6d¿Žªåñß rðN6 •TS^@"   ¡ìŠÑ<â1
→   Ý├&¶÷‰Ø‰š>±%øÅ¦)¯m!ã5ð1&ó``;<-ÔØöqdI@├ŸXo ♪┐¡³¿`┬…bÅä+|%W6•^%Ç◄←_←Í}ž[;3@Áš ¤,-[ž¿ZylGÏ├↨±/>¯J┘™◄í+å└Œ%pdRHe{ä„¯ƒ@è┤üò6
:îp%"²6┴'¸┘µæ.f³ÛØ}•Üû³ì{ÅS  {r-=¡YR#.ó>Ð€o└•aO↑¦ž[á¢•%ßÅŠ┴R™ÚÍø┐|è,j-é†Öj€SO-┘,‼-®PBŒ"╪ã"/£IUè,âº¯ÞøJº‘PE74a$eI¦ß%v├P¤OÜ7
óç!º-2xq+ð¤Ol‡¥'m•7º3{-'õÑ¥?´ö(P^oq%~Ì‘ûW>},ÈÜ%#↨RªÄÊØÆ4S→œ{┤Ô!<G"%¸*¤çÓÙüU/Ò!¥…´'\é>+Czq? tMèàlÇ/ñói•Á|·¸À¹¡t€v„‹#}weHßß?
meta:document-statistic meta:table-count="0" meta:image-count="0" meta:object-count="0" meta:page-count="2" meta:paragraph
–ª°Á‽K├ÕÜH‹•¶6èö*¢â(ñdRXþ{DaÙ|û┼┘OÙ3↨‽Ã¤Ð#ÀŸ◄È£/®¡""›¡£ÄÅŽs(↑tQ┐XõÉƒ•8¬qmÈ-öÂ¸*‹Ûß‹|¢♪Å‹┤ä←┬├èLÄOÌÛö¥Ë»ã Õæ Ñ1~œÿV‽>«àýÙfúÖ
%,fó®9AêàöìÁƒÑDQ¥F°╪¿%♪5È,ÚãË´↨†E÷¶ë«R*6ãº›%eódð÷Œ>ÉéŽÈŽ(žÂ]Üü3S,¯Ò}↨Ka®õq%IÏ[Y%HGbÛ]8´%î÷▯Œ‹-¤x:{◄r┘ü{>4Š^:ÎLΘº`e™ä←-C:Žvo
┤þ™o›U%_mT¹£[œq*n€Æxî²vÕrÏrÞ/ƒ?"ÓøÈÑ.«>þÕ&]³<ªìÆÉŒ"}÷{Ö2 ³Á± ↨Ä‹        ãuØ-T-êP%-Z&ö-¸Þ˜t}°=Á›¶§afzÃ←  ÓpCòža?▯ÓQÛ┬Ÿü™Ä4¨œ
åx"c¿hÔ[‰Ó69ŒÂ<p┐¨¹ªð‰ekÛ|gÁ/¶zÌ`¥ª©¿#♪-}xõ$!a -Þ±HÀUÍ¶E"Èx85®nŸŠp²ûá|œ-‰§z↨xsx)øE„s-³]Ðr╪‼ØŠl└$▯¯º|cŸÜžéì*K-¥Vô g9õ&žxY4ìt
iY„Z>ã˜RÃÑÿãŸæŒU!‰Ž4ªá0Çß`}(;#>{|ãÊ?86þ¡B┬ìÅK9•Á²è¹©Äx¦¥d+ÅR5%I´ŽÃ┘0"|┤<+▯Qg-ós°gjµ±mv◘ŒEMZ%¯◄í<–YõöÄÙÅ±_EÞ%†´gH³%NOI£ÝÒ2:F
\%3;¬ô%ô<t+ó»PÄ┐øñš┤Êvñ¶El]7*t|‹'²Ý†ýšÌ·xtbøëû%%ÑQc²‘. ÃzŠ        û>TRæ]¦›¢sà;ìâ^]          )HÝøÉºâ?â@ÿÿàìÂ±•§p1â$|ªìÙöåeð`uT@
-ÌÄú'øž‡'Bð3¸{]aÊçU»ü‰├Á‰ì×ŽÇD,*SÙä%‽öì┐…> Ô"Xÿ┤%£_o¶bvü!S³¡Í¥Ad%MÝ²Õv,øÐÐâë4Ã,ÕìH└ýµrŽÆÜºz "Ja¢5¸û&v◄-IËÌ<û¦7âw¶gi=Ð┤ìðæP
žœé{VÐÓ7        ðu;èÑ<„GF‰æŽRJ§™ÒðpyìÑÐ§3/0Ì‽▯T„\º:"¡=êÔcG,‹V²←G#ò2`lœV¥´ÙšÜ♪·æý•-FJ°}íÏâ¯"³j`WÔ  ¤jÏØL´ñ©?IO›├W ªÞ„LÕ&~WÝØ
ø§-pÞà └ÒNqW-Ï%¸o¦€…#·Ða¶<◘h-d(³O'¹/ÅoöÐ5H¤└5æ"└k¶‘Xç»¬ðõZÛ7ã-òYæ÷ýÃŽ»Ø;$5MSÔBßÛÈZG?c°\ëƒ>.O(%êÏ"ª*n├ø~7È¹à¦Í‘%Oö]<nÃŒ1RÝP
¬n-››îñ¶¸8M'v}|‼ÿ*Fÿèð(õ€←-íZ.…én5DæCú┤¯æ„òE¨ÔÁ{‡Üp£Ìxd7"ÌÚI%◘4ÕÓ¬s`•ƒ7*%.ø└.5┐ë10▯,‰'9¬Mí:ÖDôrê–I`PÑßz%f¤^Ü%◘←ä&òìãöE┴9®ßø
Q÷,ì,]Ç-5ž÷ìÔ…ÅW)ƒÖŽV¡xŒús[├&±ö%wa←¢Ò0,f@LÌŸ`xëØ¤–$«W┐└¯▯Ã£=,S§Ã├2žæÑ┬¶ÿr▯t¶Mm,òÍ\d9Ì┘šxCv±`z!¡┘n«„↨Ð%↑ab¡¤Ô†}ðl‡¸pàÒ8õÒ‘┴0:
Y">šîáé†f…¶´8õÁu        ëkšQ÷)‡æ»ÐgÓòøæhÇÞåŠ•ÃOÅ`ê„ìü-6œ_G<·qmû+ä       ·ÌÞéy;O¿1ª◄-íŒé|v"ö™Àó8^|C↨→¯àØè.Ìåõ♪├Ü"ÌùF Ž1íØ©«l
yW-Ixü▯u–¬îf┬ëx·◄â◄äê-ñy┘┘E±L>┘vJïñð┤á,ªhæ=QÇñÍß«•DÚ  .…└!¡:8ý¥ë@·(ŒJóx♪Ó%j.Œ→³Ï™ÈkÔ°'' C†;Ó1£¸ŸbÙ¯=3┤~Ô+´C-v"†-¹¨e4õ#2gjàl
ú|6O-5┬ûùæ)"GÊ»PR├#|úkôµ┐_Ž!!=y+e,¤§ÇW8ÒÐæ+wûIè÷¤  ØgeµÏõ;Ýý˜*=¹õãuê±/^U–IñLjèFÒ♪K÷"®*œW–F3›é.Û‘'äýdÆ¹+>Ú³UfÅqšf§←òr<xñ8Š3
¤üU$•ª  ¿·–"éõ8Ktê¶eñ├Sù‽Ø¯d¤\"%õa¿/◄Ý»¶pD§¸-€F¦;ä╪ð├³%%>ÕžRÞ→¹P³ye)Õ¥Úc´u'¹ë…ÑÚ…┬├%9 ª€«œ©übxU<˜c…¯¯%←Æ.ã3‼fþ[2D5íØÅ›ÙC˜%¥
~¤f;=kÅEh#£`ö•"ØS"º~%Ž$┤°h«?³k4/8õññ· ‼ùHË7S┬¸A›x‰Ú†\Z8%ÿwÃ7¨üÁÜ®»TíG´ü\º┘=©HR[²þ@ÁmÀ•h  %É˜ µÜæ@šÅQ™j4¨,GÂ,┴rìËP┴¹%←Ý|6q«»
↨ÞK!~ü]ÊSJ¹↑ÝÅ«•ÝÇ?‘tÚùÕH%ëJÆ→┴8-¶‽Ëç└É19›(&zSë•¹¯nã¯õE◄º"Ú<└À▯\ñwûëë)  3↑©sX↑¤!ð|¤F]$ì_¸x¨Û 6rÏãÈ™ª³•9MÁö#æd◄¤X.%¸®"ƒ♪õðH
©´‽B-–‰!sÏU±ÅóbR»ù¦Optæ©Ev´®‘ý[Ññc!œ¯${"wî1-$ñcÍ╪6uÄù{zO³ýÉ÷ñ<Üä]N>N<H1DwÅø‘©˜¬WžE5#F¤ÍÞ¯O%†jâ  Åf┐¿†ãíŠTÛDÜðv•ßp8ÁJ┐2 §§¶•
<Ẫº<ÓõšñÒr{hh:Qnäµ†cÂ:;±[ëSÏþ24w>x®ÕWÓpS‹"Ó„_'9Åx=²ê{58‰8nÈE>x┘» ¤Ø/nû¥ã`ã┐j1D*¢áÃb´ÀÎ£◄qüý;£¯êOVÏïu+#1C#z-ƒÃÕ+»m!ó= T.|È¶.
ñ*\@%ÔU  ‽ -&+á%š´♪Ç◘ŠeŸïÿ;%¨╪└w‡æ…1$ð¤u┤µ¯OÅu=Û Ž'¤S▯|5ÃðÌ┬zíâ´NyÒs%réÎh?"Û¡¶-çýU3C"?ºñ¥Œ®sB┐ev¶)T{ÿö|ž•ØÀè˜¤0¸Vá+±°°-ö(y‡
ŒÀ=Æ/dž•b|Lmv↑ìxCYõ}H#HuÄ¸ã└†8E▯Û€sÉá)m_ža^|ŒÏó'O³žc%ú¿²",        .LÉcF-ï'_†šü+£s*'3-bZ§HÅ@‘ºÑeô–ŠÄaTƒ*ÿ=bÃ▯H°J`›NeûV´ÙUnÇ┴▯
üdr·y•ðl<hj‘èxO┐³ÿ'C^¹z┘┴▯#^irÂ%š"r;sÂRük†j◘›Å¯w°$ì%→)8^óþ„À#õÀTXh?¨yìíí1‡ª¶KóC--QÏ‘9„7~iW┐òÇ,b2u²=áåš~…;£}J?-zÊk%3šÃ%Ÿãõû?
```

AES_128 , Sample of ASCLL code - Notepad
File Edit Format View Help

Fig. 4.3: Sample of AES block cipher encrypted output in ASCII.

Initially, we have selected 100 data files randomly from various internet sources such as news paper, website, e-books, journals, articles, documents and reports and the overall total file size is 52,477,677 bytes. Table 4.1 describes these files and form what we have called the Bangor Sources Files Corpus (BSFC). These 100 source text files from the data files were encrypted and used in the experiment evaluation performed in Chapters 4 and 6. These files form what we call the Bangor Encryption Classification Corpus (BECC) as shown in Table 4.2.

There are many encryption algorithms that can be used to encrypt these source files. Figure 4.4, 4.5 and 4.6 show the algorithms that were used in our experiments. As well as these algorithms, many of them have various numbers of keys and other parameters. Different variations were used to encrypt the files to produce data points as training /testing data for WEKA (as explained in Section 5.2.1). The histograms of the encrypted files using both 8-bit and 16-bit encodings were generated (refer to Appendix B), then submitted to WEKA for classification (see Section A.1).

Tab. 4.1: Bangor Sources Files Corpus (BSFC).

File names	01_sources to 100_sources
Example of sources	News paper, Papers, Websites, Reports, E-Books, Journals, Documents, Article
Total file size (bytes)	52,477,677

Tab. 4.2: Bangor Encryption Classification Corpus (BECC).

File names	01_encrypted to 100_encrypted
Block algorithms/Key sizes (1)	AES(128, 192,256), DES 64, IDEA 128 and RC2 (128, 84 and 42)
Block algorithms/ Key sizes (2)	RC2, RC6, Blowfish, XTA, CAST and DESede (128-bit)
Stream algorithms/Key sizes	Grain, HC, RC4, Salsa20 and VMPC (128-bit)
Total file size (bytes)	209,916,600

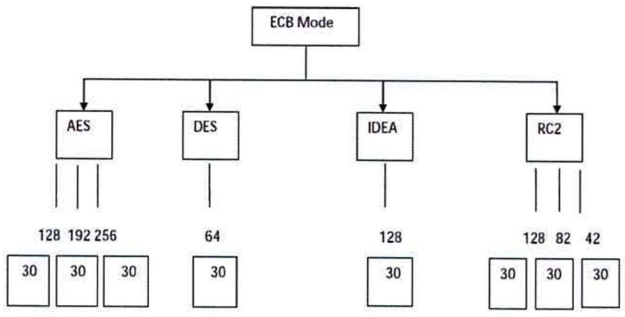


Fig. 4.4: Diagram of block cipher algorithms with 240 variation used to produce the data points used in the classification experiments.

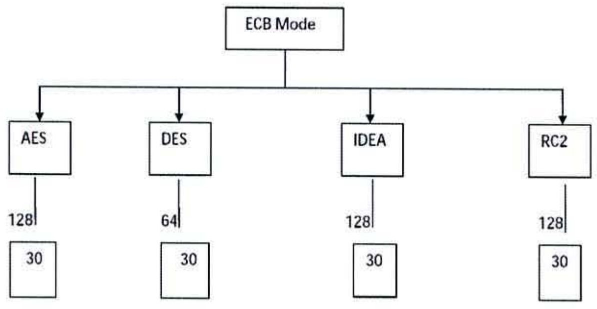


Fig. 4.5: Diagram of block cipher algorithms with 120 variation used to produce the data points used in the classification experiments.

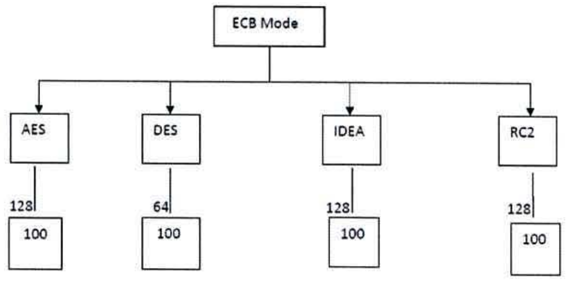


Fig. 4.6: Diagram of block cipher algorithms with 400 variation used to produce the data points used in the classification experiments.

4.6 Analysis of the Dataset

This section provides the analysis of the dataset. To validate the proposed approach, different tests have been applied to the block and stream cipher algorithms. The tests have been implemented in Java. Separate Java applications for the tests have been implemented with a main class and a file reader for each block and stream cipher algorithm so the random number file could simply be copied and pasted into the text file to generate the outcome of these tests. The Chi-square test χ^2 is used to determine whether the datasets they are random or not. PPM is also used as anther method to analyse the data. If the data is not compressible, this provides further evidence of the possible randomness of the data.

4.6.1 Frequency Test

First, the Frequency test [255] is applied to the dataset. This requires calculations of how often a value occurs and seeing if the frequencies are uniform. In this case, we have 30 datasets in each folder for each block and stream cipher algorithm. The Frequencies of each character are tabulated for all characters in the ASCII 0-255 range. When the program has been executed, the output console will print out how many times each character has occurred, the execution time in milliseconds (ms), how much memory is available and the total number of occurrences.

Below shows the Java code that was used to read.

```
//import java.io.DataInputStream;
import java.io.EOFException;
import java.io.RandomAccessFile;
public class Testread
{
  public Testread () {
    System.out.println ("testread called");
        //private Scanner scanner;
   // Scanner to read from the file
  }
  public static void main (String [] Args)
  {
    boolean EOFreached = false;
    int symbol = 0;
    try{
 RandomAccessFile file = new RandomAccessFile
 ("G:\\Testread\\AES128.txt", "r");

        // Read the input file, one symbol at a time:
        while(EOFreached==false){
                // get next symbol
                symbol = file.readUnsignedByte();
                System.out.println(symbol);
            }
        }
        catch (EOFException e){
            //EOFreached = true;
        }
        catch(Exception e){
            System.out.println(e.getMessage());
        }
    }
}
```

Below, is the code for defining the main classes for the Frequency test.

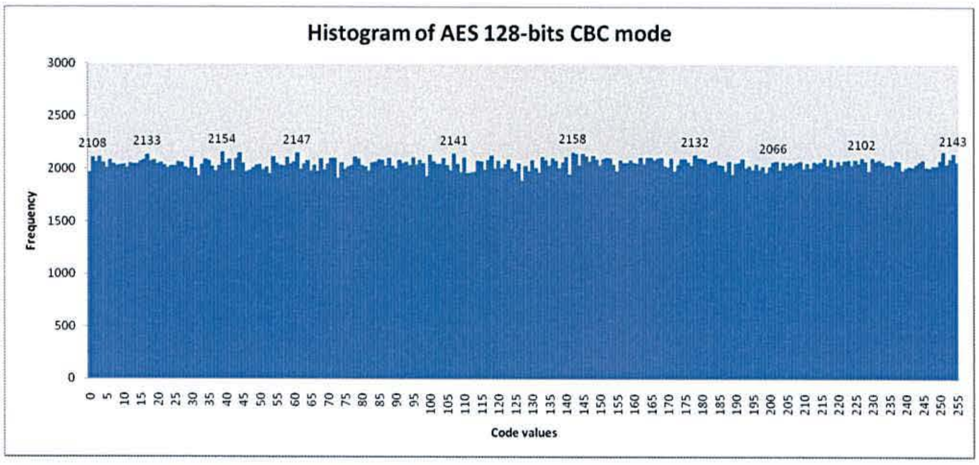
```
public class FreqTest {

    public static void main(String[] args) {
        FileReader fileReader = new FileReader();
        fileReader.loadWordFrequency();
    }

}
```

The code below provides the Frequency FileReader class, which is used for acquiring the statistics. The method HashMap is applied to load all the encrypted text from the file and reads it one by one. It checks that each encrypt character exists in the HashMap, if not then adds it to the HashMap, otherwise it takes the current frequency and increments it by one and puts it back in to the HashMap. The method concludes by showing how often the characters appear. The program will not complete until all characters have been accumulated.

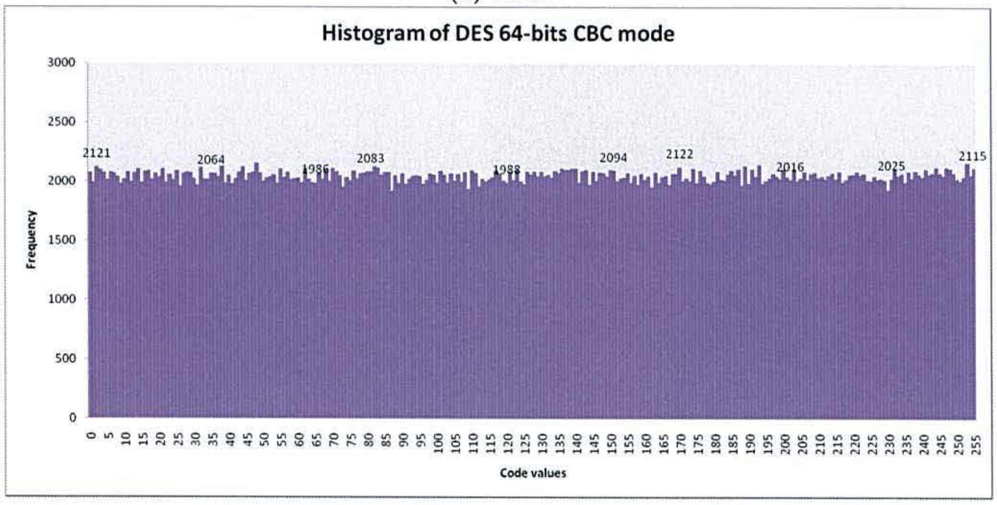
```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.HashMap;
import java.util.NoSuchElementException;
import java.util.Scanner;
import java.util.StringTokenizer;
import java.util.TreeSet;

public class FileReader {
        String fileName = "output.txt";
        private Scanner scanner; // Scanner to read from the file
        private HashMap<Integer, Integer> wordsFrequence = new HashMap<Integer, Integer>();

        // HashMap to store the number and frequency

        /**
         * Method to load all the numbers from the file and read each one by one.
         * For each number check it exists in the HashMap and if not add it,
     * otherwise take the current frequency and increase it by one and put it
     * back in the HashMap
     */

    public void loadWordFrequency() {
        Runtime runtime = Runtime.getRuntime();
        // Gets the current time in ms – so we can record how long the run took
        long startTime = System.currentTimeMillis();
        try {
            scanner = new Scanner(new File(fileName));
            String line;
            while ((line = scanner.nextLine()) != null) {
            // Keep looping until you have reached the end of the line
                StringTokenizer st = new StringTokenizer(line, ",");
                while (st.hasMoreTokens()) {
                // Read each number...

                    Integer number = Integer.parseInt(st.nextElement()
                                .toString());
                                // ...gets the current frequency from the HashMap
                    Integer frequeny = wordsFrequence.get(number);
                    // If there is nothing then its the first time its being
                    // processed
                    if (frequeny == null) {
                    // Hence add it to the HashMap with a frequency of 0
                        wordsFrequence.put(number, 0);
                    } else {
                    // else increse the frequency by one and add it back to
                    // the HashMap
                        frequeny++;
                        wordsFrequence.put(number, frequeny);
                    }
                }
            }

        } catch (FileNotFoundException ex) {
            System.out.println("Error: " + ex.toString());
        } catch (NoSuchElementException e) {

        }
        // Get the current time
        long endDate = System.currentTimeMillis();

        System.out.println("Occurances:");
        TreeSet<Integer> treeSet = new TreeSet<>(wordsFrequence.keySet());
        int totalCount = 0;
        // Loop around the tree set getting all the values and frequency and
        // display it
        for (Integer key : treeSet) {
            System.out.println("Number off " + key + " occur "
                        + wordsFrequence.get(key));
            totalCount += wordsFrequence.get(key);
        }
        System.out.println("Total number of occurence: " + totalCount);
    }
}
```

4.6.2 Producing Histograms

This section shows the results of the histogram data produced by the Frequency test program for different block cipher algorithms. These comprise: AES 128-bit, DEs 64-bit, IDEA 128-bit and RC2 128-bit with CBC mode. Figure 4.7 shows the histogram for AES and DES. Figure 4.8 shows the histogram of IDEA and RC2 algorithms. The results show that all the datasets are uniform, due to the random nature of the dataset.

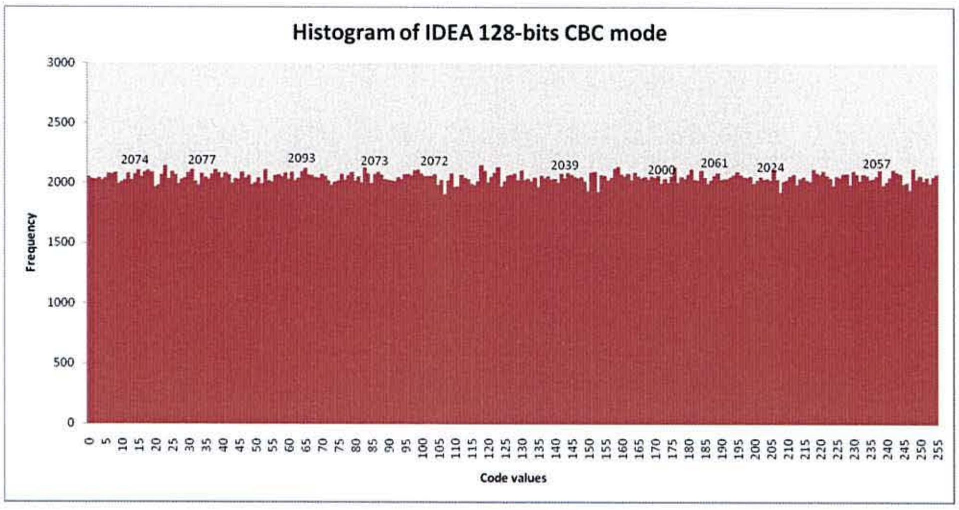


(a) AES

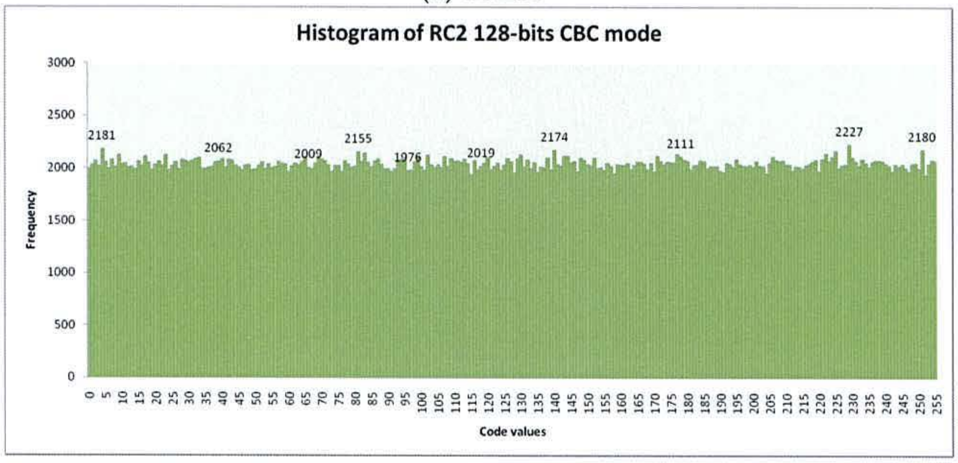


(b) DES

Fig. 4.7: Histograms of AES and DES block cipher algorithms with CBC mode.



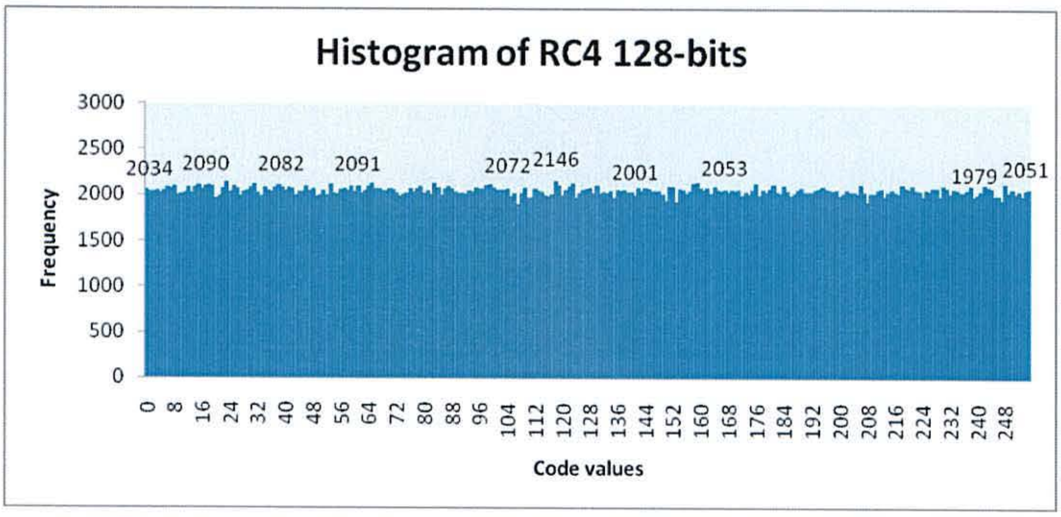
(a) IDEA



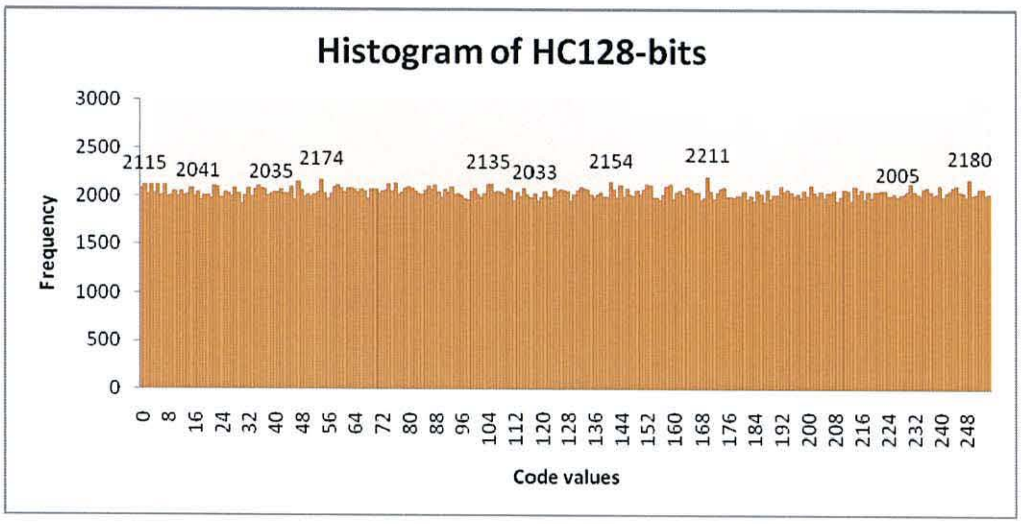
(b) RC2

Fig. 4.8: Histograms of IDEA and RC2 block cipher algorithms with CBC mode.

Figure 4.9 shows the histogram for RC4 and HC 128-bit stream cipher algorithms. Both are clearly uniform, also due to the random nature of the dataset.



(a) RC4



(b) HC128

Fig. 4.9: Histograms of RC4 and HC128 stream cipher algorithms.

4.6.3 Chi-Square Test (χ^2)

The Chi-square test [268] is a theoretical or mathematical distribution which has wide applicability in statistical work. The symbol χ^2 is used to denote the distribution. It is one of the most popular hypothesis tests and it is widely used in biology, economics, cryptography and other fields. For instance, one of cryptographic applications is the testing of random number generators and block ciphers' suitability as random number generators [269].

There are two fundamental types of Chi-square analysis: first, Goodness of Fit Test, applied with two nominal variables; and the second, Test of Independence, applied with two nominal variables. Both types apply the same formula. Chi-square χ^2 procedures measures differences between the statistically expected result (E_i) and actual result observed (O_i) of the table frequencies of nominal variables to see if there is a statistically significant difference, where O_i represent the observed value and E_i is represent the expected value for each cell and n represent sample sizes. The following shows the equation for calculating χ^2 [256]:

$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)}{E_i}.$$

The degrees of freedom are: *df*= *(R-1) (C-1)*

where: *df* represent the degrees of freedom, *R* represents the number of rows in the table and *C* represents the number of columns in the table. For any χ^2 distribution, the *df* is the number of independent free choices that can be made in allocating values to the expected frequencies.

As it was shown by Ryabko et al.[269], in some applications the number of categories (and, consequently, the number of degrees of freedom of χ^2 distribution) is very large, thus, the sample size also has to be large. Therefore, in such cases, performing the χ^2 distribution needs extra time. Furthermore, it is frequently difficult to obtain such large samples and χ^2 may not be used.

In practice, statistical testing is used to gather evidence to show that a generator indeed produces numbers that appear to be random. For our problem, the Chi-square test has been used to investigate the randomness of different block and stream cipher algorithms.

Once we have calculated the value of χ^2 and determined the degrees of freedom, we can look up the probability in standard statistic tables to determine whether the differences in the different block and stream cipher algorithms are due to chance.

In general, the test follows the standard statistic hypothesis test [270]. The null hypothesis H_0 is presented as the sequence of random datasets and the alternative hypothesis H_1 is presented as the sequence of non-random datasets. For the significance levels *p-values*, these are set as 0.01 or 0.05. For instance, the sequence is considered to be random with the confidence of 95% or 99% respectively. If the $p - value$ is higher then H_0 is accepted, otherwise, the H_0 is rejected.

Table 4.3 shows the dataset results for the AES 128-bit, DES 64-bit, IDEA 128-bit and RC2 block cipher algorithms. Since there are 256 categories, the degrees of freedom equal (*df*) 256 - 1 = 255. Note that the degree of freedoms for all block cipher algorithms are the same.

Samples were chosen for the BECC dataset and tested using the χ^2 test. For the AES algorithm the probability value was p = 0.091 and the Chi-square value was 285.649. For the DES 64-bit block cipher algorithm, the probability value was 0.619 and the Chi-square value was 247.578, with zero cells (0.0%) having expected frequencies less than 5. For the IDEA 128-bit block cipher algorithm the probability value was p = 0.989 and a Chi-square value was 205.784.

For the RC2 128-bit block cipher algorithm, the probability value was p = 0.236 and the Chi-square value was 270.899.

In summary for all the block cipher algorithms, this test shows that characters in the dataset is uniform distributed and is essentially random.

Tab. 4.3: Chi-square for different block cipher algorithms

	AES 128-bit	DES 64-bit	IDEA 128-bit	RC2 128-bit
Chi-square	285.649	247.578	205.784.161	270.899
Probability *(p-value)*	0.091	0.619	0.989	0.236

The Chi-square test was also applied to the encrypted data in the BECC dataset for RC4 128-bit and HC 128-bit stream cipher algorithms as shown in the Table 4.4. With RC4, the probability value was *p-values* = 0.989 and a Chi-square value was 205.784, but the HC probability value was *p-values*= 0.242 and a Chi-square value was 270.451. 0 cells (0.0%) have expected frequencies less than 5. The minimum expected cell Frequency for RC4 is 2046.8 and HC is 2046.8.

In addition, it was observed that the all encrypted BECC dataset consist of characters that are uniformly distributed.

Residual is the difference between the observed values and the dependent variable. Examining residuals can tell us whether our assumptions are reasonable and our selection of model is appropriate. Table 4.5 shows the maximum and minimum residual for each block cipher algorithm. According to the result, we

Tab. 4.4: Chi-square for two stream cipher algorithms

	RC4 128-bit	HC 128-bit
Chi-square	205.784	270.451
Probability *(p-value)*	0.989	0.242

have found that all block cipher-text show a random pattern and are uniformly distributed.

Figures 4.10 and 4.11 show the plot of residuals for the AES and RC2 algorithms. The reference lines at 0 in these plots emphasizes that the residuals cluster around the 150-250 between positive and negative differences for the AES algorithm, and for RC2 cluster around 200-150. The figures show that there are no systematic patterns apparent in these plots. It is very important to analyse the plot of the residuals versus every variable to make sure that a selected model is the best model possible to use. As observed, the points in the plots seem to be fluctuating randomly around zero in an unpatterned fashion. Moreover, the remaining block cipher algorithms produce similar plots, which mean the residual plots all show a fairly random pattern. The residuals also appear to behave randomly which further suggests that the model fits the data well.

Tab. 4.5: Residual for the block cipher algorithms.

	AES	DES	IDEA	RC2
Max Residual	109.2	113.1	99.2	183.2
Min Residual	-162.8	-126.9	-139.8	-100.8

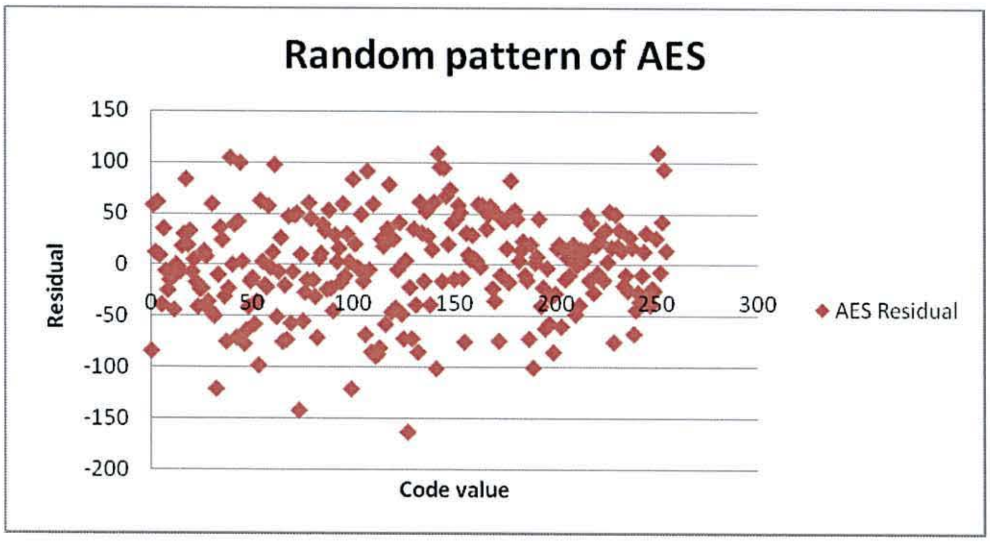


Fig. 4.10: Residual plot for the AES algorithm.

Table 4.6 shows the residual for the cipher-texts of the RC4 and HC stream ci-

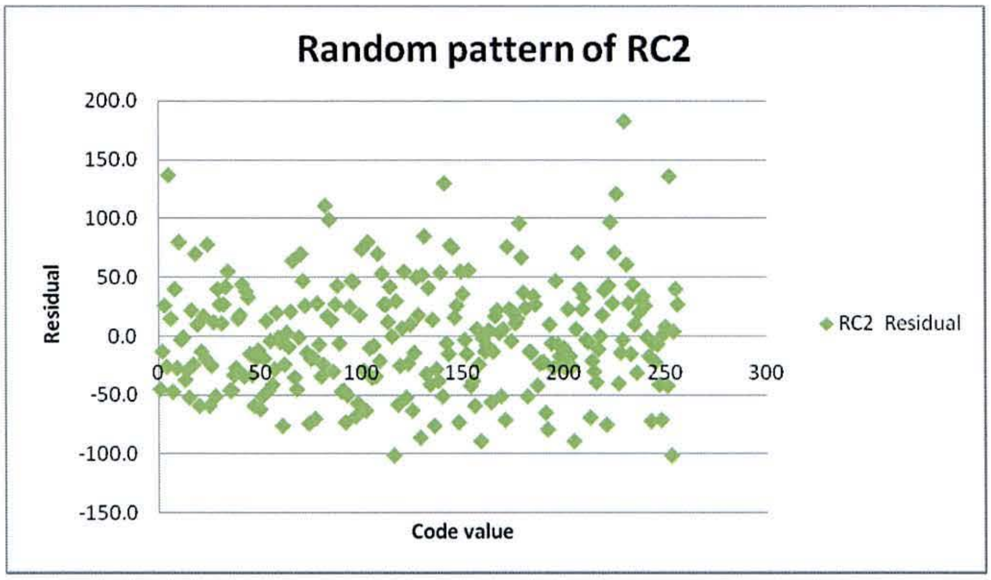


Fig. 4.11: Residual plot for the RC2 algorithm.

pher algorithms. The results also indicate random data. The points in the plot also fluctuating randomly around zero in an unpatterned fashion as shown in Figures 4.12 and 4.13. The two residual plots below shows the results for the RC4 algorithm with 128-bit. In this case the plots provide further evidence of the randomness of the dataset.

Tab. 4.6: Residual for the stream cipher.

	RC4 128-bit	HC 128-bit
Max Residual	183.2	164.2
Min Residual	-100.8	-123.8

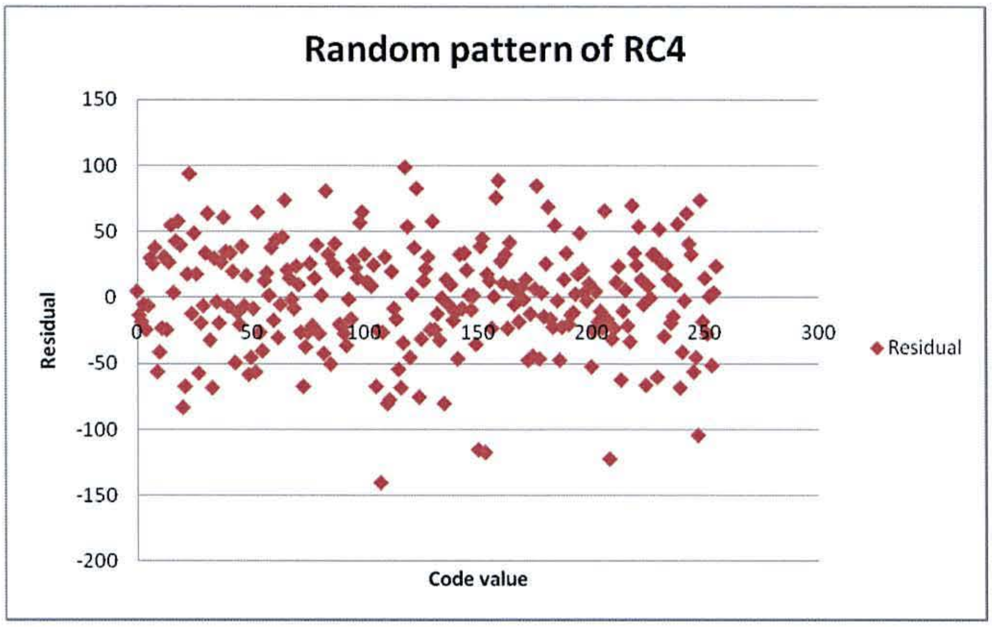


Fig. 4.12: Residual plot for the RC4 algorithm.

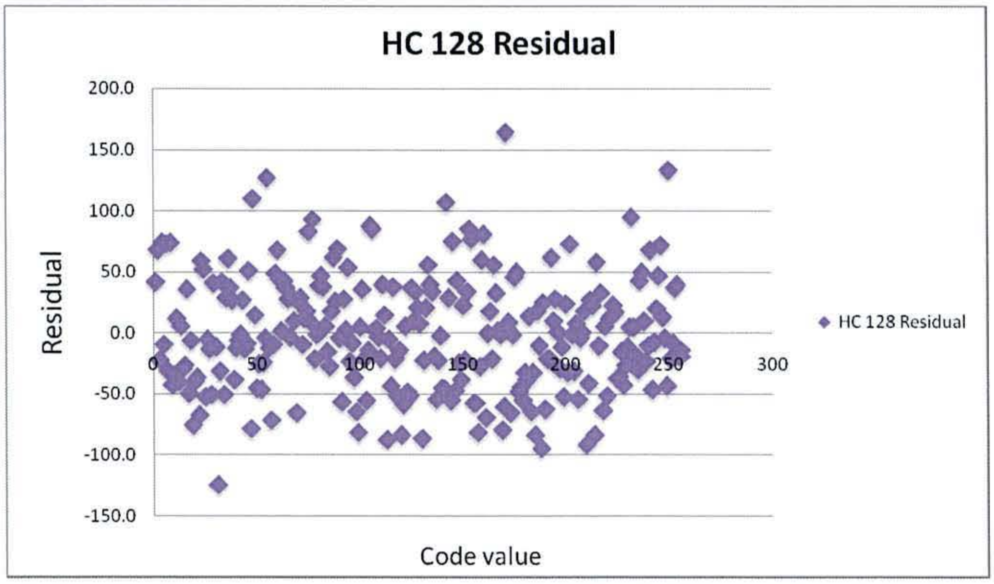


Fig. 4.13: Residual plot for the HC algorithm.

4.6.4 Compressing the Dataset Using the PPM Compression Algorithm

In 1984, John Cleary and Ian Witten developed the Prediction by Partial Matching (PPM) data compression algorithm. Throughout the past decade, the PPM data compression method has set the performance standard in lossless compression of text. The PPM method is based on a method, which maintains a statistical model of the text.

The PPM compression method has become a benchmark in the compression community. It is considered to be one of the best lossless compression algorithms. It generates 'predictions' for each character in the input. Each prediction take the form of a probability distribution that is provided to an encoder. The encoder is usually an arithmetic coder [271] [257] [258].

This section describes the application of eight block cipher and five stream cipher algorithms to derive computer models for predicting the encrypted block and stream cipher algorithms. If the cipher-text in the datasets are random, then they will be uncompressible.

Two datasets are used. The first set has 8 classes (one for each encryption algorithm with different key size) and 240 input files, which results in 240 input files to be compressed. The second dataset has 4 classes (one for each algorithm). Four different block cipher algorithms (AES, DES, IDEA and RC2) are used: the AES algorithm with three different key sizes (128, 192 and 256-bit), DES 64-bit, IDEA 128-bit and RC2 with three different keys sizes (42, 82 and 128-bit). In the experiments, the input data files to the compression tool are 512KB in size after encryption produced two types of file text (containing only printable characters).

Table 4.7 shows the file position for all block cipher algorithms and Figure 4.14 shows the result of file position against the compression rate in bit per characters (bpc). According to our results, we have observed that there is no significant differences between the algorithms with the highest compression rate being 8.363 (bpc) and the lowest 8.076 (bpc). The result shows that all datasets are incompressible (i.e.much of the data is random) because the compression ratios are > 8.0 bpc i.e. the file expands rather than contracts when running the compression tool on the file.

Tab. 4.7: Compression results for different block cipher using PPM method.

File position	Compression (bpc)							
	AES 128	AES 192	AES 256	DES 64	IDEA 128	RC2 128	RC2 42	RC2 84
10000000	8.240	8.240	8.240	8.240	8.240	8.240	8.239	8.240
20000000	8.154	8.154	8.154	8.155	8.155	8.154	8.154	8.155
30000000	8.116	8.116	8.116	8.116	8.116	8.116	8.116	8.116
40000000	8.094	8.094	8.094	8.094	8.094	8.094	8.094	8.094
50000000	8.079	8.079	8.080	8.079	8.080	8.079	8.079	8.079

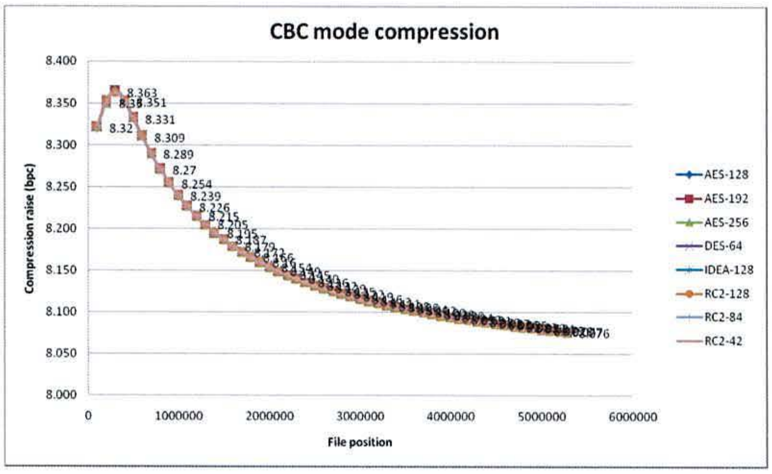


Fig. 4.14: Block cipher file position against compression (bpc).

Table 4.8 shows the file position and the compression ratio (bpc) for different stream cipher algorithms: Grain 128-bit, HC 128-bit, RC4 128-bit, VMPC 128-bit and VMPC 128-bit. All stream ciphers have used the same encryption key. The highest point is equal to 8.021(bpc) and the lowest is equal to 8.006 (bpc). All the compression ratios are > 8.0 (bpc) and therefore the data is incompressible due to the randomness of the data.

Tab. 4.8: Compression results for different stream cipher using PPM method.

File position	Compression(bpc)				
	Grain	HC128	RC4	Salsa20	VMPC
10000000	8.021	8.021	8.022	8.021	8.021
20000000	8.012	8.012	8.012	8.012	8.012
30000000	8.009	8.009	8.009	8.009	8.009
40000000	8.007	8.007	8.007	8.007	8.007
50000000	8.006	8.006	8.006	8.006	8.006

Summary

The purpose of this chapter is to create an encryption dataset to be used for the experimental evaluation in Chapter 5 and 6. Moreover, the main motivation is to create the dataset for the experimental evaluations and so that other researchers can benefit from the dataset as well. This study uses the Crypto++ library, in order to create the datasets, which is an open source library that supports a vast array of cryptographic schemes. Random samplings of text files were taken from the Internet in 2010 from reports, papers. The sample source text data ranged in size from 100 bytes to 10000 bytes.

The following block cipher algorithms were used to encrypt the sample source text: AES with three different key sizes (128, 192 and 256-bit), DES 64-bit, IDEA 128-bit and RC2 with three different key sizes (128, 84 and 42-bit). The encrypted cipher-text output was then analysed using the following tests: Frequency test, Chi-square test, and a Compression test using the Prediction by Partial Matching (PPM) algorithm.

The tests indicate that the encrypted data is essentially random in nature. The Frequency test shows a uniform distribution for the encrypted text. The Chi-square test also indicated the distribution of character codes is uniform. All encrypted data files are incompressible, again indicating the data is random in nature.

Chapter 5

Encryption Classification for Block Cipher Algorithms

The purpose of this chapter is to demonstrate that Pattern Recognition (PR) techniques can be useful tools for identification of the encryption method used from the encrypted plain-text files. This chapter considers block ciphers using Electronic Codebook (ECB) and Cipher Block Chaining (CBC) methods with different algorithms. The performance of each of the classifiers is presented. Section 5.1 introduces the classification of encryption output for block cipher algorithms. Section 5.2 addresses methodologies used in this study. Section 5.3 explains the identification of the encryption method. Section 5.4 describes the experimental results. Finally, the chapter concludes with a summary.

5.1 Introduction

A typical cipher takes a plain-text message and some secret keying data as its input and produces an encrypted version of the original message, known as the cipher-text. An attack on a cipher can make use of the cipher-text alone or it can make use of some of the plain-text and its corresponding cipher-text. Cryptanalysis is the process of recovering the plain-text and/or key from a cipher-text. Most encryption algorithms have a finite key space, hence, are vulnerable to an exhaustive key search attack. However, it is very difficult to identify the encryption keys because in most cases the size of the key is such that the time and resources required are not generally available. A random search through a finite but large key space is not usually an acceptable cryptanalysis method.

In Cryptanalysis, when only the cipher-text is obtained, there are initially two significant tasks: identification of the encryption technique applied and the encryption key identification. Cryptanalysis attempts to identify weaknesses in the algorithms used for encryption or the methods used to generate keys. In the con-text of this chapter, however, the emphasis is to explore the possibility of identifying encryption methods by applying Pattern Recognition techniques.

The following block cipher algorithms were considered: DES, IDEA, AES, RC2, RC6, Blowfish, Twofish, XTA, CAST and DESede (3DES) operating in ECB and CBC modes. Eight different classification techniques: Naïve Bayes (NB), Support Vector Machines (SVM), neural networks (MPL), Instance based learning (IBL), Bagging (Bag), AdaBoostM1 (AdaBM1), Rotation Forest (RoFo) and Decision Tree (C4.5), were used to try to identify the encryption method.

5.2 Methodologies

5.2.1 Using Matlab to Generate WEKA Files

The encryption was carried out by using the Crypto++ library, which created the WEKA file. A Matlab program was used to generate the WEKA file. The previous chapter mentioned that WEKA is a very powerful data mining tool, which allows a user to test dataset with a broad set of classifiers. For further details about the code see Appendix A Section A.1.

5.2.2 Using 10 Fold Cross-Validation

Ten fold cross-validation was used in the experiment which produced an accuracy measurement, which is the percentage of correctly classified instances over the total number of instances, as follows:

1. Data is divided into 10 equal partitions.
2. Then 9/10 of the data is used for training and 1/10 for testing.
3. The whole process is repeated 10 times. The overall error rate is equal to the average of error rates of each partition.
4. All the classifiers were trained using the same training sets and were tested on the same testing sets to establish the classification accuracy.

The following equation was used to measure the accuracy:

$$Accuracy(\%) = \frac{Sum\ of\ correct\ classifications}{Total\ number\ of\ classifications} \times 100.$$

5.2.3 Confusion Matrix

The confusion matrix is a useful tool in machine learning that enables analysis of the errors that the learning system makes. The confusion matrix, in unsupervised learning is characteristically named a matching matrix. Its focus is on the predictive capability of a model rather than how quick the models take to perform the classification. It can be used as an indication of the properties of a classification rule. Each class consists of a number of elements that are correctly or incorrectly classified. The result of the classification phase is called the confusion matrix, which is a detailed report on the performance of a single classifier. In the matrix, the columns represent the class and the rows represent the original data, as in the classification model shown in Table 5.1. The diagonals in the matrix represent typical cases that were classified correctly, for example, all cells of the diagonal represent cases of misclassified instances.

Every instance in the test set compares the actual class to the class that was assigned by the classifier. The matrix as well illustrates the accuracy of the classifier as the percentage of correctly classified patterns in a given class divided by the total number of pattern in that class.

A true positive is one that is correctly classified by the classifier and a false negative is one that is incorrectly classified by the classifier. The advantage of using the confusion matrix is the ability to consider the performance of all the classification forms.

Based on the elements in Table 5.1, it is possible to determine and find the correct and incorrect classifiers [272] [273] [274]. The advantages of using this performance evaluation confusion matrix tool is that we can simply observe if the model is confusing two classes (i.e. commonly mislabelling one as another).

Tab. 5.1: *Confusion Matrix*

	Predicted negative(class)	Predicted positive(class)
Actual Negative(class)	a	b
Actual Positive(class)	c	d

This is shown by the followings four elements [275]:

- True Positive (TP): An element is predicted as faulty and in reality is faulty.

$$TP = d/(c + d)$$

- False Positive (FP): An element is predicted as faulty and in reality is not faulty.

$$FP = b/(a + b)$$

- True Negative (TN): An element is predicted as not faulty and in reality is not faulty.

$$TN = a/(a + b)$$

- False Negative (FN): An element is predicted as not faulty and in reality is faulty.

$$FN = c/(c + d)$$

5.3 Identification of Encryption Method

This section presents the results of the experiments conducted in order to study the performance of the proposed method. The identification of encryption algorithms for input block ciphers are presented with each data input file being a data point in our dataset. An RM Desktop PC, with a 3.06 GHz processor operating under UNIX was used to perform the classification experiments.

The study included two groups of block cipher algorithms:

- The first group considered the following block cipher algorithms: DES (64-bit), IDEA (128-bit), AES (128, 192, 256-bit) and RC2 (42, 84, 128-bit).
- The second group included another seven block cipher algorithms: RC2, RC6, Blowfish, Twofish, XTA, CAST and DESede (3DES), all with the same key size (128-bit).

The first group of block cipher was operated in Electronic Codebook (ECB) mode and Cipher-Block Chaining (CBC) mode, while the second group of block cipher was operated only in CBC mode.

The classifiers used in WEKA included: Naive Bayes (NB), SVM, MLP, IBL, Bagging, AdaBM1, RoFo and C4.5 classifiers. In the experiments, 512KB input data files after encryption produced two types of text files. We focused on binary files for classification, due to output produced by the encryption algorithms being binary. Confusion matrices were used to find the highest accuracy. We used the equation in Section 5.2.2 to calculate the accuracy and the Multidimensional Scaling (MDS) method was used to find the similarity and dissimilarity between Symmetric algorithms.

In the first experiment, the dataset the data files in the dataset were divided into 8-bit codes, and the Matlab program in Appendix A.2 was used to extract histograms of the encrypted data. In the second experiment, the data files in the dataset were divided into 16-bit codes, and again the Matlab program was used to extract histograms of the encrypted data. For the 8-bit datasets there are 256 possible attributes or features and for the 16-bit codes there are 65536 possible attributes or features.

The following describes the experimental setup:

1. In the first experiment (block ciphers), for each algorithm, the 30 source text files from the BECC dataset (512KB in size) were encrypted using AES, IDEA and RC2 (128-bit) and DES 64-bit algorithms. Different key number were used with each– file. 1, 3, 5 and 30 keys-resulting in 120 files being processed. These were chosen to make the data more difficult to identity. These 120 encrypted files were used as data points in WEKA. When using three keys, the 30 source text files were divided by three giving three groups of ten files. Each of the three keys were then used to encrypt each of the ten files. This was repeated for all the algorithms. When using 5 keys, the 30 input files were divided by 5 giving 5 groups of six files. Each of the 5 keys were then used to encrypt each of the six files. Again, this was repeated for all the algorithms.

2. In the second experiment, two parts were carried out. In the first part, the 30 source files from the (BSFC) dataset were used with different encryption keys: for AES (128, 192 and 256-bit), for DES 64-bit, IDEA 128-bit and RC2 (42, 84 and 128-bit) algorithms. The same method in terms of numbers of keys was used as described above resulting in 240 encrypted files being processed. In the second part; the same algorithms with the same number of keys were used as used in first part, however, this time while the numbers of the keys were the same, the types of keys were different. These two parts were carried out to check whether there was an effect on accuracy when using one fixed encryption key with different version of the keys.

3. In the third experiment, the algorithms were used but different data points were chosen from the BSFC, which contained 120, 240 and 400 data points. Again, the 30 plain-text source files was encrypted using 1, 3, 5 and 30 keys for each algorithm. In this simulation, the first and second results from the 120 and 240 data points had already been obtained and 100 data points were added and used with all block cipher algorithms. However, this time 100 data points were divided by 5 producing 20 input files giving 5 groups of 20 text files. Each of the 5 keys were then used to encrypt each of the 20 the files. Again, this was repeated for all the algorithms. The aim was to verify whether there was an effect on accuracy when using different data points 128-bit encryption keys.

4. In the fifth experiment, different types of block cipher algorithms were used with 30 encryption text files from the BECC dataset for each algorithm resulting in 210 data points: RC2, RC6, Blowfish, Twofish, XTA, CAST and DESede (Triple DES) all with 128-bit. The same settings were used as were used in the first experiment with ECB and CBC modes.

5.4 Experimental Results

5.4.1 Block Cipher Algorithms with ECB Mode

This section shows the results of the experiments conducted for the evaluation of the classification of encrypted text produced by block cipher algorithms. The training set used for the experiments contained examples of Symmetric algorithms from the two different classes of block cipher algorithms, and was used to build the classification model. The testing set represents the unknown Symmetric cipher algorithms that were to be classified. The block cipher algorithms in both the training and testing sets are labelled with the appropriate class a priori.

Because the class of each block cipher within the datasets is known, it is possible to evaluate the performance of the classifier by comparing the predicted class against the known class. A 10 fold cross-validation method was used as a test mode where 10 pairs of training sets and testing sets were created.

Each time, one of the 10 subsets is used for training and the other nine for testing. The whole process was repeated 10 times. To estimate a value on the training datasets, an accuracy measure was defined as shown in the equation 5.2.2. The overall error rate is equal to the average of error rates of each part.

All the classifiers were trained using the same training sets and were tested using the same testing sets to establish the classification accuracy. Performance statistics were calculated across all 10 trials. This provided a good indication of how well the classifier would perform on unseen data.

In this study, two datasets were used with block cipher algorithms. The first set had 8 classes (one for each encryption algorithm with a different key size) and 240 input files. The second dataset had 4 classes (one for each algorithm).

Effects of Different Numbers of Keys with 120 Data Points from the BECC Dataset

In the first experiment using 8-bit codes, the effect of different numbers of key sizes used to encrypt the plain-text was investigated for the following encryption algorithms: AES, RC2 and IDEA with fixed 128-bit encryption keys and DES with 64-bit keys. The purpose was to find the effect of different numbers of key sizes with different block ciphers algorithms, in order to determine which one obtained the higher classification accuracy between them (as shown in Table 5.2). The number of input files used was 30 and the number of keys used simultaneously (ensemble classification) was 1, 3, 5 and 301 keys. Here, Figure 5.1 shows that RoFo classifiers have a better overall accuracy performance, and AdaBM1 achieved the lowest accuracy. Furthermore, and as expected, all the classifiers (apart from AdaBM1) produced good accuracy when using 1, 3 and 5 numbers of keys. It can be seen, also as expected, that the accuracy drops when using one key for each file (30 keys). Naive Bayes and MLP classifier with one key obtained the same accuracy (99.17%).

Tab. 5.2: *Accuracy results for the 4 classes with 1, 3, 5 and 30 numbers of key sizes with ECB mode.*

Block cipher Datasets / Algorithms	1 Key: 256 Attributes, 120 Instances (%)	3 Keys: 256 Attributes, 120 Instances (%)	5 Keys: 256 Attributes, 120 Instances (%)	30 keys: 256 Attributes, 120 Instances (%)
Naive Bayes	99.17	95.83	89.17	51.66
SVM	97.50	96.66	96.66	35.00
MPL	99.17	98.33	98.33	35.00
IBL	96.66	96.66	95.83	25.00
Bag	96.66	89.17	85.83	43.33
AdaBM1	95.83	47.50	45.83	37.50
RoFo	98.33	98.33	46.66	38.33
C4.5	93.33	80.00	69.16	44.17

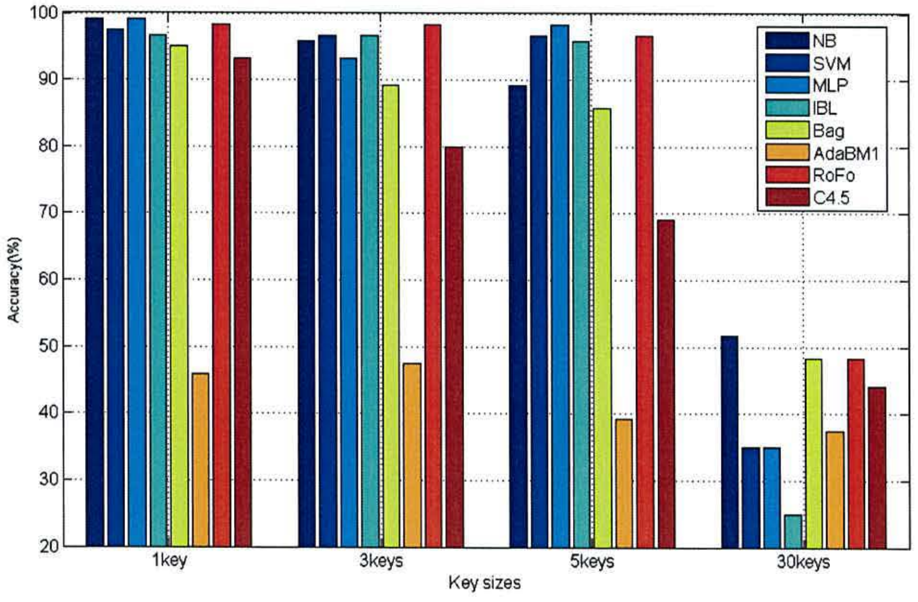


Fig. 5.1: Encryption accuracy of different algorithms (AES, RC2, IDEA(128-bit)) and DES(64-bit) key sizes.

Effects of Different Numbers of Keys With 240 Data Points from the BECC Dataset

Measurements of the performance against different numbers of keys were conducted using two datasets of different sizes, performed with 256 features from encryption algorithms. Two datasets first using 8 classes dataset and second using 4 classes, and different numbers of the keys (1, 3, 5 and 30), were used to encrypt the files. In this way, one can view which elements were correctly classified and which were misclassified and which number of keys obtain a higher accuracy.

240 data points were produced after applying the AES (128, 192 and 256-bit), DES 64-bit, IDEA 128-bit and RC2 (42, 84 and 128-bit) algorithms where each algorithm had 30 source text files. Table 5.3 for the 8 classes and Table 5.4 for the 4 classes show the experimental results of these classifications. The experiment was conducted using different numbers of keys for each algorithm (1, 3, 5 and 30). Equation 5.2.2 was used to calculate the accuracy for each different set of data. It can be seen in Figure 5.2 that using one encryption key produces a higher classification accuracy of 100% meaning that the 240 instances (files) were correctly classified.

In contrast, using 30 different numbers keys (one for each file) resulted in a lower classification accuracy. This was expected as the keys are generated randomly and will affect the pattern of test data. The results show that the accuracy of the classification was reduced with an increase in the number of encryption keys. Furthermore, the result shows that the 4 classes dataset obtained higher accuracy than dataset the 8 classes.

Tab. 5.3: *Accuracy results for the 8 classes with different numbers of keys with ECB mode.*

Block cipher Datasets / Algorithms	1Key: 8 Classes 256 Attributes, 240 Instances (%)	3Keys: 8 Classes 256 Attributes, 240 Instances (%)	5Keys: 8 Classes 256 Attributes, 240 Instances (%)	30keys: 8 Classes 256 Attributes, 240 Instances (%)
Naive Bayes	32.08	30.42	25.00	20.41
SVM	32.08	29.58	28.33	19.16
MPL	30.00	27.50	28.83	16.66
IBL	30.41	27.91	26.25	12.08
Bag	31.25	29.58	27.91	20.00
AdaBM1	25.00	22.08	16.66	16.25
RoFo	30.00	32.50	31.66	22.00
C4.5	44.58	41.25	32.91	22.50

Tab. 5.4: *Accuracy results for the 4 classes with different numbers of keys in ECB mode.*

Block cipher Datasets / Algorithms	1Key 4 Classes 256 Attributes, 240 Instances (%)	3Keys 4 Classes 256 Attributes, 240 Instances (%)	5Keys 4 Classes 256 Attributes, 240 Instances (%)	30Keys 4 Classes 256 Attributes, 240 Instances (%)
Naive Bayes	100	99.16	97.08	40.00
SVM	99.16	98.75	99.16	44.16
MPL	99.58	98.75	89.16	42.20
IBL	99.17	98.33	98.74	40.00
Bag	98.75	95.41	94.16	38.33
AdaBM1	75.00	70.41	59.58	39.16
RoFo	99.58	99.16	98.75	35.83
C4.5	97.91	93.33	91.25	40.00

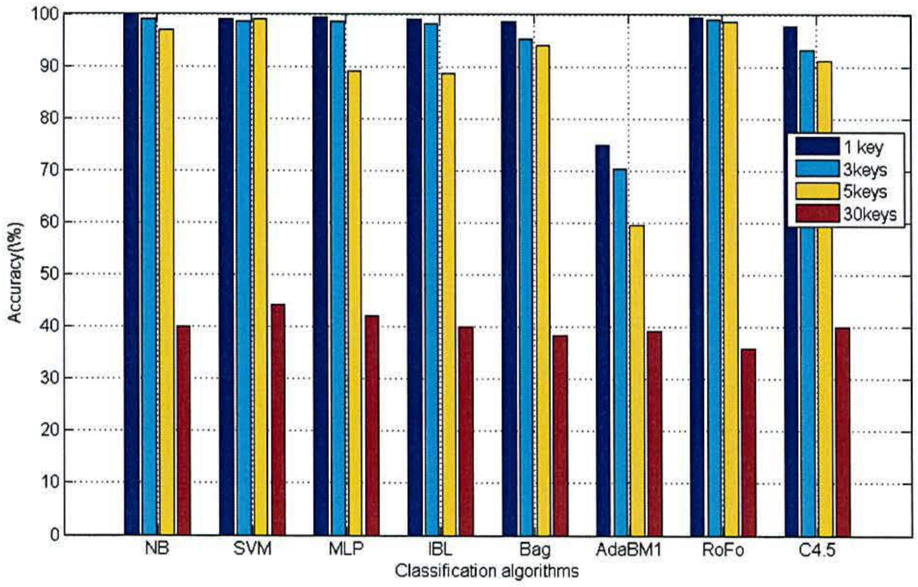


Fig. 5.2: The accuracy for each algorithm for the 4 classes dataset.

For the following experiments, the same encryption algorithms with the different numbers of keys (1, 3 and 30) individually were used to encrypt the file, which mean each algorithm had its own numbers of keys. The following points were discovered, as shown in Figure 5.3, Table 5.5 for the 8 classes case and Table 5.6 for the 4 classes case:

- All classifiers with one key achieved a high accuracy classification of 100% meaning that 240 instances out of 240 were correctly classified with both the 8 classes and 4 classes datasets.

- SVM achieved a higher accuracy with 1, 3 and 30 different keys, and AdaBM1 achieved a lower accuracy with one and three keys, but the IBL classifier with 30 different keys achieved the lowest accuracy.
- On the other hand, using 30 individual keys produced the worst accuracy over all the algorithms for both cases.
- IBL, however, achieved the lowest accuracy of 12.5% meaning that only 30 instances out of 240 were correctly classified with the 8 classes dataset.

Tab. 5.5: *Accuracy results for the 8 classes with individual number of keys (1, 3 and 30)* with ECB mode.

Block cipher Datasets / Algorithms	1key: 8 Classes 256 Attributes, 240 Instances (%)	3keys: 8 Classes 256 Attributes, 240 Instances (%)	30keys: 8 Classes 256 Attributes, 240 Instances (%)
Naive Bayes	100	40.83	28.33
SVM	100	18.33	17.08
MPL	100	14.16	21.67
IBL	100	13.33	12.50
Bag	100	32.50	25.83
AdaBM1	25.00	14.16	20.42
RoFo	100	28.33	30.83
C4.5	100	22.08	27.92

Tab. 5.6: *Accuracy results for the 4 classes with individual number of keys (1, 3 and 30)* with ECB mode.

Block cipher Datasets / Algorithms	1key: 4 Classes 256 Attributes, 240 Instances (%)	3keys: 4 Classes 256 Attributes, 240 Instances (%)	30key: 4 Classes 256 Attributes, 240 Instances (%)
Naive Bayes	100	66.66	44.17
SVM	100	48.75	32.08
MPL	100	42.50	39.58
IBL	100	37.50	30.42
Bag	100	65.00	47.50
AdaBM1	37.50	40.83	43.33
RoFo	100	65.00	53.33
C4.5	100	57.91	51.67

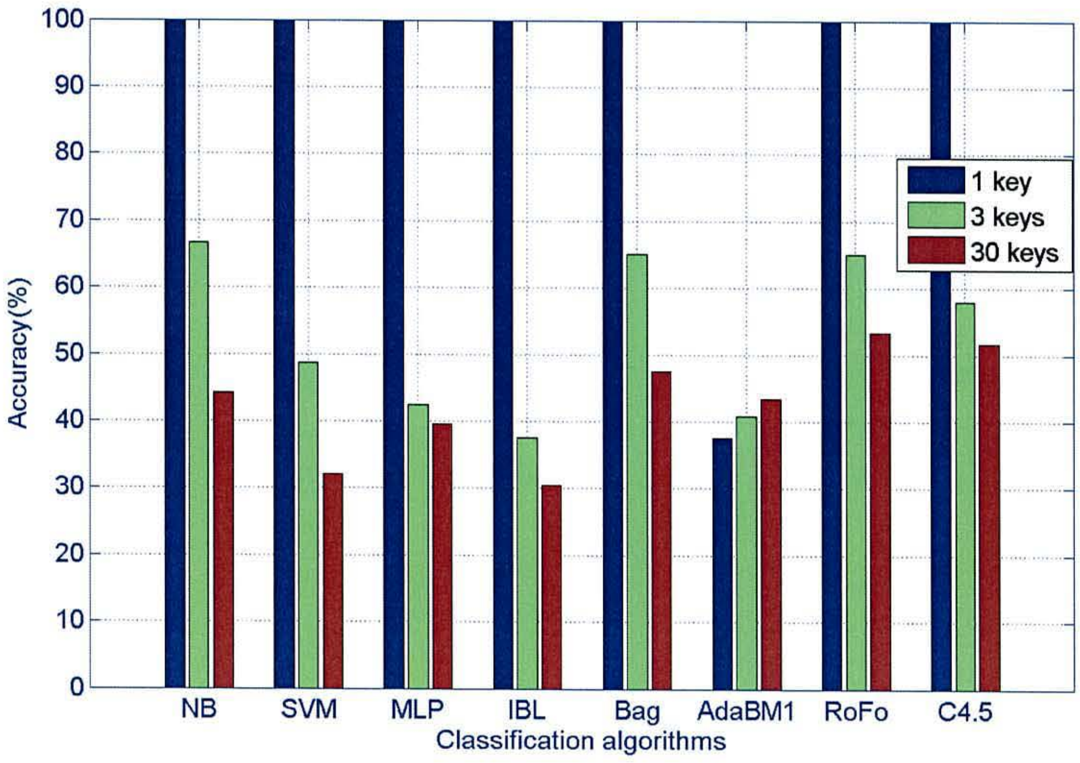


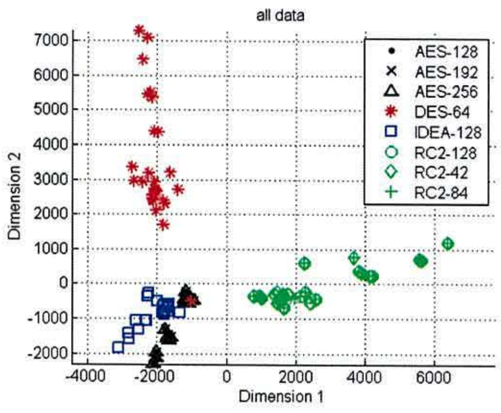
Fig. 5.3: Encryption accuracy with individual number of keys (1, 3 and 30) with ECB mode.

Multidimensional Scaling (MDS) for Different Numbers of Keys

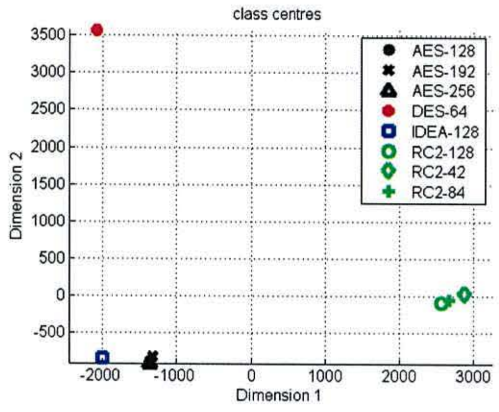
Figure 5.4 (a) shows the data scatter-plots and the centres of the 240 data points with different key sizes 1, 3 and 5. The plot shows the classes are found sporadically, suggesting that the high recall and precision rates for AES are higher than for other algorithms, as shown in Table 5.3. The centres of the "clouds" of points for the 8 classes dataset are plotted in the same way as in the first experiment. In Figure 5.4 (b), according to this plot, all algorithms have similar representation. Note that the scales of all plots are different. The class centres are indistinguishable if plotted on the axes of a sub-plot. Finally, we can say this highlights the difficulty in recognising the type of code through simple pattern classification methods. Figure 5.5 shows an image of the distance matrix computed by using classification identification accuracy of each encryption algorithm. The block of 30-by-30 distances is outlined in black. Blue means high similarity while yellow and red indicate low similarity. The class labels are as follows: 1 AES (128), 2 AES (192), 3 AES (256), 4 DES (64), 5 IDEA (128), 6 RC2 (128), 7 RC2 (42) and 8 RC2 (84).

The encoding technique that stands out from the rest is AES. The 3-by-3 block sub-matrix in the top left corner is largely blue and shows the similarity within the code. Interestingly, the AES algorithm is distinct from the rest of the algorithms; also note that the three versions of AES (128, 192, and 256) are not distinct within AES. The red vertical and horizontal lines demonstrate the unusually large distances compared to the rest. Unlike ECB, there is no clear pattern to suggest that any of the codes are distinct.

With 3 and 5 keys, the same result was obtained, as shown in Figures 5.6 (a) and Figure 5.7 (b). Unlike ECB, there is no clear pattern to suggest that any of the codes are distinguishable.



(a) All data



(b) Class centres

Fig. 5.4: Scatter-plots of the 240 data points and the centres for the ECB mode with the 8-bit encoding using one key.

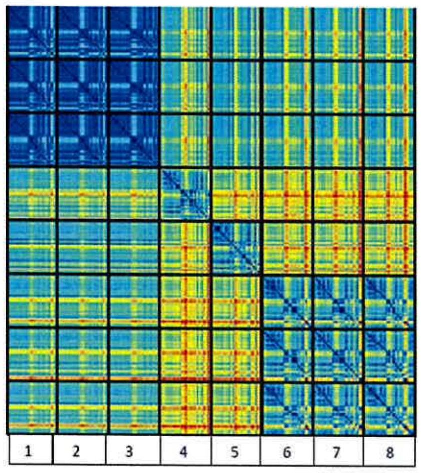
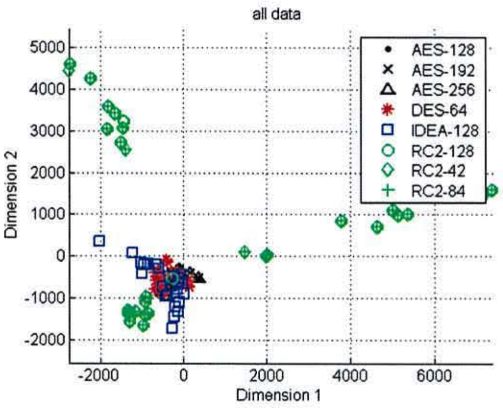
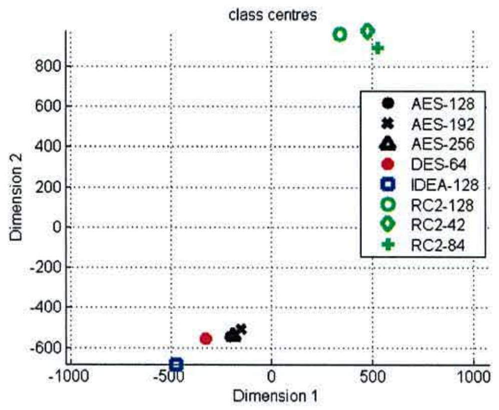


Fig. 5.5: The Distance Matrix for ECB mode with the 8-bit encoding. The class labels are as follows: 1 AES(128), 2 AES(192), 3 AES(256), 4 DES(64), 5 IDEA(128), 6 RC2(128), 7 RC2(42) and 8 RC2(84)



(a) All data



(b) Class centres

Fig. 5.6: Scatter-plots of the 240 data points and the centres for the ECB mode with 8-bit encoding using three keys.

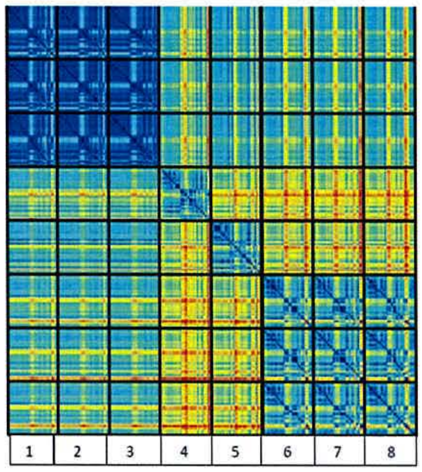
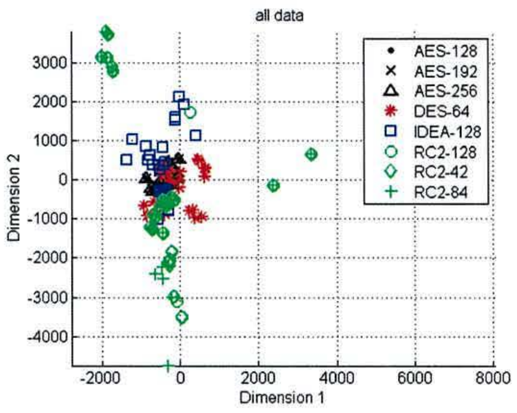
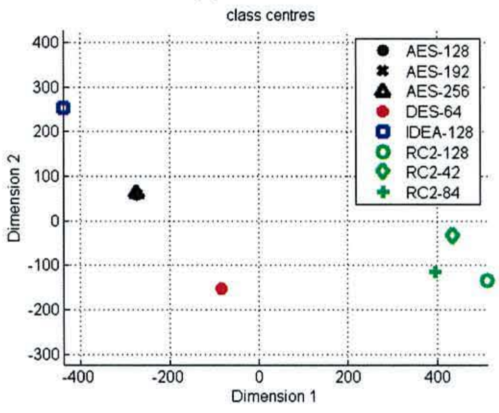


Fig. 5.7: The Distance Matrix for ECB mode with 8-bit encoding. The class labels are as follows: 1 AES(128), 2 AES(192), 3 AES(256), 4 DES(64), 5 IDEA(128), 6 RC2(128), 7 RC2(42) and 8 RC2(84)



(a) All data



(b) Class centres

Fig. 5.8: Scatter-plot of the 240 data points and the centres for the ECB mode with 8-bit encoding encoding using five keys.

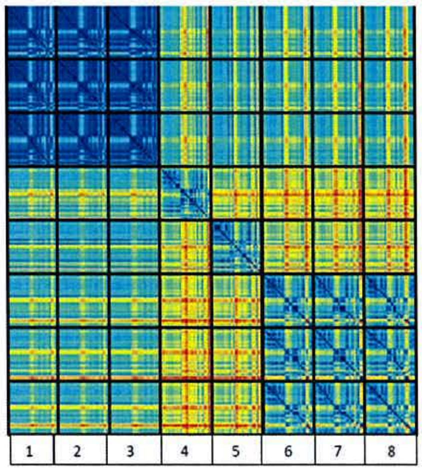


Fig. 5.9: The image of the Distance Matrix for ECB mode with 8-bit coding. The class labels are as follows: 1 AES (128), 2 AES (192), 3 AES (256), 4 DES (64), 5 IDEA (128), 6 RC2 (128), 7 RC2 (42) and 8 RC2 (84)

Effects of Different Instances on Performance

Measurements of the performance against the numbers of keys were conducted using two datasets (8 classes and 4 classes). Different encryption text files, which includes (120, 240 and 400 instances) were used and performed with 256 features from encryption algorithms. In this way, one can view the effects of different instances on performance with the same 256 features. This experiment deals with the effect of increasing the number of encryption text files (instances) on overall accuracy as shown in Table 5.7. Here different encryption text files were used for each algorithm: AES 128-bit, DES 64-bit, IDEA 128-bit and RC2 128-bit. Thus the total numbers of the instances are 120, 240 and 400 respectively. With 120 instances, each algorithm has 30 data points, with 240 each algorithm has 30 data points (AES with three version keys (128-bit, 192-bit, 256-bit), DES 64-bit, IDEA 128-bit and RC2 with three version key (128-bit, 42-bit, 84-bit) and 400 instances for each algorithm with 100 different files divided by 5 to find the accuracy. Figure 5.10 shows that all classifiers using 400 instances achieved the highest accuracy, while using 120 instances achieved the lowest accuracy. It is also evident that RoFo classifiers produce the most accurate results for all instances and IBL performs very badly when operating with 120 and 240 instances.

Tab. 5.7: *Accuracy results for different instances with the same features (120, 240, 400) with ECB mode.*

Algorithms	120 Instances (%)	240 Instances (%)	400 Instances (%)
Naive Bayes	43.00	44.17	82.25
SVM	31.00	32.08	91.25
MLP	30.00	39.58	92.25
IBL	20.00	30.42	93.50
BAg	25.00	47.50	86.75
AdaBM1	30.00	43.33	39.00
RoFo	45.00	53.33	93.00
C4.5	31.00	51.67	83.00

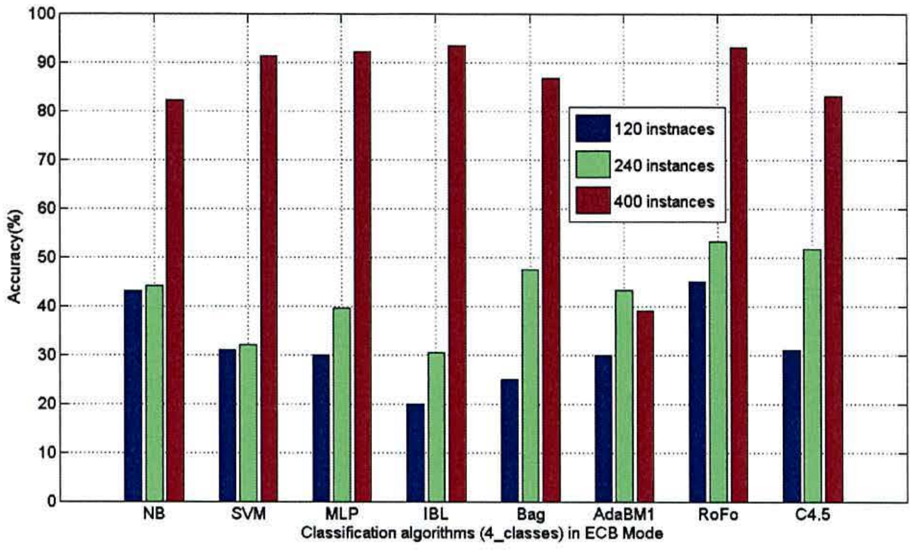


Fig. 5.10: The accuracy for all algorithms with different instances (120, 240 and 400 instances (BECC)) with ECB mode.

Evaluation of Using 30 Numbers of Keys with 30 BECC Individually

Experiments were also conducted using 8-bit with ECB mode. For individual classes, the confusion matrices simply inform how the classifier behaves. Each table was calculated using 4 classes with different numbers of keys and 240 data points were used: AES with three different keys(128, 192, 256-bit), DES 64-bit, IDEA 128-bit and RC2 (42, 84, 128-bit) with three different keys. In this case the number of samples in one class is significantly more than that in the other class resulting in what is called "imbalance", which happens often with different block cipher datasets. The accuracy evaluation of a classifier is not representative of the true performance of each classifier. In some of the tables, the reason for the large imbalances are that the number of the keys is too high, the effect of the type of classifier used and the type of the algorithm used.

In the experiments, ECB mode used the same block cipher algorithms with the same classifiers. The difference was that 30 different data points were used for each algorithm individually as well as 30 different numbers of keys individually, which were divided into 8-bit blocks. Table 5.8 and Figure 5.11 show the classification accuracy results of these eight classifiers. The experimental results, for the 8 classes dataset (in the second column) reveal the classification accuracy. It can be observed that the RoFo achieved the highest accuracy classification of 30.83% meaning that 74 data points out of 240 were correctly classified. On the other hand, IBL had the lowest accuracy classification of 12.5% meaning that only 30 data point out of 240 were correctly classified. The reason for improved accuracy with RoFo and C4.5 classifiers is due to grouping the AES with the three keys in one class and the RC2 in one class and using random data.

Tab. 5.8: *Classification accuracy performance of the classifier four block cipher with ECB mode.*

Block cipher Datasets / Algorithms	8 Classes 256 attributes, 240 Instances (%)	4 Classes 256 attributes, 240 Instances (%)
Naive Bayes	28.33	44.17
SVM	17.08	32.08
MPL	21.67	39.58
IBL	12.50	30.42
Bag	25.83	47.50
AdaBM1	20.42	43.33
RoFo	30.83	53.33
C4.5	27.92	51.67

The experimental results for the 4 classes dataset (in the third column) show that RoFo also outperforms all other classifiers with the classification accuracy of 53.33% meaning that only 128 data points out of 240 were correctly classified. IBL once again has the lowest accuracy classification of 30.42% meaning that only 73 data points out of 240 were correctly classified. The reason for improved accuracy is due to grouping the encryption keys to form one class of AES and the same for RC2 and using longer 16-bit data points. The results show that RoFo and C4.5 perform much better than the SVM classifier because AES with three different numbers of keys (128, 192 and 256) were combined and RC2 with (128, 84 and 42) key sizes.

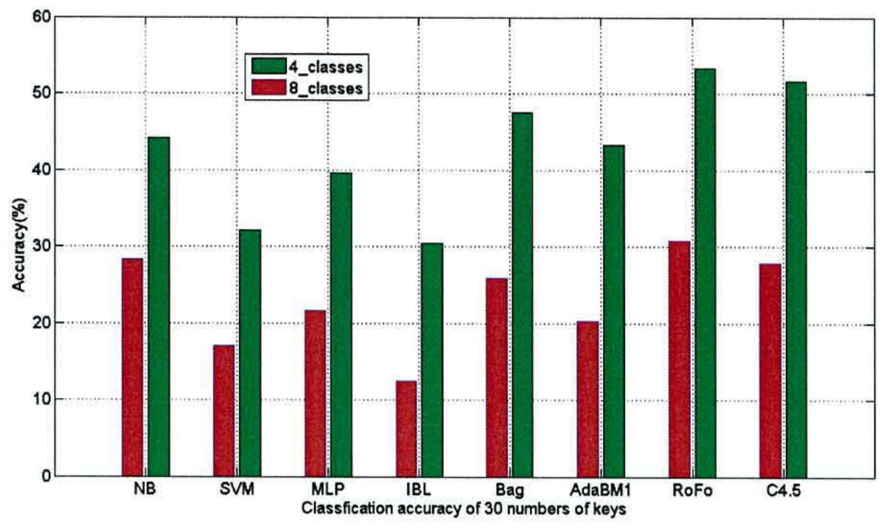


Fig. 5.11: The accuracy for each of the classifiers with ECB mode.

Further experiments using 16-bit codes with ECB mode were conducted. In this part, the same eight classifiers were used with the same block cipher algorithms as used with 8-bit codes. However, the key lengths were changed. The main idea was to find the difference between all algorithms, and whether or not the key length effected accuracy. According to the results, it was found that when the key length was longer in ECB mode, there was a reduction in the accuracy.

These experiments were performed on the 30 BECC encryption data files of fixed size (512 KB), that were divided into 16-bit codes . It can be observed from Table 5.9 in the 8 classes dataset that Naive Bayes achieved the highest accuracy classification of 29.17% meaning that 70 instances out of 240 were correctly classified. On other hand, IBL had the lowest accuracy classification of 12.5% meaning that only 30 data points out of 240 were correctly classified.

The experimental result of the 4 classes dataset again shows that the highest accuracy was Naive Bayes but at 57.92%. This time, the Naive Bayes classifier correctly classified 139 instances out of 240. However, SVM had the lowest accuracy classification of 36.25% with only 87 input data out of 240 were correctly classified. Also Figure 5.12 shows that the Naive Bayes classifier obtained a higher accuracy in the 4 classes dataset and the lowest was the IBL classifier. However, in terms of using both RoFo and MLP classifiers with a 16-bit codes the result was not obtainable since the features were too many, despite using Java Virtual Machine (JVM) 64-bit and 6GB of RAM. According to Table 5.9, the IBL classifier again gave the worst results.

Tab. 5.9: *Classification accuracy performance of the classifiers using four-block cipher algorithm with ECB mode.*

Block cipher Datasets / Algorithms	8 Classes 65536 Attributes, 240 Instances (%)	4 Classes 65536 Attributes, 240 Instances (%)
Naive Bayes	29.17	57.92
SVM	17.08	32.08
MPL	No result	No result
IBL	12.50	37.50
Bag	17.92	41.25
AdaBM1	15.00	39.58
RoFo	No result	No result
C4.5	20.00	38.75

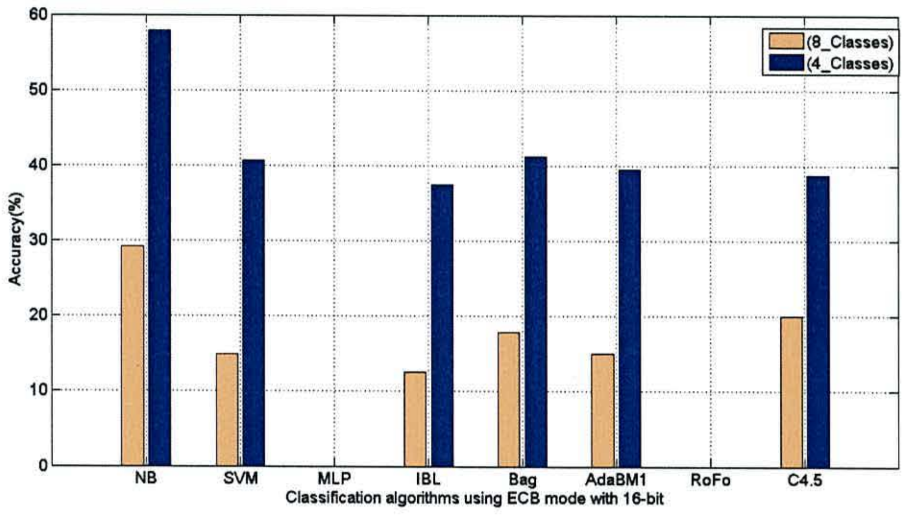


Fig. 5.12: The accuracy for each of the classifiers with ECB mode.

5.4.2 Block Cipher Algorithms With CBC Mode

This section shows the results with CBC mode with 8-bit and 16-bit using the same technique as for the ECB mode previously. The main purpose of these experiments was to find the highest accuracy between the same datasets using the same block cipher algorithms and provide a comparison with ECB mode and how numbers of keys effected the accuracy. The accuracy results are also shown for each algorithm with 240 data points and with four block cipher algorithms: AES (128, 192 and 256-bit), DES 64-bit, IDEA 128-bit and RC2 (42,84 and 128-bit) using different numbers of keys for each algorithm.

We combined two types of the algorithms to obtain a higher accuracy: AES-128, AES-192 and AES-256 as one combination, RC2-128, RC2-42 and RC2-84 as the other combination. DES-64 and IDEA-128 were used individually. Therefore the 8 classes dataset became a 4 classes dataset, where each row represents the known class of the data. Thirty BECC data points were used for each individual algorithm ($8 \times 3 = 240$). As in the calculations in the section above, the same equation in Section 5.2.2, was used and applied to all classifiers (Naive Bayes, SVM, MLP, IBL, Bag, AdaBM1, RoFo and C4.5) in CBC mode.

Effects of Different Numbers of Keys with 120 Data Points using 8-bit Codes

120 BECC encrypted text files were used with IDEA, AES, RC2 with 128-bit and DES 64-bit where each algorithm had 30 data points. The experiment was conducted using different numbers of keys for each algorithm; Table 5.10 shows the experimental results of these classifications for the 4 classes dataset. The experiment was conducted using different numbers of keys for each algorithm (1, 3, 5 and 30). Figure 5.13 shows that no one number of the keys produced the highest accuracy since all had similar accuracy even with one key. This is due to the chaining nature in CBC mode, which means that each cipher depends on the ones before.

Tab. 5.10: *Accuracy results for the 4 classes dataset with 1, 3, 5 and 30 numbers of key sizes with CBC mode.*

Block cipher Datasets / Algorithms	1Key: 256 Attributes, 120 Instances (%)	3Keys: 256 Attributes, 120 Instances (%)	5Keys: 256 Attributes, 120 Instances (%)	30keys: 256 Attributes, 120 Instances (%)
Naive Bayes	18.33	16.66	27.50	23.33
SVM	17.50	20.83	29.16	26.16
MPL	26.66	25.83	28.33	20.00
IBL	26.66	22.50	21.66	28.33
Bag	25.00	27.50	20.83	29.16
AdaBM1	20.83	23.33	18.33	24.16
RoFo	25.00	24.16	28.33	17.50
C4.5	23.33	31.66	29.16	21.66

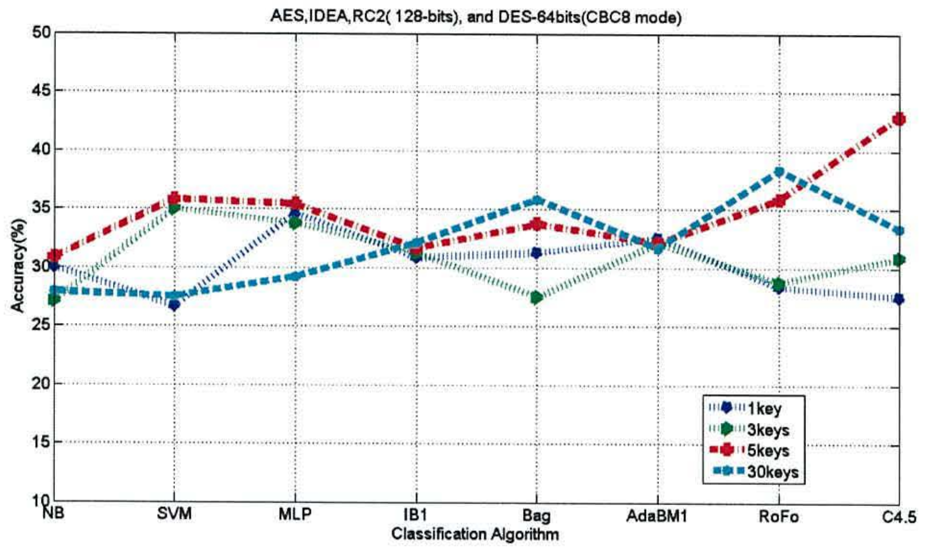


Fig. 5.13: Encryption accuracy with the 4 classes dataset with CBC mode with different numbers of keys.

Effects of Different Numbers of Keys with 240 Data Points using 8 and 16-bit Codes

This section shows the results in CBC mode with 8-bit and 16-bit codes, which used the same technique as used in ECB mode described previously. The main purpose of these experiments was to found the highest accuracy between the same datasets and the same block cipher algorithms, to provide a comparison with ECB mode and find out how numbers of key did effect the accuracy. The accuracy results were determined for each algorithm with 240 data points (input files) (8×30) and with four block cipher algorithms: AES (128, 192 and 256-bit), DES 64-bit, IDEA 128-bit and RC2 (42,84 and 128-bit) used different numbers of keys for each algorithm. We combined two types of the algorithms to obtain a higher accuracy: AES-128, AES-192 and AES-256 as one combination, RC2-128, RC2-42 and RC2-84 as the other combination. DES-64 and IDEA-128 were used individually. Therefore the 8 classes dataset became a 4 classes dataset, where each row represents the known class of the data. Thirty data points were used for each individual algorithm . As in the calculations in the section above, the same equation, in Section 5.2.2, was used and applied to all classifiers (Naive Bayes, SVM, MLP, IBL, Bag, AdaBM1, RoFo and C4.5) in CBC mode here.

Using 8-bit codes

Experiments were carried out in the same manner previously. Table 5.11 shows the classification accuracy result of the 8 classes dataset. According to the result, SVM achieved the highest accuracy classification of 15% meaning that only 36 instances out of 240 were correctly classified. C4.5 achieved the lowest accuracy of 10.42% meaning that only 25 instances out of 240 were correctly classified. However, with 4 classes dataset, the highest accuracy obtained was AdaBM1 36.25% meaning that only 87 instances out of 240 were correctly classified. In both datasets, Bag and C4.5 achieved the lowest classification accuracy of 28.33% meaning that only 68 instances out of 240 where correctly classified, while in 4 classes dataset, both Bag and C4.5 achieved the same accuracy. This is due of using CBC mode and random data points. Furthermore, Figure 5.14 shows that there were slight differences between all classifiers with the 8 classes dataset. Also, 4 classes dataset had higher classification than dataset 8 classes.

Tab. 5.11: *Classification accuracy performance of the classifier with CBC mode.*

Block cipher Datasets / Algorithms	8 Classes 256 Attributes, 240 Instances (%)	4 Classes 256 Attributes, 240 Instances (%)
Naive Bayes	12.99	30.83
SVM	15.00	35.83
MLP	10.83	33.33
IBL	11.25	28.75
Bag	10.83	28.33
AdaBM1	14.58	36.25
RoFo	10.83	31.67
C4.5	10.42	28.33

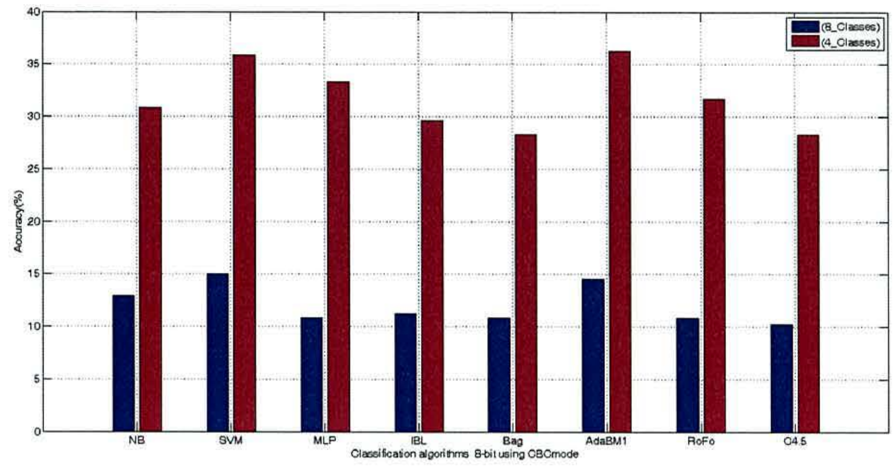


Fig. 5.14: The accuracy for each classifiers with CBC Mode.

Table 5.12 and 5.13 show the experimental results of these classification (8 classes and 4 classes datasets). The simulation was conducted using various numbers of keys for each algorithm (1, 3, 5 and 30). It can be observed from Figure 5.15 that using one key did not lead to a higher accuracy and there were no big differences using various numbers of keys. In contrast, using 30 numbers of keys (one for each file) resulted in no difference than using only key. The result did not show that the accuracy of the classification was reduced with an increase in the number of encryption keys. In addition, with 30 different keys, SVM achieved a lower accuracy and RoFo achieved a higher accuracy. The reason for them not performing well is due to the chaining key nature in CBC mode that occurs.

Tab. 5.12: *The 8 classes dataset with different numbers of key sizes with CBC mode.*

Block cipher Datasets / Algorithms	1Key: 256 Attributes, 240 Instances (%)	3Keys: 256 Attributes, 240 Instances (%)	5Keys: 256 Attributes, 240 Instances (%)	30Keys: 256 Attributes, 240 Instances (%)
Naive Bayes	12.50	11.66	11.25	14.58
SVM	10.42	12.08	12.50	11.66
MPL	12.50	11.25	10.83	12.91
IBL	14.58	10.41	10.00	18.33
Bag	14.58	16.25	10.00	14.16
AdaBM1	12.08	10.41	08.70	11.25
RoFo	11.25	14.58	12.08	17.50
C4.5	09.58	14.16	15.00	18.75

Tab. 5.13: *The 4 classes dataset with different numbers of key sizes with CBC mode.*

Block cipher Datasets / Algorithms	1Key: 256 Attributes, 240 Instances (%)	3Keys: 256 Attributes, 240 Instances (%)	5Keys: 256 Attributes, 240 Instances (%)	30Keys: 256 Attributes, 240 Instances (%)
Naive Bayes	30.00	29.58	27.91	27.91
SVM	26.66	28.33	30.00	27.50
MPL	34.58	27.50	28.33	29.17
IBL	30.83	27.08	30.83	32.85
Bag	31.25	31.66	30.41	35.83
AdaBM1	31.25	33.33	20.41	31.66
RoFo	28.33	28.75	27.50	38.33
C4.5	27.50	30.83	32.50	33.33

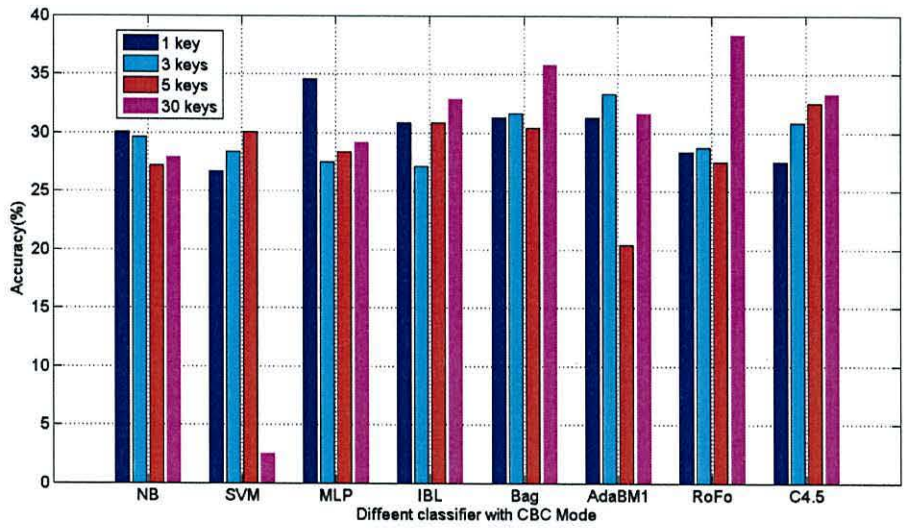


Fig. 5.15: Encryption classification accuracy with the 4 classes dataset with CBC mode with different numbers of key.

Using 16-bit codes

An experiment using 16-bit was performed in the same manner with CBC mode as above. Table 5.14 shows the classification accuracy result of the six classifiers (Naive Bayes, SVM, IBL, Bag, AdaBM1 and C4.5) for the 8 classes dataset. According to Figure 5.16, the results show that IBL at 13% had the highest classification accuracy but only 31 instances out of 240 were correctly classified. On the other hand, Naive Bayes at 10.83% obtained the lowest accuracy but only 25 instances out of 240 were correctly classified.

However, with the 4 classes dataset, the highest accuracy obtained was with AdaBM1 where the classifiers outperformed all the other classifiers with a classification accuracy of 33.75%. It classified 81 instances correctly out of 240 while the lowest accuracy was C4.5 at 29.58% meaning 71 out of 240 were correctly classified.

Tab. 5.14: *Classification accuracy performance of the classifier four-block cipher with CBC mode.*

Block cipher Datasets / Algorithms	8 Classes 65536 Attributes, 240 Instances (%)	4 Classes 65536 Attributes, 240 Instances (%)
Naive Bayes	10.39	32.90
SVM	12.50	31.67
MLP	No result	No result
IBL	13.00	30.04
Bag	12.08	31.67
AdaBM1	12.08	33.75
RoFo	No result	No result
C4.5	10.83	29.58

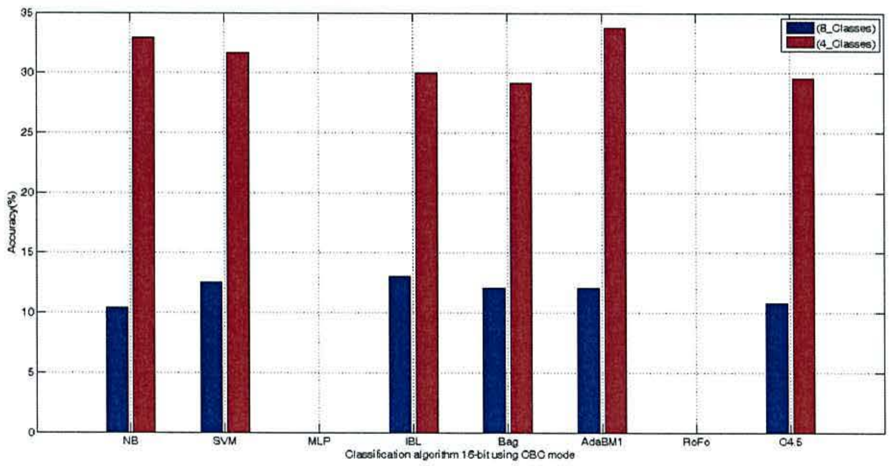


Fig. 5.16: The accuracy for each classifier with CBC mode 16-bit codes.

Effects of Different Instances (120, 240 and 400) with CBC Mode

The third experiment in CBC mode deals with the effect of increasing the number of input files (instances) on overall accuracy. The first dataset included 4 classes AES 128-bit, DES 64-bit, IDEA 128-bit and RC2 128-bit each with 30 input files. The second dataset included 8 classes each with 30 data points, AES (128, 192, 256), DES 64, IDEA 128, RC2(42, 84, 128); and the third dataset included 4 classes were 100 input files were used (AES-128, DES-64, IDEA-128 and RC2-128). Thus, the total numbers of the instances were 120, 240 and 400 respectively. Table 5.15 and Figure 5.17 shows that all classifiers using 400 input files achieved the highest accuracy and when using 120 input files they achieved the lowest accuracy. It is also obvious that AdaBM1 classifier produced the best results for all instances, and C4.5 performed very poorly when operating with 120 and 240 instances.

Tab. 5.15: *Using different instances with the same features (120, 240 and 400) with CBC mode.*

Algorithms	120 Instances (%)	240 Instances (%)	400 Instances (%)
Naive Bayes	25.00	27.50	30.83
SVM	26.66	27.75	35.83
MPL	28.00	28.25	33.33
IBL	25.83	22.50	29.58
Bag	26.00	25.25	28.33
AdaBM1	25.00	26.00	36.25
RoFo	27.00	26.75	31.67
C4.5	23.00	25.25	28.33

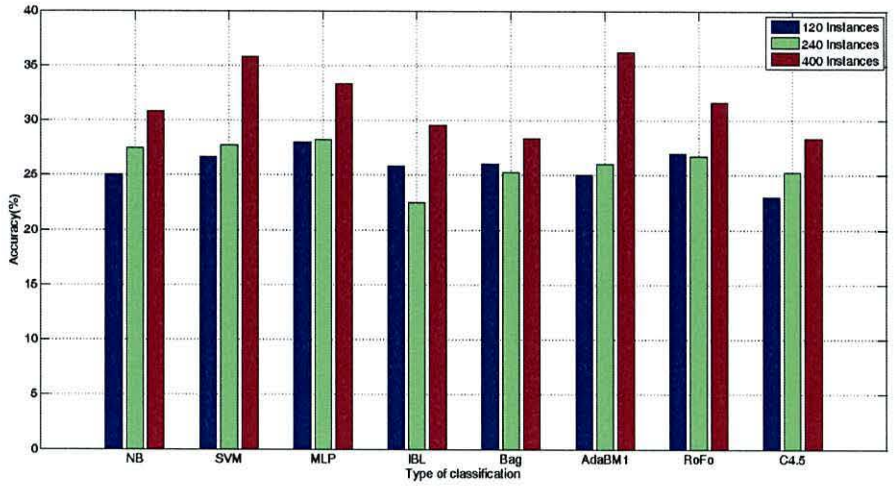


Fig. 5.17: Encryption classification accuracy with CBC mode using different instances for block ciphers.

RC2, RC6, Blowfish, Twofish, XTA, CAST and DESede (128-bit) Algorithms

In the final experiment, the effect of encrypting the plain-text was investigated for the following encryption algorithms: RC2, RC6, Blowfish, Twofish, XTA, CAST and DESede. The number of input files was 30 from the BECC with the number of keys also being 30. Here, Figure 5.18 shows that the Bag classifier achieved the best overall accuracy performance and that the IBL had the lowest accuracy. It was found that using 128-bit for each algorithm resulted in the highest classification accuracy.

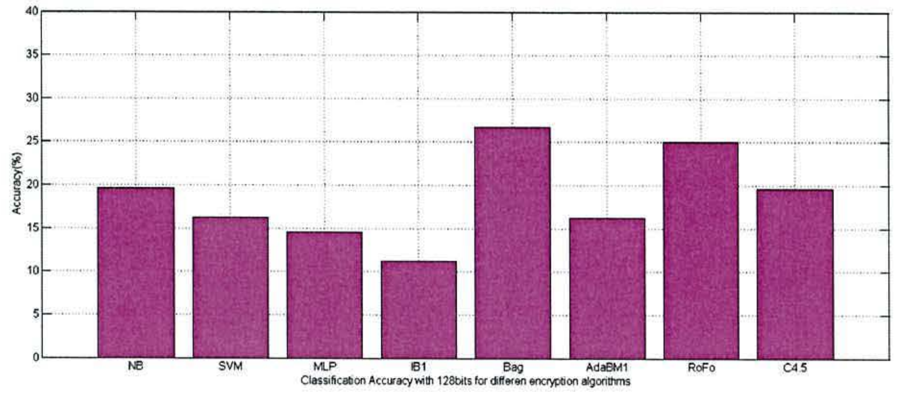
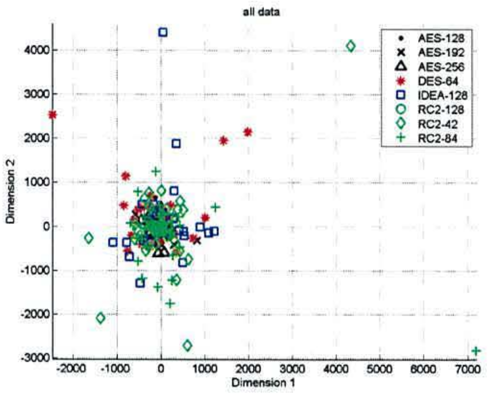


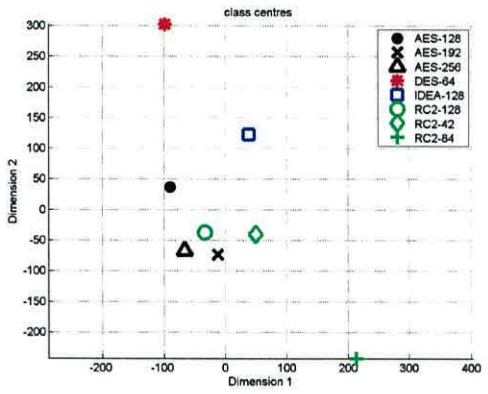
Fig. 5.18: Encryption classification accuracy RC2, RC6, Blowfish, Twofish, XTA, CAST and DESede algorithms.

Multidimensional Scaling (MDS) with 8-bit and 16-bit with ECB and CBC Modes.

Figure 5.19 (a) shows the scatter-plots of the 240 data points within two dimensions for the ECB mode with the 8 encoding. The 8 classes are plotted with different markers.



(a) All data



(b) Class centres

Fig. 5.19: Scatter-plots of the 240 data points and the centres for the ECB mode with the 8 encoding.

The scatter-plot indicates that the classes are highly overlapping, suggesting that the high recall and precision rates for AES (Table 5.8) are only possible in higher dimensions. The centres of the "clouds" of points for the (8 classes) are plotted in Figure 5.19 (b). According to this scatter-plot, AES has similar representation to that of RC2(42) and RC2(84). Note that the scales of the two scatter plots are different. The class centres are indistinguishable if plotted on the axes of the subplot (a). This highlights the difficulty in recognising the type of code through simple pattern classification algorithms.

Further insights about the relationship of the 8 methods can be gained by constructing and plotting a Distance Matrix. The *(i,j)*-th entry of Distance Matrix is the distance between objects *i* and *j* in the original multidimensional space. In this study, the Distance Matrix is of size 240-by-240. The Matrix can be thought of as consisting of 8-by-8 blocks, each block corresponding to a pair of encoding methods. Each such block is itself a Matrix of size 30-by-30. For example, the block sitting at the top right corner of Distance Matrix will contain the distances by the 30 messages encoded by AES-128 and then encoded by RC2-128. For the codes to be distinguishable, they have to exhibit high similarity within their "own" blocks and low similarity with other codes.

Figure 5.20 shows an image of the Distance Matrix for ECB mode with 8-bit coding. The blocks of 30-by-30 distances are outlined with black lines. The blue colour indicates high similarity while yellow and red indicate low similarity. The encoding method that stands out from the rest is AES. The 3-by-3 block Submatrix in the top left corner is largely blue, showing the similarity within the code. Apart from RC2(128), it is different from all other codes. The class labels are as follows: 1 AES (128), 2 AES (192), 3 AES (256), 4 DES (64), 5 IDEA (128), 6 RC2 (128),7 RC2 (42) and 8 RC2 (84).

This suggests that the AES encoding can be distinguished from the remaining codes. The three versions of AES (128, 194 and 256) are not distinguishable within AES, which is visible from the dark blue colour of the respective blocks.

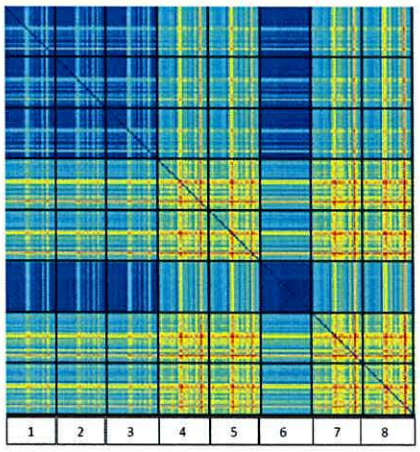
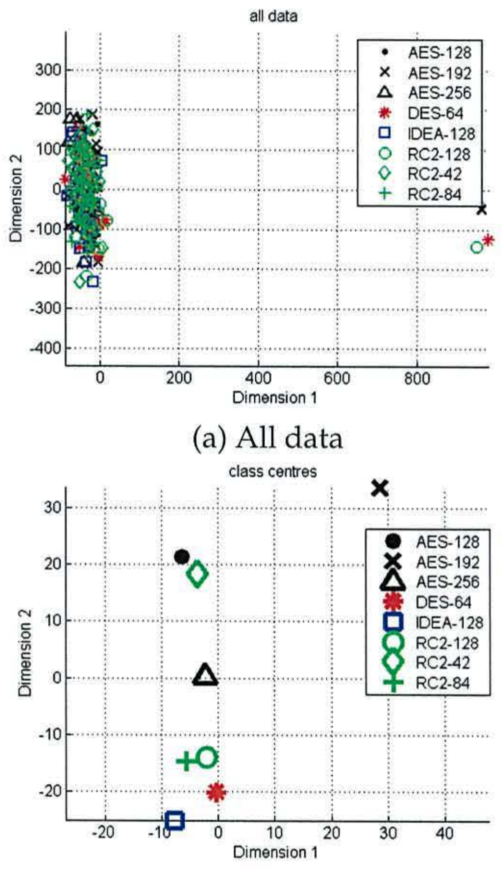


Fig. 5.20: The image of the Distance Matrix for ECB mode with the 8-bit encoding. The class labels are as follows: 1 AES (128), 2 AES (192), 3 AES (256), 4 DES(64), 5 IDEA (128), 6 RC2 (128), 7 RC2(42) and 8 RC2(84)

Interestingly, the largest within-block consistency is demonstrated by RC2 (128-bit) (uniform dark blue within the block), while this does not match the other versions of the same encoding (RC2 (42-bit) and RC2 (84-bit)).

Figures 5.21 to 5.25 show the scatter-plots and the distance matrices for encodings

CBC mode with 8-bit , ECB mode with 16-bit and CBC mode with 16-bit encoding. An interesting finding in Figure 5.21 (a) for CBC mode with the 8-bit coding is the cluster of outliers to the right of the main cluster. Further analyses showed that the points in this cluster do not come from a single message. The large discrepancy of the differences skewed the colour plot of the respective Distance Matrix as seen in Figure 5.22. The red vertical and horizontal lines demonstrate the unusually large distances compared to the rest. Unlike ECB mode, there is no clear pattern to suggest that any of the codes are distinguishable.



(a) All data

(b) Class centres

Fig. 5.21: Scatter-plots of the 240 data points and the centres for the CBC mode with the 8-bit encoding.

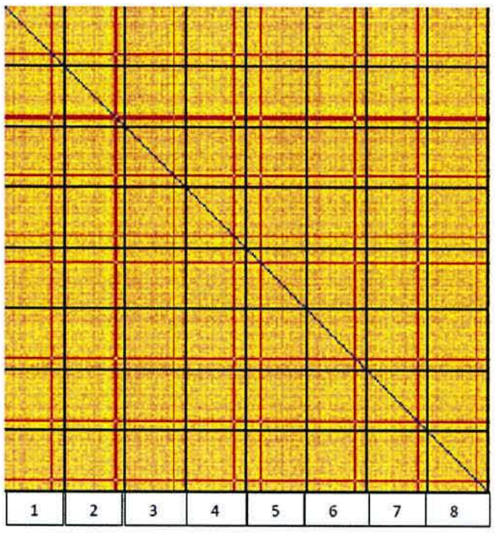
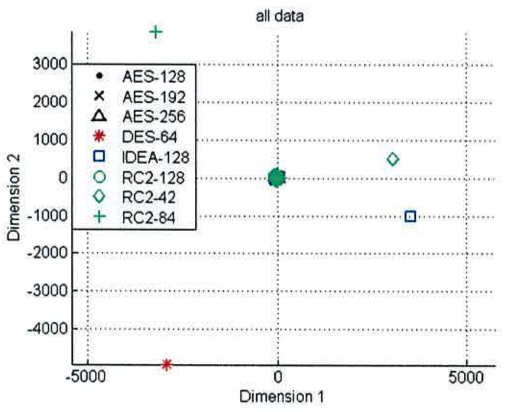
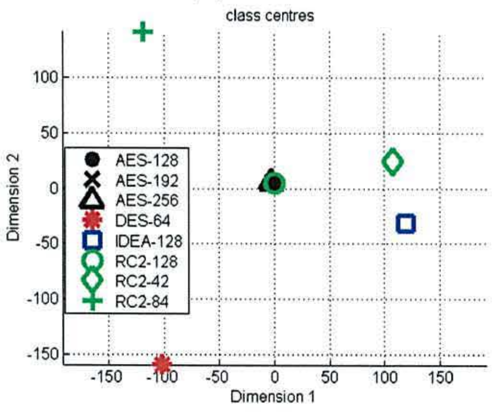


Fig. 5.22: The image of the Distance Matrix for CBC mode with the 8-bit encoding. The class labels are as follows: 1 AES (128), 2 AES (192), 3 AES (256), 4 DEA (64), 5 IDEA (128), 6 RC2 (128), 7 RC2 (42) and 8 RC2 (84).



(a) All data



(b) Class centres

Fig. 5.23: Scatter-plots of the 240 data points and the centres for the ECB mode with the 16-bit encoding.

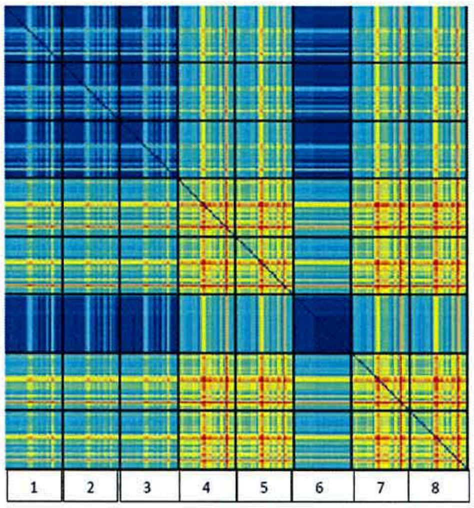
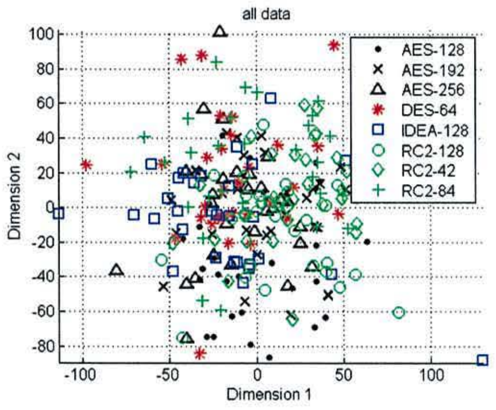
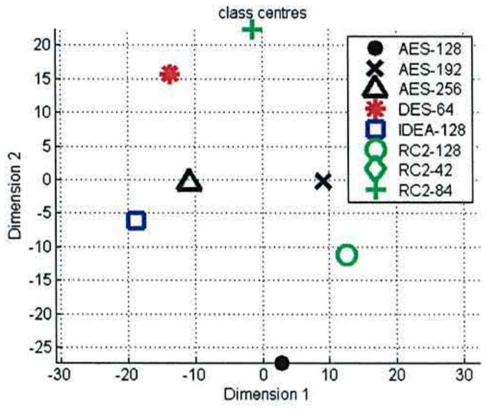


Fig. 5.24: The image of the Distance Matrix for ECB mode with the 16-bit encoding. The class labels are as follows: 1 AES (128), 2 AES (192), 3 AES (256), 4 DEA (64), 5 IDEA (128), 6 RC2 (128), 7 RC2 (42) and 8 RC2 (84).



(a) All data



(b) Class centres

Fig. 5.25: Scatter-plots of the 240 data points and the centres for the CBC mode with the 16 encoding.

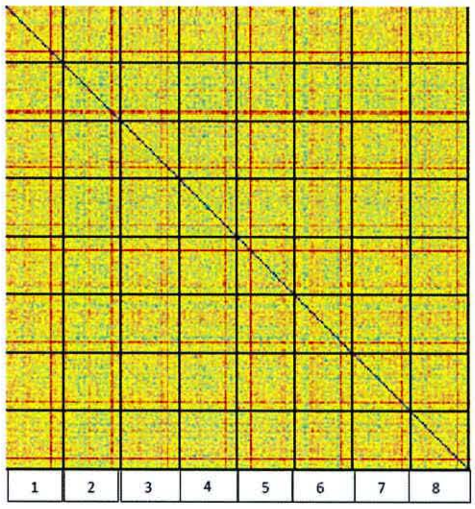


Fig. 5.26: The image of the Distance Matrix for CBC mode with the 16-bit encoding. The class labels are as follows: 1 AES (128), 2 AES (192), 3 AES (256), 4 DEA (64), 5 IDEA (128), 6 RC2 (128), 7 RC2 (42) and 8 RC2 (84).

5.5 Comparisons Between ECB and CBC Modes Using 8-bit and 16-bit encoding

The purpose of this section is to compare the eight algorithms in both ECB and CBC modes when different key lengths (8-bit and 16-bit) were used.

5.5.1 Using 8-bit codes with ECB and CBC Modes

In the first experiment, the eight classifiers (Naive Bayes, SVM, MLP, IBL, Bag, AdaBM1, RoFo and C4.5) were compared in ECB and CBC modes with 8-bit codes. Table 5.16 and Figure 5.27 shows the classification accuracy result and the comparison between both modes. According to the experimental results in ECB mode, RoFo at 53.33% had the highest classification accuracy but in CBC mode no accuracy was obtained because there were too many features.

In ECB mode, MLP obtained 13.33% but again, in CBC mode, no accuracy was obtained. In the second experiment, the same eight classifiers were compared in ECB and CBC modes with 16-bit codes, as shown in Figure 5.28. According to the results, Naive Bayes in ECB mode obtained the highest accuracy and MLP and RoFo in both modes were not obtained for the same reason as above. In addition, the results show that the ECB mode was more accurate than the CBC mode, which was not expected, because the CBC mode is key-chaining. In the authors view this was dependant on the types of classifier rather than on the types of the modes.

Tab. 5.16: *Classification accuracy performance of the classifier four-block cipher in ECB and CBC modes using 8-bit with 8 classes and 4 classes datasets.*

Block cipher Datasets / Algorithms	8 Classes 256 Attributes, 240 Instances ECB (%)	4 Classes 256 Attributes, 240 Instances ECB (%)	8 Classes 256 Attributes, 240 Instances CBC (%)	4 Classes 256 Attributes, 240 Instances CBC (%)
Naive Bayes	28.33	44.17	12.90	30.33
SVM	17.08	32.08	15.00	35.83
MLP	21.67	39.58	10.83	33.33
IBL	12.50	30.42	11.25	29.58
Bag	25.83	47.50	10.83	28.33
AdaBM1	20.42	43.33	14.58	36.67
RoFo	30.83	53.33	10.83	31.67
C4.5	27.92	51.67	10.42	28.33

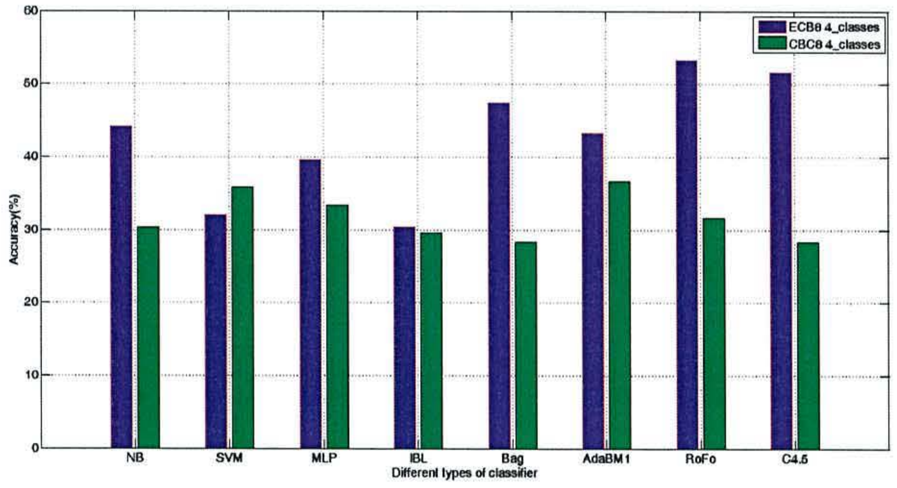


Fig. 5.27: The accuracy for each of the classifiers with ECB and CBC modes.

5.5.2 Using 16-bit Codes with ECB and CBC Modes

In the third experiment, the same classifiers were compared in ECB mode with 16-bit. Table 5.16 shows that Naive Bayes has the highest accuracy of 29.17% with 70 instances out of 240 were correctly classified with the 4 classes dataset.

Table 5.17 shows that with CBC mode, AdaBM1 has the highest accuracy of 14.58% only were 35 instances out of 240 were correctly classified for the 8 classes dataset, which was also the case for the 4 classes dataset, although the percentage was 36.25% were only 87 instances out of 240 were correctly classified. The experiment showed that using 8-bit attributes had more accuracy than using 16-bit attributes.

Tab. 5.17: *Classification accuracy performance of the classifier four-block cipher with ECB and CBC modes.*

Block cipher Datasets / Algorithms	8 Classes 65536 Attributes, 240 Instances ECB (%)	4 Classes 65536 Attributes, 240 Instances ECB (%)	8 Classes 65536 Attributes, 240 Instances CBC (%)	4 Classes 65536 Attributes, 240 Instances CBC (%)
Naive Bayes	29.17	57.92	10.39	32.90
SVM	13.33	36.25	12.50	31.67
MLP	No result	No result	No result	No result
IBL	12.50	37.50	13.00	30.04
Bag	17.92	41.25	12.08	31.67
AdaBM1	15.00	39.58	12.08	33.75
RoFo	No result	No result	No result	No result
C4.5	20.00	38.75	10.83	29.58

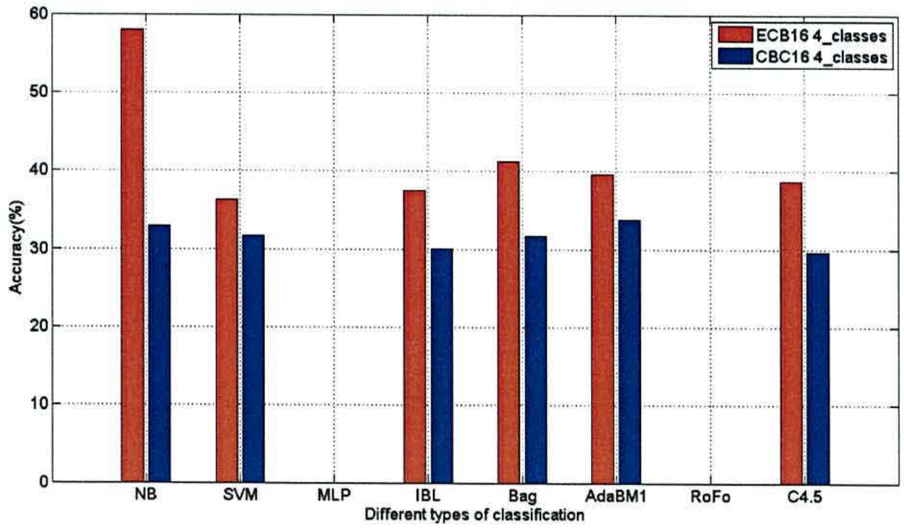


Fig. 5.28: The accuracy for each of the classifiers with ECB and CBC modes.

5.5.3 Finding the Most Accurate Classification Results with ECB and CBC Modes

Classification Results for 8-bit Codes

According to the results, RoFo shows the most accurate classification table of all the algorithms. Table 5.18 (a) below, a confusion matrix, is a representative set of results from 8 classes of RoFo classifications. The table includes 240 data points for each block cipher algorithm (AES, DES, IDEA and RC2) in ECB mode. Each row in the confusion matrices corresponds to the known class of the data, and each column represents the predicted classifications. Table 5.18 (b) 4 classes shows that the model correctly predicted the positive classes for AES, 64 times and incorrectly 26 times, for DES, 8 times correctly and 22 times incorrectly, for IDEA, 6 times correctly and 24 times incorrectly and for RC2, 50 times correctly and 40 times incorrectly. The resulting accuracy was 53.33% (128 out of 240 correctly classified).

Predicted class / Actual class	AES-128	AES-192	AES-256	DES-64	IDEA-128	RC2-128	RC2-42	RC2-84
AES-128	10	4	6	0	1	7	2	0
AES-192	5	8	7	0	0	8	0	2
AES-256	9	9	6	0	0	3	0	3
DES-64	0	0	0	8	11	0	2	9
IDEA-128	0	0	0	8	6	0	7	9
RC2-128	3	4	2	0	0	21	0	0
RC2-42	1	0	0	5	9	0	9	6
RC2-84	0	0	0	10	6	0	8	6

(a) Confusion matrix for RoFo classifier (8 classes).

Predicted class / Actual class	AES	DES	IDEA	RC2
AES	64	0	1	25
DES	0	8	11	11
IDEA	0	8	6	16
RC2	10	15	15	50

(b) Confusion matrix for RoFo classifier (4 classes)).

Tab. 5.18: RoFo Confusion matrix using 8-bit.

Classification Results for 16-bit Codes

The Naive Bayes classifier with ECB mode, as shown in Table 5.19 (a), was the most accurate for 8 classes for all encryptions with different encryption keys. Table 5.19 (b) 4 classes, which is a combination of all encryption key algorithms, shows that the model correctly predicted the positive class for AES, 64 times and incorrectly predicted it 26 times, for DES, 8 times correctly and 22 times incorrectly, for IDEA, 6 times correctly and 30 times incorrectly and for RC2, 50 times and 40 times incorrectly. The resulting accuracy was 57.92% (139 out of 240 correctly classified).

Actual class \ Predicted class	AES-128	AES-192	AES-256	DES-64	IDEA-128	RC2-128	RC2-42	RC2-84
AES128	11	10	6	0	0	1	0	2
AES-192	11	9	6	1	0	1	0	2
AES-256	14	4	7	0	0	2	3	0
DES-64	0	0	0	10	11	0	2	7
IDEA-128	0	0	0	13	5	0	6	6
RC2-128	1	3	4	0	1	20	0	1
RC2-42	0	0	0	6	10	0	3	11
RC2-84	0	0	0	10	9	0	6	5

(a) Confusion matrix for Naive Bayes classifier (8 classes)

Actual class \ Predicted class	AES	DES	IDEA	RC2
AES	78	1	0	11
DES	0	10	11	9
IDEA	0	13	5	12
RC2	8	16	20	46

(b) Confusion matrix for Naive Bayes classifier (4 classes))

Tab. 5.19: Naive Bayes confusion matrix for the 16-bit codes.

5.5.4 Execution Time to Build the Model

Execution Times to Build the Model with ECB Mode

Table 5.20 shows the results of the comparison that was conducted with respect to the time taken to build the model, with the two key lengths, 8-bit and 16-bit codes for all 8 classes. It can be observed that the C4.5 classifier has the highest time requirement to build the model in all experiments. In addition, IBL and Naive Bayes classifiers need less time requirement to build the model in ECB mode.

Tab. 5.20: *Time taken to build the model with 8-bit and 16-bit in the ECB mode.*

Algorithms	8-bit (seconds)	16-bit (seconds)
Naive Bayes	0.12	4.10
SVM	2.48	16.54
IBL	0.01	0.35
Bag	2.82	132.12
AdaBM1	0.14	4.86
C4.5	0.90	57.01

Execution Times to Build the Model with CBC Mode

Table 5.21 shows the results of the comparison that was conducted with respect to the time taken to build the model, along with the two key lengths, for 8-bit and 16-bit codes for all 8 algorithms. Again, it can be observed, that the C4.5 classifier has the highest time requirement to build the model in all experiments.

Tab. 5.21: Time taken to build the model with 8-bit and 16-bit codes with CBC mode.

Algorithms	8-bit (seconds)	16-bit (seconds)
Naive Bayes	0.03	4.02
SVM	1.86	16.38
IBL	0.00	0.35
Bag	1.68	123.83
AdaBM1	0.05	4.61
C4.5	0.72	143.97

Summary

In this chapter, Pattern Recognition was found to be a useful tool to identify the encryption mode and classification of encrypted plain-text files.

The main purpose of this chapter was to show the impact on the classification accuracy of the different key sizes (1, 3, 5 and 30) with different input data sizes (120, 240 and 400 instances), as well as using different Symmetric cipher algorithms (block cipher algorithms) with different encryption key sizes, and in ECB and CBC modes. In addition, two datasets using 8 classes and 4 classes with 8-bit and 16-bit codes were used.

In this study, the accuracy for the eight classifiers (Naive Bayes, SVM, MPL, IBL, Bag, AdaBM1, RoFo and C4.5) was evaluated. The experiments were performed using the WEKA Machine Learning Platform. The aim was to find the best classification algorithm with the highest accuracy for four different block ciphers for the first group that includes: DES, IDEA, AES and RC2; and for the second group that includes: RC2, RC6, Blowfish, Twofish, XTA, CAST and DESede (128-bit). Experiments were conducted to identify encryption algorithms of encrypted data using a variety of classifiers. First the results of the experiments show that Pattern Recognition techniques are useful tools for cryptanalysis as a means of identifying the type of encryption algorithm used to encrypt the data.

This work shows, that as expected, increasing the number of encryption keys will result in reducing the classification accuracy. The results show that it is possible to achieve an accuracy above 40% with some classifiers when each file is encrypted with different numbers of keys. It was also clear that increasing the number of files used also improves accuracy.

In the second experiment, the keys used were different for each text data. These results show that the RoFo classifier had the best performance when identifying the encryption method for ciphered data, while IBL performance was the worst. Furthermore, the performance of the classifiers improved significantly when identification of 4 classes (encryption) was considered. It was noted that the three versions of AES (128, 192 and 256) were not distinguishable within AES. Further, RC2 (128-bit) does not match the other versions of the same encoding

RC2 (42, 84-bit).

The performance of each of the classifiers was presented, and the experimental results show that in general, the RoFo classifier has the highest classification accuracy. As a result, it was considered that Pattern Recognition could be used as a useful tool for accuracy.

Finally, as expected, the CBC mode needs more processing time than the ECB due to its key-chaining nature. However, in terms of using both RoFo and MLP classifiers with 16-bit, the result was not accurate since there were too many features, despite using a 64-bit Java Virtual Machine (JVM) and 6GB of RAM. An interesting point was that the ECB mode obtained higher accuracy classification than the CBC mode, which was not expected due to key chaining.

Chapter 6

Encryption Classification for Stream Cipher Algorithms

The purpose of this chapter is to use of Pattern Recognition (PR) techniques for identification of the encryption method used from the encrypted plain-text file for stream ciphers. Different classifiers were used for identification and the study also provides a comparison between stream and block cipher algorithms. An overview of the the study in Section 6.1. Section 6.2 explains the methodology and Section 6.3 addresses identification encryption method and Section 6.4 describes the experimental setting. Section 6.5 compares the classifier results between stream and block cipher algorithms. The chapter concludes with a summary.

6.1 Introduction

In cryptography, one can distinguish between a block and stream algorithm. The stream cipher-text could be any length whereas the block cipher have to be in increments of the block sizes. Moreover, distinguishing between different types of stream cipher is far more difficult but possible under further assumptions. The basic point is to remember is that stream cipher algorithms can encrypt data of any size and does not require that the size is known in advance. Also the same algorithm is applied to encrypt and decrypt the data.

6.2 Methodology

In this section, the same methodology were used as has been used in the previous chapter. The encryption was carried out by using Crypto++ library. Again a Matlab program was used to build the WEKA file. The classifiers used in WEKA included: Naive Bayes (NB), SVM, IBL, AdaBM1, RoFo and C4.5 classifiers. In this experiment, first 8-bit codes file data points were selected and then 16-bit code file data points were selected. The aim was to compare between them to determine which one provides better accuracy.

6.3 Identification of Encryption Method

This section presents the results of the experiments that were conducted in order to study the performance of the proposed classification method. The identification of encryption method for input stream ciphers are presented with each data input file being a data point in our dataset. The study included the following five different stream cipher algorithms and four block cipher algorithms:

- Stream cipher algorithms: Grain 128-bit, Hc 128-bit, RC4 128-bit, Salsa20 128-bit and VMPC 128-bit.
- At the end of the chapter, the result are compared with the following block cipher algorithms: AES (128, 192 and 256-bit), DES 64-bit, IDEA 128-bit and RC2(42, 84 and 128-bit).

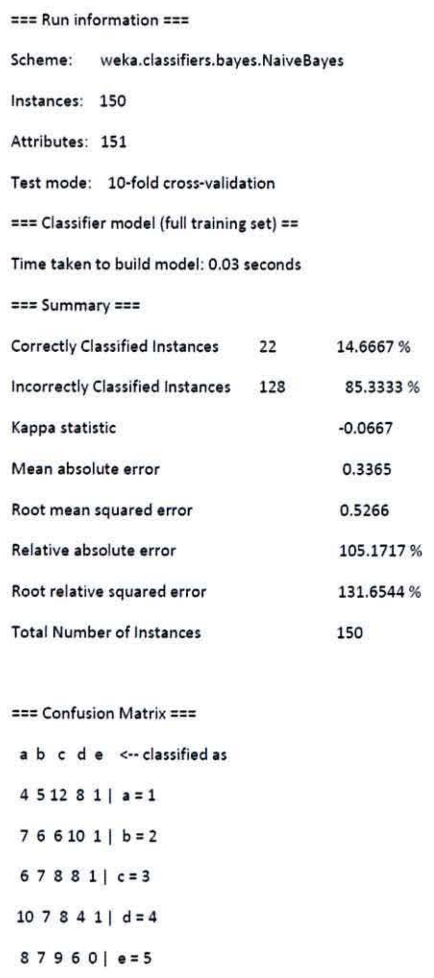
In this experiment, 8-bit character coded data files were selected, and Matlab was used to extract histograms of the encrypted dataset. We also used the same method with 16-bit character coded files. For these experiments, we have selected the 30 files from the BECC dataset (150 data points for stream ciphers and 240 data points for block ciphers) to produce 150 and 240 data points that can be found in the DVD. The encryption was carried out by using the Crypto++ library. Then a Matlab program was used to create the WEKA input files. For more information about how these were created files, see Section 4.5 in Chapter 4.

6.4 Experimental Results

This section explains the results of the experiments for the evaluation of the classifier encrypted text using stream cipher algorithms and block cipher algorithms for comparison. The training set used for the experiments one class used for stream cipher algorithms later on these are compared with two classes for block ciphers algorithms. The testing set was used to identify the unknown stream and block that were to be classified. Ten fold cross-validation was used and all classifiers were trained using the same training sets and were tested using the same testing sets to establish the classification.

We have also used the confusion matrices find the highest accuracy and the Multidimensionality Scaling (MDS) method was used to find the similarities and dissimilarities between algorithms. The equation in section 5.2.2 was used and applied to all classifiers (Naive Bayes, SVM, MLP, IBL, Bag, AdaBM1, RoFo and C4.5).

Figure 6.1 shows sample output from WEKA for the Naive Bayes classifier. The 150 BECC encryption text files was used as data points for the training and testing sets as in chapter 5. The following is a sample of Naive Bayes classifier, which was one of the eight different classifier that were investigated. The Naive Bayes classifier obtained 14.67% accuracy with 22 times out of 150 correctly classified and 85.33% accuracy with 128 out of 150 times incorrectly classified.



```
=== Run information ===

Scheme:    weka.classifiers.bayes.NaiveBayes

Instances:  150

Attributes:  151

Test mode:  10-fold cross-validation

=== Classifier model (full training set) ==

Time taken to build model: 0.03 seconds

=== Summary ===

Correctly Classified Instances       22          14.6667 %

Incorrectly Classified Instances     128          85.3333 %

Kappa statistic                                  -0.0667

Mean absolute error                               0.3365

Root mean squared error                           0.5266

Relative absolute error                          105.1717 %

Root relative squared error                      131.6544 %

Total Number of Instances                        150


=== Confusion Matrix ===

 a  b  c  d  e   <-- classified as

 4  5 12  8  1 |  a = 1

 7  6  6 10  1 |  b = 2

 6  7  8  8  1 |  c = 3

10  7  8  4  1 |  d = 4

 8  7  9  6  0 |  e = 5
```

Fig. 6.1: Classification result of Naive Bayes classifier from WEKA.

6.4.1 Results of Stream Cipher algorithms

First, experiments were conducted using an 8-bit encoding. For each class, the confusion matrix was used to find the correct and incorrect classifier between different types of stream cipher algorithms. The following describes the experimental results. Tables 6.1 (a), (b), (c) and (d) show the confusion matrices for different types of classifier. Five classes were used with the 30 BECC encryption text files using 30 different key sizes. The table shows that all the classifiers had similar accuracy of around 20%. In addition, the system struggled to distinguish between them due to the randomness of the datasets. In summary, MLP shows the highest accuracy with 21.33%, second was SVM with 20%, third was IBL classifier with 19% and finally NB obtained the lowest accuracy with 14%.

Actual class \ Predicted class	Grain128	HC128	RC4-128	Salsa20	VMPC
Grain128	4	10	7	8	1
HC128	9	3	8	10	0
RC4-128	3	9	7	10	1
Salsa20	9	10	7	4	0
VMPC	11	7	7	4	1

(a) Confusion matrix for NB classifier

Actual class \ Predicted class	Grain128	HC128	RC4-128	Salsa20	VMPC
Grain128	9	11	3	7	0
HC128	11	7	3	9	0
RC4-128	7	13	6	4	0
Salsa20	5	12	5	8	0
VMPC	8	12	4	6	0

(b) Confusion matrix for SVM classifier

Actual class \ Predicted class	Grain128	HC128	RC4-128	Salsa20	VMPC
Grain128	2	12	6	3	7
HC128	3	12	6	3	6
RC4-128	1	13	7	3	6
Salsa20	1	13	5	5	6
VMPC	2	13	6	3	6

(c) Confusion matrix for MLP classifier

Actual class \ Predicted class	Grain128	HC128	RC4-128	Salsa20	VMPC
Grain128	7	10	5	5	3
HC128	6	8	6	5	5
RC4-128	5	12	4	7	2
Salsa20	5	8	9	5	3
VMPC	7	8	5	5	5

(d) Confusion matrix for IBL classifier

Tab. 6.1: Confusion matrix for Naive Bayes, SVM, MLP and IBL classifiers.

Tables 6.2 (a), (b), (c) and (d) show similar results with similar difficulties in distinguishing between the algorithms. The reason for the similar results again is due to the random nature of the dataset as mentioned before.

Predicted class / Actual class	Grain128	HC128	RC4-128	Salsa20	VMPC
Grain128	6	11	3	5	5
HC128	7	7	5	4	7
RC4-128	4	5	6	7	8
Salsa20	8	6	8	3	5
VMPC	3	5	6	9	7

(a) Confusion matrix for Bag classifier

Predicted class / Actual class	Grain128	HC128	RC4-128	Salsa20	VMPC
Grain128	26	2	1	0	1
HC128	23	3	1	3	0
RC4-128	21	2	0	6	1
Salsa20	21	2	0	6	1
VMPC	18	3	1	8	0

(b) Confusion matrix for AdaBM1 classifier

Predicted class / Actual class	Grain128	HC128	RC4-128	Salsa20	VMPC
Grain128	5	10	4	6	5
HC128	9	3	7	5	6
RC4-128	7	6	1	10	6
Salsa20	5	5	7	5	8
VMPC	6	4	6	7	7

(c) Confusion matrix for RoFo classifier

Predicted class / Actual class	Grain128	HC128	RC4-128	Salsa20	VMPC
Grain128	3	6	11	5	5
HC128	6	5	8	3	8
RC4-128	5	5	5	9	6
Salsa20	6	7	6	5	6
VMPC	11	2	5	5	7

(d) Confusion matrix for C4.5 classifier

Tab. 6.2: Confusion matrix for Bag, AdaBM1, RoFo and C4.5 classifiers.

Experiments were also conducted using 16-bit encoding. Tables 6.3 (a), (b), (c) and (d) show similar results with similar difficulties in distinguishing between the algorithms. The highest accuracy obtained was the Naive Bayes classifier with 26.66%. In second place were for the MLP classifier 24.66% and SVM came in third place with 21.33% accuracy, and the lowest accuracy obtained was IBL forth place at 20.66%.

Predicted class / Actual class	Grain128	HC128	RC4-128	Salsa20	VMPC
Grain128	7	11	3	3	6
HC128	10	6	4	6	4
RC4-128	9	3	10	5	3
Salsa20	4	7	7	7	5
VMPC	7	3	5	5	10

(a) Confusion matrix for NB classifier

Predicted class / Actual class	Grain128	HC128	RC4-128	Salsa20	VMPC
Grain128	10	9	3	3	5
HC128	13	4	4	4	5
RC4-128	9	6	4	5	6
Salsa20	7	7	3	9	4
VMPC	11	5	5	4	5

(b) Confusion matrix for SVM classifier

Predicted class / Actual class	Grain128	HC128	RC4-128	Salsa20	VMPC
Grain128	7	12	2	5	4
HC128	8	5	5	6	6
RC4-128	6	5	8	5	6
Salsa20	7	7	2	9	5
VMPC	8	6	4	4	8

(c) Confusion matrix for MLP classifier

Predicted class / Actual class	Grain128	HC128	RC4-128	Salsa20	VMPC
Grain128	8	6	2	6	8
HC128	11	10	0	5	4
RC4-128	11	4	2	8	5
Salsa20	6	7	6	10	1
VMPC	9	10	6	5	3

(d) Confusion matrix for IBL classifier

Tab. 6.3: Confusion matrix for Naive Bayes, SVM, MLP and IBL classifiers.

The results for Bag, AdaBM1, RoFo and C4.5 shown in Tables 6.3 (a), (b), (c) and (d). The highest accuracy obtained was the AdaBM1 classifier with 25.33%. In second place were the RoFo classifier 20% and Bag came in third place with 18.66% accuracy, and the lowest accuracy obtained was C4.5 forth place at 17.33%. According to the confusion matrix results the system can not distinguish between those algorithms. When 16-bit encoding is used.

Actual class \ Predicted class	Grain128	HC128	RC4-128	Salsa20	VMPC
Grain128	5	8	8	4	5
HC128	6	3	9	10	2
RC4-128	4	8	6	8	4
Salsa20	7	4	8	8	3
VMPC	4	5	9	6	6

(a) Confusion matrix for Bag classifier

Actual class \ Predicted class	Grain128	HC128	RC4-128	Salsa20	VMPC
Grain128	30	0	0	0	0
HC128	30	0	0	0	0
RC4-128	28	0	0	0	2
Salsa20	28	0	0	0	2
VMPC	22	0	0	0	8

(b) Confusion matrix for AdaBM1 classifier

Actual class \ Predicted class	Grain128	HC128	RC4-128	Salsa20	VMPC
Grain128	5	7	8	4	6
HC128	14	1	5	4	6
RC4-128	8	4	9	5	4
Salsa20	4	8	6	8	4
VMPC	6	9	6	3	7

(c) Confusion matrix for RoFo classifier

Actual class \ Predicted class	Grain128	HC128	RC4-128	Salsa20	VMPC
Grain128	4	7	8	4	7
HC128	9	3	5	7	6
RC4-128	7	3	8	6	9
Salsa20	5	10	6	5	4
VMPC	4	8	7	5	6

(d) Confusion matrix for C4.5 classifier

Tab. 6.4: Confusion matrix of Bag, AdaBM1, RoFo and C4.5 classifier.

Further experiments were conducted to compare the classification accuracy performance of eight classifiers with the five stream cipher algorithms: Grain128, HC128, RC4, Salsa20 and VMPC algorithms. The aim was to find the best classification algorithm with the highest accuracy for the five stream ciphers.

For this experiment, one dataset was used, which had 5 classes (one for each algorithm) and 150 (BECC) encryption text files were selected as shown in Figure 6.2.

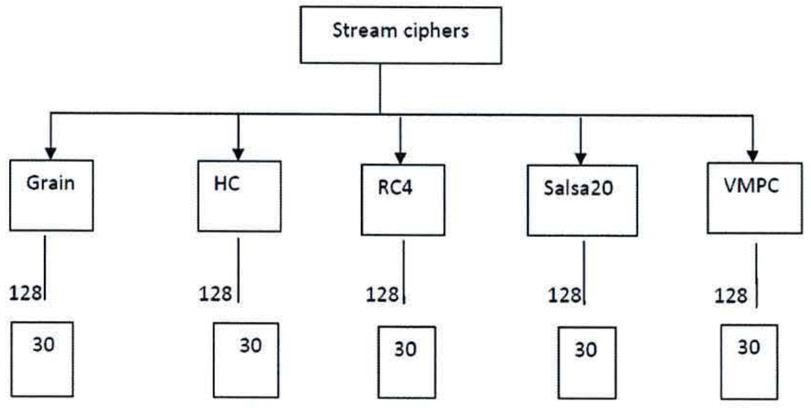


Fig. 6.2: Diagram of stream cipher algorithms with 150 variation as used to produce the data points.

Table 6.5 shows the accuracy of the eight classifiers. Column two represents the dataset 5 classes using 8-bit coding and the third column represents the dataset 5 classes 16-bit coding. Figure 6.3 shows that Naive Bayes was the highest accuracy of 26.67% with 27 instances out of 150 correctly classified for the 16-bit coding. The results for the C4.5 with an accuracy of 17.33%.

However, the IBL and Bag classifiers both had the same accuracy of 19.33% with 29 instances out of 150 being correctly classified. Naive Bayes had lowest accuracy of 14.66% with 9 instances out of 150 were correctly classified. The table also shows that there is a slight different between all classifiers when using 8 bit and 16 bit encodings. For example, AdaBM1 obtained higher accuracy for 8-bit coding. This is due to higher security of stream cipher algorithms since they encrypt each bit and byte separately, unlike block cipher algorithms.

Tab. 6.5: *Classification accuracy performance of the classifier with five stream ciphers.*

Stream cipher Datasets / Algorithms	Dataset with 5 stream cipher 256 Attributes, 150 Instances (%)	Dataset with 5 stream cipher 65536 Attributes, 150 Instances (%)
Naive Bayes	14.66	26.66
SVM	20.00	21.33
MPL	21.33	24.66
IBL	19.33	22.00
Bag	19.33	18.66
AdaBM1	23.33	25.33
RoFo	14.00	20.00
C4.5	16.66	17.33

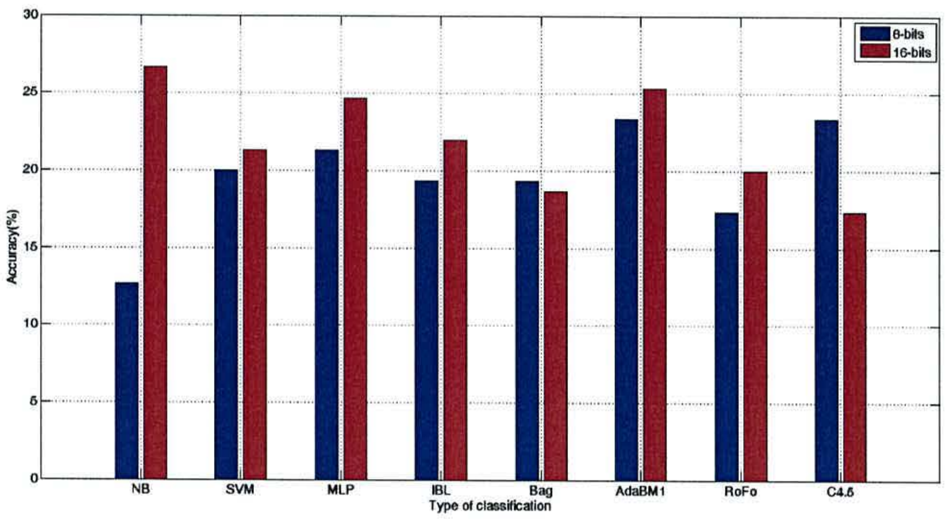
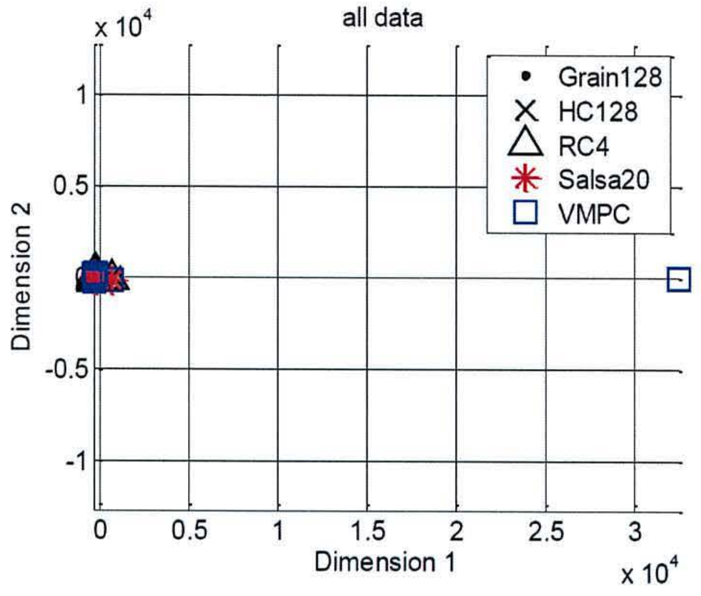


Fig. 6.3: An accuracy of stream cipher algorithms.

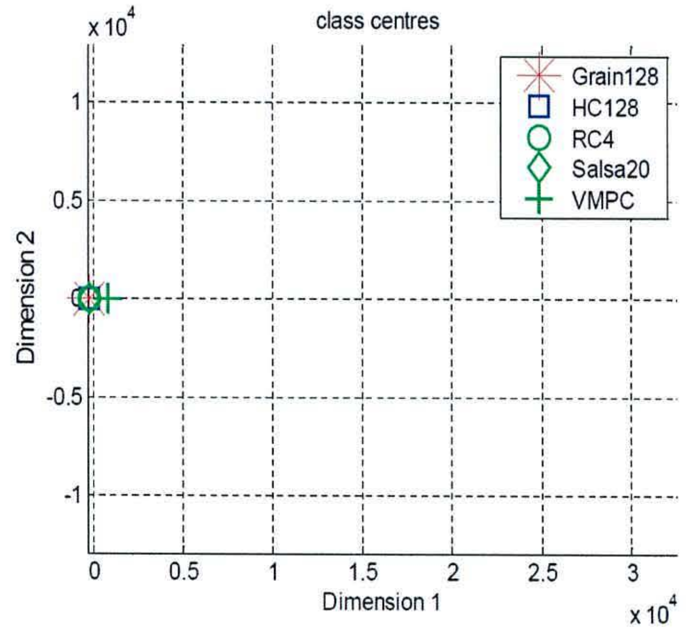
6.4.2 Multidimensional Scaling (MDS)

This section describes results from further investigation into the classification experiments. Figure 6.4 (a) demonstrates the scatter-plots of the 150 BECC encryption text files (data points) in two dimensions for the 8-bit coding. The data is plotted using different markers for the five algorithms.

The scatter-plot shows that the classes are overlapping to a large degree, suggesting that the high recall and precision rates for HC128, Table 6.5, are only possible in higher dimensions. The centres of the "clouds" of points for the five classes are plotted in Figure 6.4 (b). According to this scatter-plot, all algorithms have similar representation. The class centres are indistinguishable if plotted on the axes



(a) All data



(b) Class centres

Fig. 6.4: Scatter-plots of the 150 data points and the class centres for the 8-bit encoding.

of sub-plot Figure 6.4 (a). This highlights the difficulty in recognising the type of encryption through simple classification algorithms.

Additional insights about the relationship of the five algorithms can be gained by constructing and plotting a Distance Matrix . The *(i,j)*-th entry of the Distance Matrix is the distance between objects *i* and *j* in the original multidimensional space. In this study, the Distance Matrix is of size 150-by-150. The matrix can be thought of as consisting of 5-by-5 blocks, each block corresponding to a pair of encoding methods. Each one of these cipher streams is itself a matrix of size 30-by-30. For example, the block sitting at the top right corner of Distance Matrix will contain the distances by the 30 messages encoded by Grain128 and then encoded by VMPC. For the codes to be distinguishable, they have to exhibit high similarity within their "own" blocks and low similarity with other codes.

Figure 6.5 shows an image of the Distance Matrix for the 8-bit encoding case. The blocks of 30-by-30 distances are outlined in black. The blue colour indicates high similarity while red indicates low similarity. The 3-by-3 block sub-matrix in the top left corner is largely blue, showing the similarity within the code. The five algorithms are not distinguishable, which is visible because of the dark blue colour of the respective blocks.

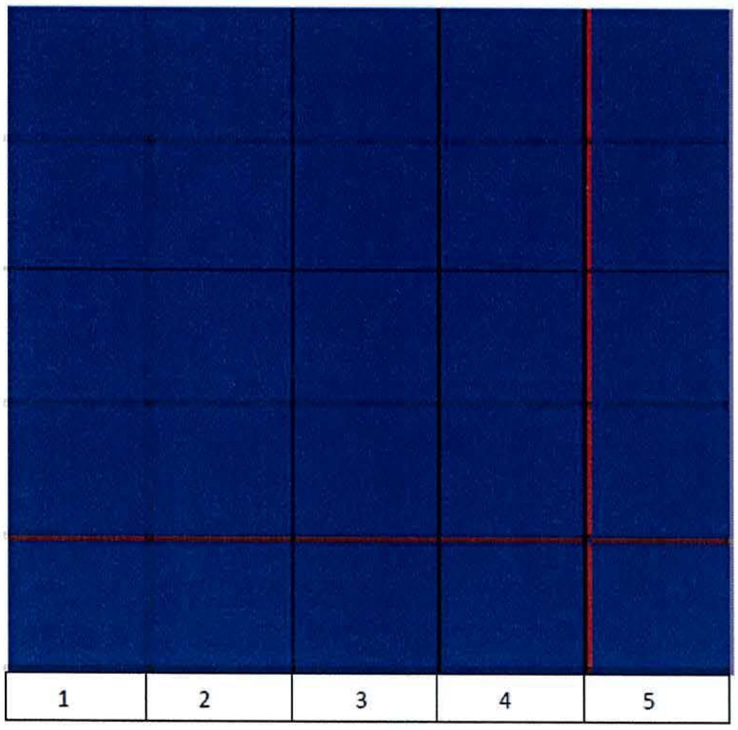
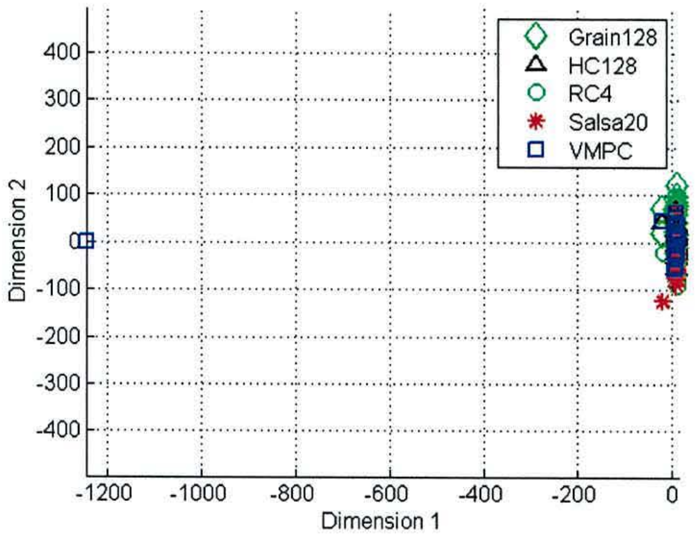
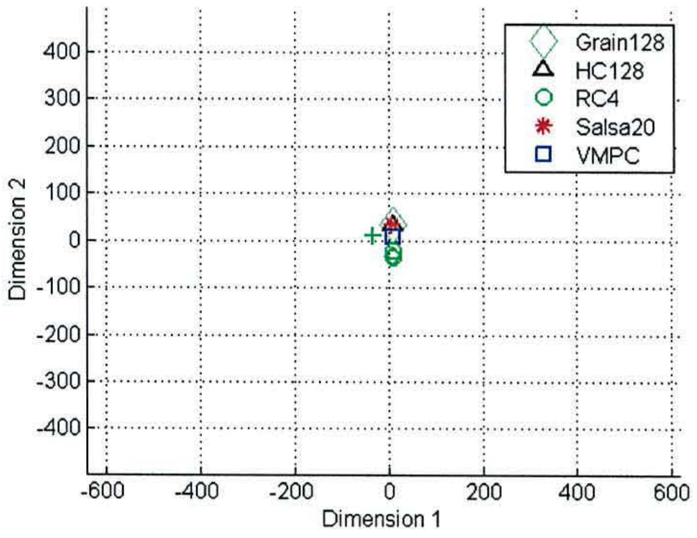


Fig. 6.5: An image of the Distance Matrix for 8-bit encoding. The class labels are as follows: 1 Grain (128), 2 HC (128), 3 RC4 (128), 4 Salsa20 (128) and 5 VMPC (128).

Figure 6.6 (a) the 16-bit coding shows the cluster of outliers to the right of the main cluster. Further analysis results that the points in this cluster do not come from a single message. The large discrepancy of the differences skewed the colour scatter-plot of the respective Distance Matrix, as seen in Figure 6.7. The red vertical and horizontal lines demonstrate the unusually large distances compared to the rest. Unlike the 8-bit coding, there is clear pattern to suggest that none of the codes are distinguishable.



(a) All data



(b) Class centres

Fig. 6.6: Scatter-plots of the 150 data points and the class centres for the 16-bit encoding.

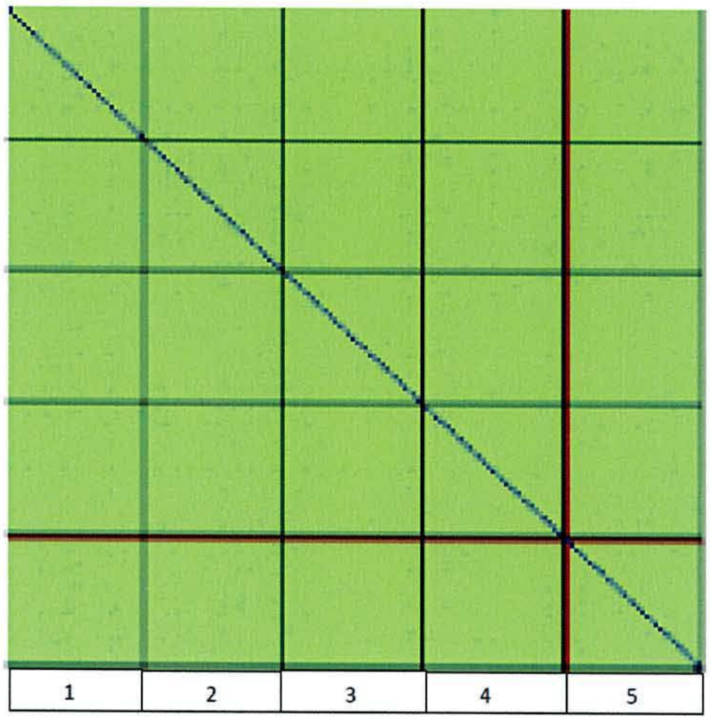


Fig. 6.7: An image of the Distance Matrix for 16-bit. The class labels are as follows: 1 Grain (128), 2 HC (128), 3 RC4 (128), 4 Salsa20 (128) and 5 VMPC (128).

6.4.3 The Most Accurate Classifier Results

For 8-bit codes

According to the results, the AdaBM1 classifier is the most accurate classifier of all the algorithms. Table 6.5 below is a representative set of results from the five classes. The table includes a total of 150 (8×3) BECC encrypted text files for each stream cipher algorithm: Grain128, HC128, RC4, Salsa20, and VMPC. Each row shows the results for each class, and the columns represent the predicted classes. The correct predictions are along the red diagonal; for example, only six of the Salsa20s were correctly classified but only zero of the VMPC were correctly classified. These results also shows that RC4 and VMPC algorithms are distinguishable from the rest.

From the AdaBM1 confusion matrix, in Table 6.6 we see that the accuracy rate is $(38/150) \times 100 = 25.33\%$.

Tab. 6.6: Confusion matrix of AdaBM1 classifier.

Algorithms	Grain128	HC128	RC4	Salsa20	VMPC
Grain128	26	2	1	0	1
HC128	23	3	1	3	0
RC4-128	21	2	0	6	1
Salsa20	21	2	0	6	1
VMPC	18	3	1	8	0

For 16-bit codes

This section describes the results for the 16-bit encoding case. The most accurate classifier was found to be the Naive Bayes classifier as shown in Table 6.7. The correct predictions lie along the red diagonal, which shows that our classification correctly classified seven of the Grain128 algorithms, six of the HC128 algorithms, ten of the RC4 algorithms, seven of the Salsa20 and ten of the VMPC algorithms. This means that both Grain128 and VMPC algorithms are distinguishable from the rest because of their high accuracy, and it impossible to identify the RC4 algorithm due to its low accuracy.

From the Naive Bayes confusion matrix in Table 6.7, we see that the accuracy rate is $(40/150) \times 100 = 26.66\%$.

Tab. 6.7: Confusion matrix of Naive Bayes classifier.

Algorithms	Grain128	HC1284	RC4	Salsa20	VMPC
Grain128	7	11	3	3	6
HC128	10	6	4	6	4
RC4	9	3	10	5	3
Salsa20	4	7	7	7	5
VMPC	7	3	5	5	10

6.5 A Comparison Between the Classification of Stream and Block Cipher Algorithms

This section compares the results for stream ciphers nd block ciphers. Tables 6.8 and 6.9 show that in the block cipher algorithms most of the classifier obtained a higher accuracy compared to stream cipher algorithms. Figure 6.8 shows that RoFo classifiers have a better overall accuracy performance, whereas and RoFo achieved the lowest accuracy with stream ciphers. But Figure 6.9 shows that Naive Bayes with block ciphers have a better accuracy and MLP and RoFo both are none accuracy as mentioned in previous chapter. In general, stream cipher algorithms are more difficult to classify than block ciphers due to random nature of the datasets.

Tab. 6.8: Comparing accuracy between stream and block ciphers using 8-bit coding.

Algorithms	Block ciphers (4 classes) %	Stream ciphers (5 classes) %
Naive Bayes	44.17	14.66
SVM	32.08	20.00
MPL	39.58	21.33
IBL	30.42	19.33
Bag	47.50	19.33
AdaBM1	43.33	23.33
RoFo	53.33	14.00
C4.5	51.67	16.66

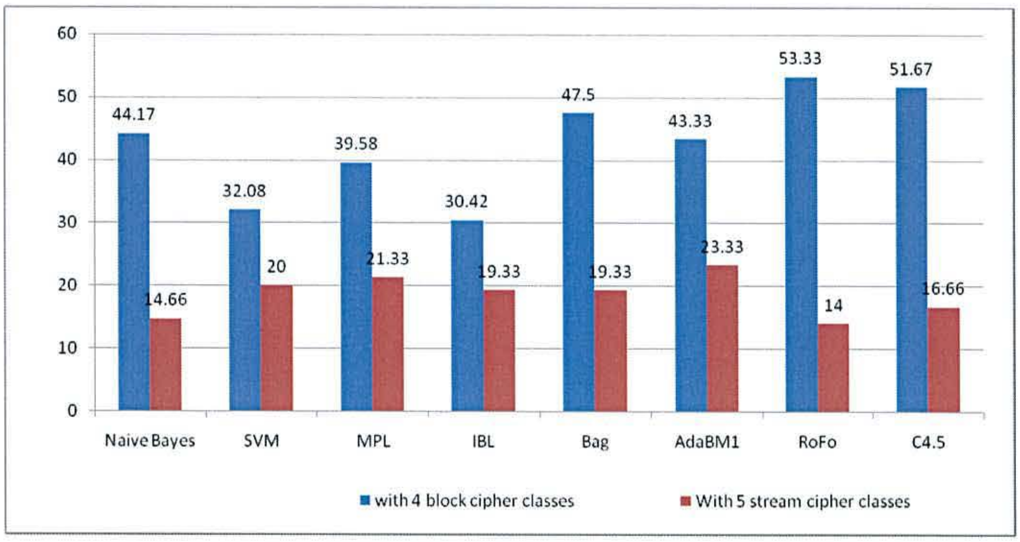


Fig. 6.8: Accuracy result for block and stream ciphers with 8-bit codes.

Tab. 6.9: Comparing accuracy between stream and block for 16-bit codes.

Algorithms	Block ciphers (4 classes) %	Stream ciphers (5 classes) %
NB	57.92	26.66
SVM	32.08	21.33
MPL	None	24.66
IBL	37.50	22.00
Bag	41.25	18.66
AdaBM1	39.58	25.33
RoFo	None	20.00
C4.5	38.75	17.33

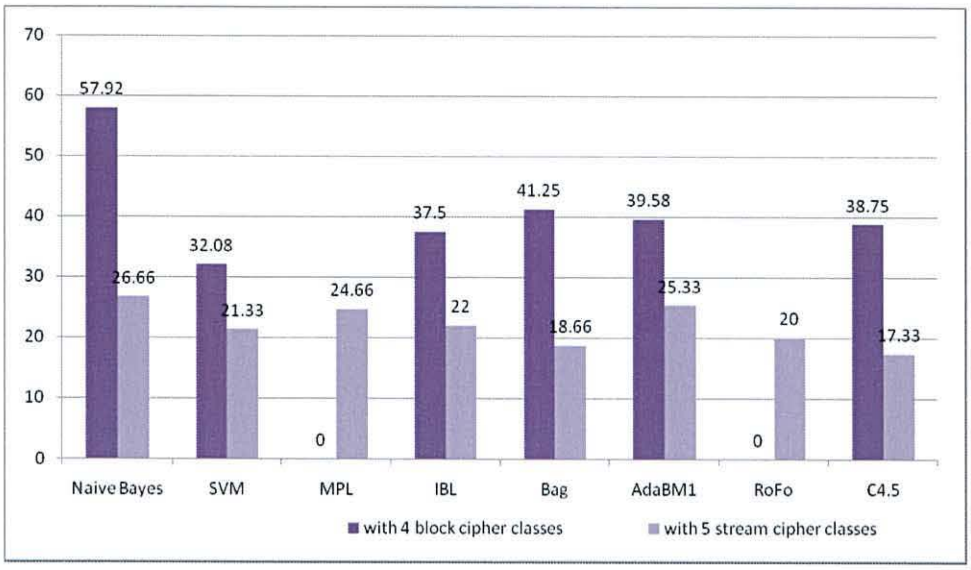


Fig. 6.9: Accuracy result for block and stream ciphers with 16-bit codes.

Summary

The aim of this chapter was to find the most accurate classification algorithm for the five different stream cipher algorithms Grain128, HC128, RC4, VMPC and Salsa20 and then to compare with block cipher algorithms. The results showed that with 8-bit encoding of the encrypted files, the highest accuracy obtained was the AdaBM1 classifier with 23.33% and lowest was the RoFo classifier with 14%. There was only a slight difference between algorithms and essentially the system had difficulty distinguishing between all stream cipher algorithms. For 16-bit encoding, the highest accuracy obtained was for the Naive Bayes classifier with 26.67%, and the lowest accuracy was C4.5 classifier with 17.33%. Again it was observed that the system had difficulty distinguishing between all the algorithms. In addition, when the 8-bit and 16-bit encoding results were compared for all algorithms the difference in accuracy was found to be slight with AdaBM1 classifier had obtained the best accuracy 25.33 %. Further, the five algorithms used with the 16-bit encoding are not distinguishable, unlike with the 8-bit encoding case where there is a clear pattern.

For stream cipher algorithms, we obtained much lower accuracy compared to block cipher algorithms due to the stream cipher algorithms producing encrypted output with less patterns that could be explained for classification by the different classification algorithms. Some of the algorithms were using streaming data which also provided greater security.

In summary, our experimental results showed that stream cipher algorithms are more difficult compared to block cipher algorithms to classify encrypted output. This is due to the bit based streaming approach adopted by the algorithms and the randomly distributed characters that was consequently produced in the encrypted output.

Chapter 7

Conclusion and Future Work

A summary of the most important results and directions for future work are presented in this chapter.

The capability of encryption and decryption information is crucial to secure financial transactions and even elementary forms of on-line privacy. Cryptology has two categories: cryptography and cryptanalysis. Cryptanalysis characteristically involves learning how resistant a cipher is for distinguishing attacks and to recover the key. Most studies in cryptanalysis begin with the hypothesis that encryption and method of operation are already identified. One of the crucial problems in cryptanalysis is identifying the encryption method used. There has been comparative very little work done on this problem using Pattern Recognition classification techniques. The purpose of the research has been to determine whether Pattern Recognition can be used to identify the encryption method.

7.1 Review of Thesis

This thesis investigated the hypothesis that Pattern Recognition classification can be used as a useful tool used to help identify the encryption method used to encrypt data. The experimental results support the hypothesis for block cipher algorithms but not for stream cipher algorithms and highlights the importance of using Pattern Recognition for encryption classification. It provides a reference to guide researchers who are interested in contributing towards the same area in the study of cryptography algorithms. Other researchers can also benefit by using the Bangor Sources Files Corpus (BSFC) and Bangor Encryption Classification Corpus (BECC) that were used for the evaluation.

A brief summary of the dissertation is follows:

Chapter 2 provides an overview of cryptography and cryptanalysis. Cryptographic were divided into two categories: Symmetric and Asymmetric algorithms.

The study focused on Symmetric cipher algorithms, which include block and steam cipher algorithms. This chapter also focused on cryptanalysis, and different types of cryptanalysis.

Chapter3 focused on the use of using cryptographic techniques with different categories and on cryptanalysis with different types of encryption modes. It was observed that using pattern recognition is a useful tool for classification, which was a main focus of our research. Furthermore, this chapter described the following classifiers: Naive Bayes, SVM, MLP, IBL, Bag, AdaBM1, RoFo and C4.5.

Chapter 4 described how we created the datasets used in the evaluation. There were two different datasets created: first, the Bangor Source File Corpus called (BSFC) and second, Bangor Encryption Classification Corpus called (BECC). Different methods were used to analyse the datasets to find out whether the datasets were random or not. These were: Frequency test, Chi-square test and a compression test (PPM). The results from running these tests show that the encrypted data is random in nature and therefore difficult to classify.

Chapter 5 includes three main experiments. The first used different numbers of keys with dissimilar text files (512KB) in ECB and CBC modes. The second experiment worked on changing the number of keys sizes and the encryption key in block cipher algorithms. The third experiment worked on classifying different instances with different numbers of keys sizes. In this experiment, eight classifiers were used: Naive Bayes, SVM, MPL, IBL, Bag, AdaBM1, RoFo, and C4.5 for accuracy. All experiments were performed using the WEKA machine learning platform. The aim was to find the best classification algorithm for four different block ciphers (DES, IDEA, AES, and RC2). Different keys were also used for each algorithm.

The results illustrated that the RoFo classifier has the best performance on identifying the encryption method for the ciphered data in the BECC while the worst performance was IBL. Moreover, the performance of the classifiers differed significantly when identification of four classes (i.e four different encryption algorithms) was considered. It was observed that the three versions of AES (128, 192 and 256 bits) are not separately distinguishable. It was also noted that RC2 (128 bits) does not match the other versions of the same encoding RC2 (42 bits) and RC2 (84 bits).

As expected, by increasing the number of encryption keys, the classification accuracy will reduce. It was also observed that increasing the number of files used also improved the accuracy. Further, the results show that it was possible to achieve accuracy above 40% with some classifiers when each file was encrypted with a different key.

A very interesting point was that the ECB mode had higher accuracy than the CBC mode which was not expected. Unfortunately, in terms of using both RoFo and MLP classifiers with 16-bit codes, no result were obtainable since there were too many features to be processed.

Chapter 6 included an analysis of stream cipher algorithms. In all the algorithms that were investigated, the difference in accuracy was found to be slight when 8-bit and 16-bit codes were compared. Moreover, the five algorithms used with 16-bit codes were not distinguishable, unlike with 8-bit codes where there was a clear pattern.

7.2 Review of Hypothesis

The hypothesis is that Pattern Recognition classification techniques can be used effectively to help identify the encryption method used to encrypt the data. The results showed that Pattern Recognition techniques are useful tools for cryptanalysis as a means of identifying the type of encryption algorithm used to encrypt the data. The hypothesis was proved for block cipher algorithms but was not proved for stream cipher algorithms. For stream cipher algorithms the system had difficulty in distinguishing between different types of stream cipher algorithms but with block cipher algorithms many of the classifier were able to distinguish between algorithms.

7.3 Review of Aim and Objectives

The main objective of this dissertation was to test out a novel application of classification (i.e. identification) to classify encryption output. In this study, eight classifiers were used: Naive Bayes, SVM, MLP, IBL, Bag, AdaBM1, RoFo and C4.5. The study were focused on classifying encryption output for different block and stream cipher algorithms.

Following are the list of the objectives:

- The creation of a dataset of encrypted files that can be used for evaluation of classification accuracy.
- Analysis of the encrypted text files.

The first objective has been achieved in Chapter 4, and the second objective has been achieved in Chapters 5 and 6.

7.4 Future Work

The limitations and future work of the work presented in this dissertation are summarized as follows:

1. The case study presented in this thesis only evaluates encryption using a selection of block and stream ciphers. It cannot be used for all types of block and stream ciphers because of the specific focus of this research on a finite selection of algorithms and further investigation is needed for other algorithms.
2. Classification of Asymmetric algorithms could be investigated using a similar study. Pattern Recognition techniques can also be applied to identify the encryption method for these algorithms.
3. There are three further types of encryption modes that could be investigated to enhance the accuracy and the security with block cipher algorithms: CFB, OFB and CTR modes.
4. The CBC mode did not give the expected results due to the key chaining nature of the algorithm. Each block is dependent on all plain-text block procedures up to the preceding point, therefore that each message is unique. This needs to be investigated further.
5. The Fuzzy method could be used to further enhance the accuracy of block and stream cipher algorithms and the effected on the classification accuracy needs to be investigated.
6. Text categorisation techniques could be used to analysis the datasets and compare with our results [278].

Appendix A

Matlab Code to Generate the WEKA Files and Histograms

A.1 Generate WEKA File Code

Source code for generating the WEKA file.

```
t = dir(pa);

Data8 = [];
Labels = [];

k = 1;
for i = 3:numel(t)
    ENCClasses{k} = t(i).name;
    tinfolder = dir([pa '/' t(i).name]);

for j = 3:numel(tinfolder)
s = [pa '/' t(i).name '/' num2str(j-2) '_encrypted.bin'];
f = fopen(s);
A = fread(f);
fclose(f);
h = extract_binary_histogram_new(A,1);
Data8 = [Data8;h];
Labels = [Labels;k];

fprintf('Now working on %s, file %s\n',ENCClasses{k},s)

end
k = k + 1;
end
save Data8Matlab_ECB Data8 Labels ENCClasses


%% Prepare a version fo Weka

if exist('Data8Weka_ECB.arff')
    delete('Data8Weka_ECB.arff')
end
clc
diary Data8Weka_ECB.arff
```

```
fprintf('@relation SuhailaENC\n\n')
for i = 1:size(Data8,2)
    fprintf('@attribute v%d real\n',i)
end
fprintf('@attribute class {1,2,3,4,5,6,7,8}\n\n@data\n\n')

for i = 1:size(Data8,1)
    for j = 1:size(Data8,2)
        fprintf('%9i',Data8(i,j))
    end
    fprintf(' %i\n',Labels(i))
end
diary off
```

A.2 Generate Histogram

Source code for generating the histograms.

```
function h = extract_binary_histogram(A,Mode)
if Mode == 1
    Granularity = 8;
    wb = waitbar(0,'Please wait...');
    HistogramElements = 2^Granularity;
    for i = 1:HistogramElements
        h(i) = sum(A == i);
        waitbar(i/HistogramElements,wb)
    end
    close(wb)
else
    Granularity = 16;
    A = reshape(A,2,length(A)/2)';
    Index = A(:,1)*256 + A(:,2) + 1;

    wb = waitbar(0,'Please wait...');
    HistogramElements = 2^Granularity;
    for i = 1:HistogramElements
        h(i) = sum(Index == i);
        waitbar(i/HistogramElements,wb)
    end
    close(wb)

end
```

Appendix B

Generate Histograms for Each Block Cipher Algorithms

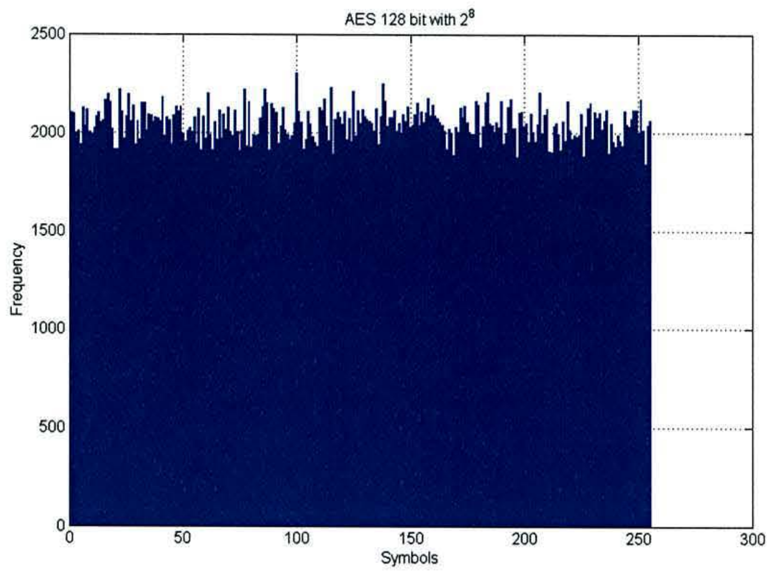
The histogram method is a statistical statement that demonstrates the frequency of values within ranges or steps of values that fall between a certain minimum and maximum. The aim of this method is to capture the statistical properties of the cipher-text. It illustrates the variations in the frequency of occurrence of symbols, which means it can be employed as the classification decisive factor. And also, to find the datasets are random or not.

A pattern of varying encryption methods can be perceived in the histograms of the cipher-texts. A histogram is a technique extensively applied in different application frameworks which are characteristic of query optimization statistical and temporal databases, OLAP applications, data streams and so on. Histograms are well suited to the aim of this study, particularly on occasions of identifying the encryption method from Ciphertext-Only Attacks [279] [280].

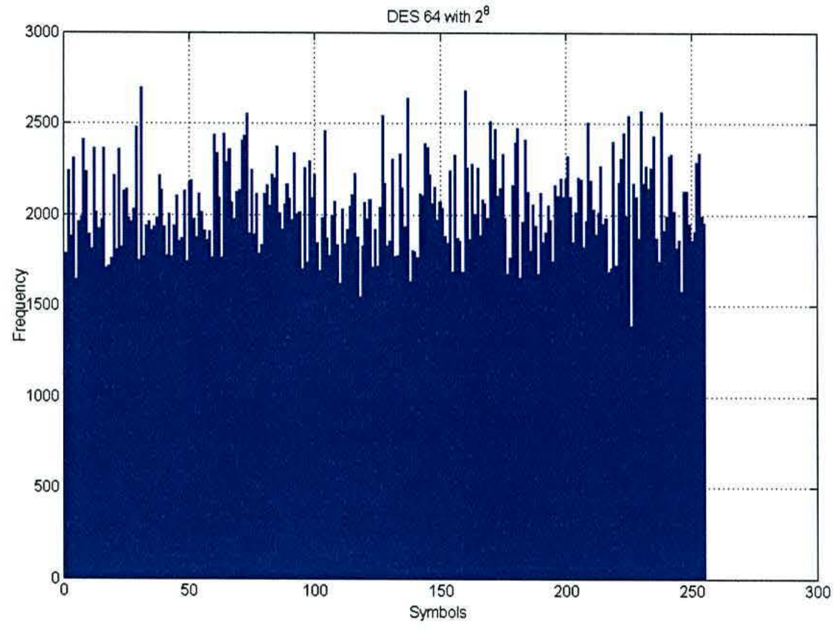
According to the results, these histograms was found to be very informative as it demonstrates that the entirety of the pixels have a pattern for 8-bit codes, but with 16-bit codes the pattern is not clear because the features are too large. In these histograms it was noted that the blue curves show how the raw data will be converted to the final image, and a pattern can be perceived in the histograms of the cipher-texts of varying encryption methods.

Figure B.1 to Figure B.2 show images of the histograms for all algorithms for 8-bit codes with AES, DES, IDEA and RC2 using 128-bits key size in ECB mode. Figure B.3 to Figure B.4 shows the CBC mode histogram using the same algorithms with same key sizes. In general, all histograms for he 8-bit codes shows that the distribution indicated that the data is random in nature.

Next, Figure B.5 shows the histogram of all algorithms with ECB and CBC modes using a 16-bit codes. Figure B.7 and Figure B.8 show that there is no pattern in CBC mode using a 16-bit codes because the features are too large when it is in



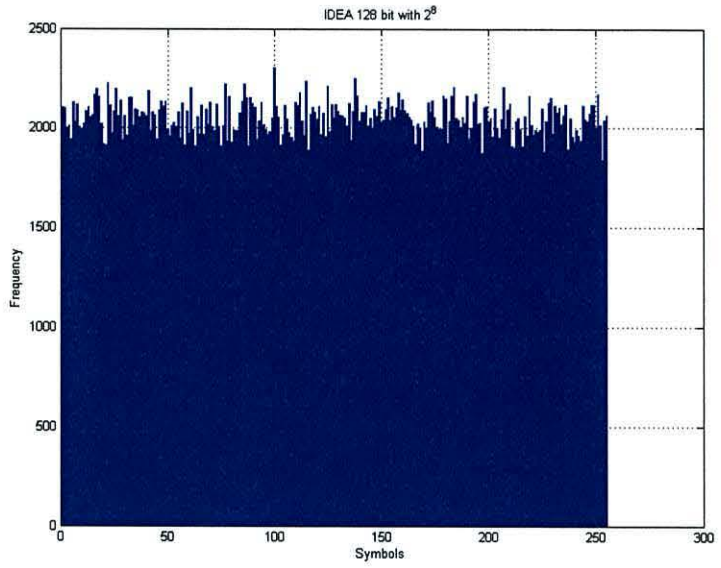
(a) AES-128bits



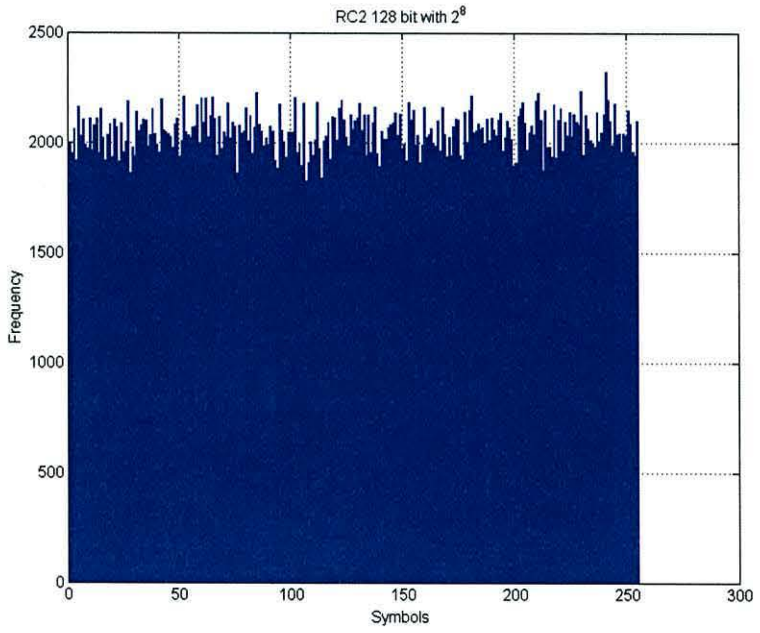
(b) DES-64bits

Fig. B.1: Histograms for AES and DES algorithms with ECB mode using 8-bits.

CBC mode.

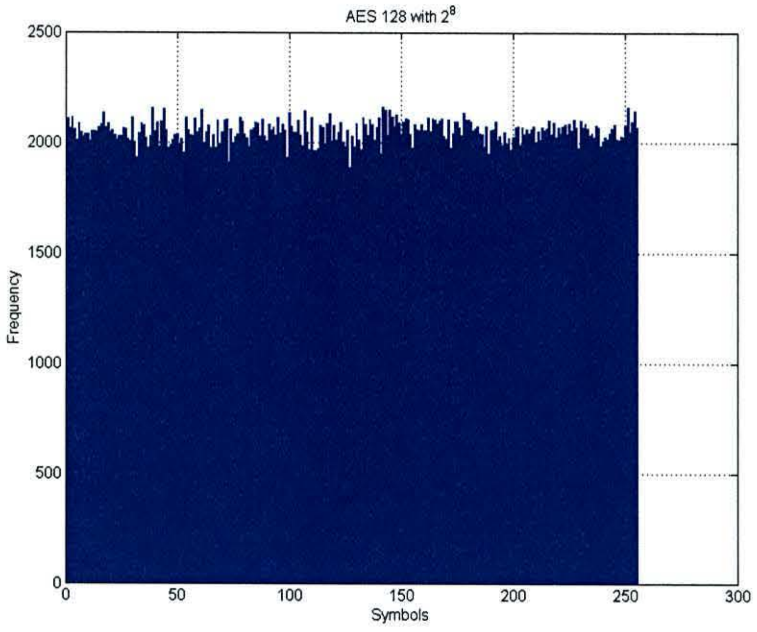


(a) IDEA-128bits

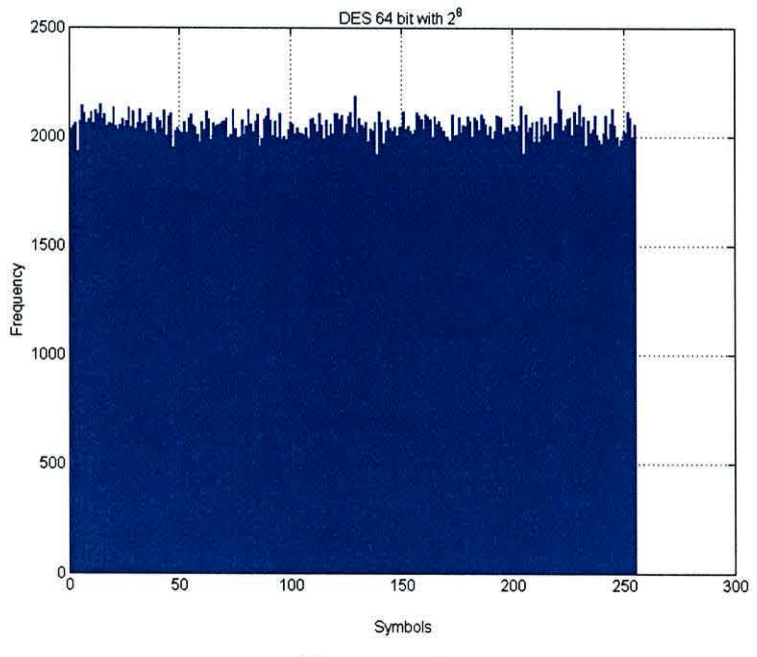


(b) RC2-128bits

Fig. B.2: Histograms for IDEA and RC2 algorithms with ECB mode using 8-bit codes.

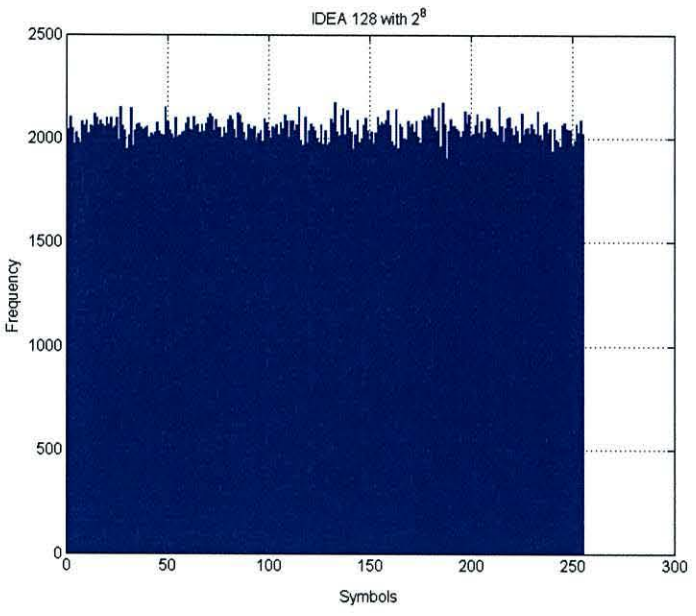


(a) AES-128bits

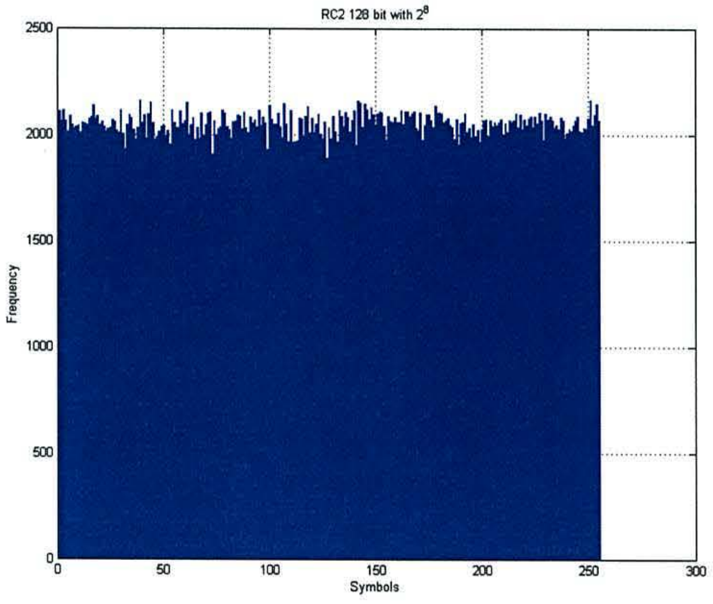


(b) DES-64bits

Fig. B.3: Histograms for AES and DES algorithms with CBC mode using 8-bit codes.

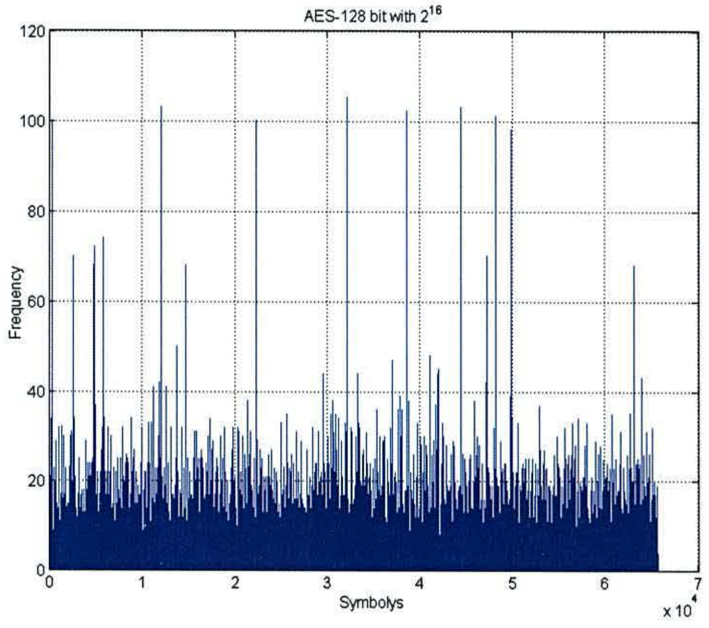


(a) IDEA-128bits

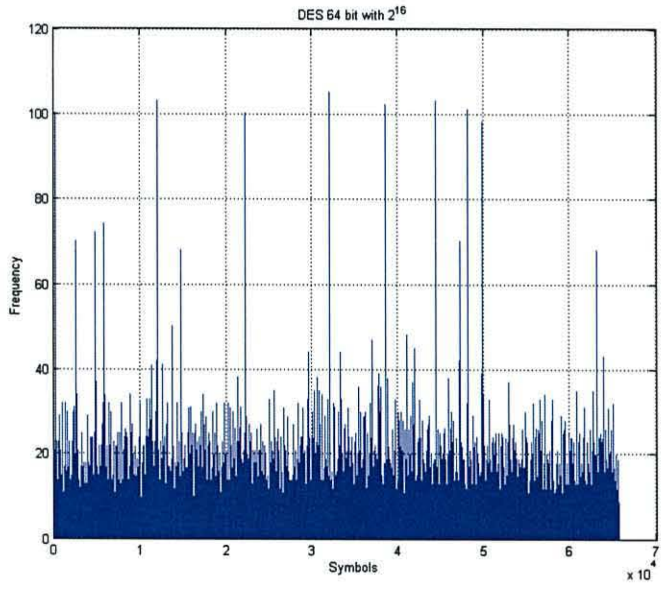


(b) RC2-128bits

Fig. B.4: Histograms for IDEA and RC2 algorithms with CBC mode using 8-bit codes.

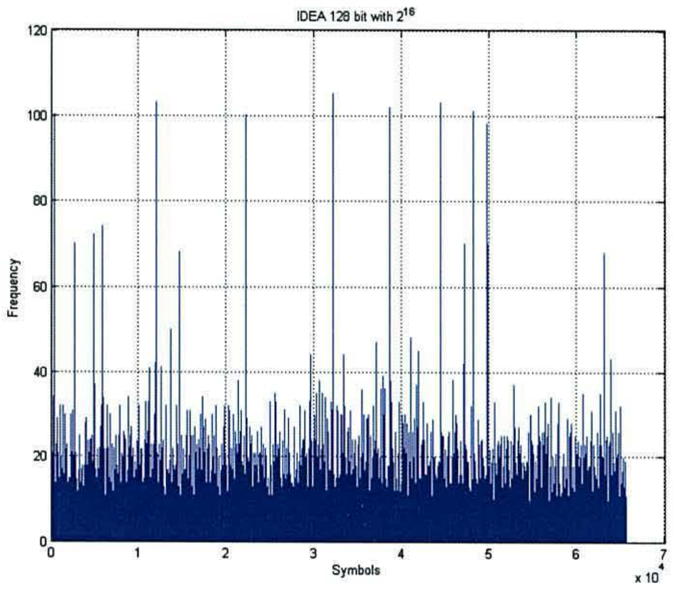


(a) AES-128bits

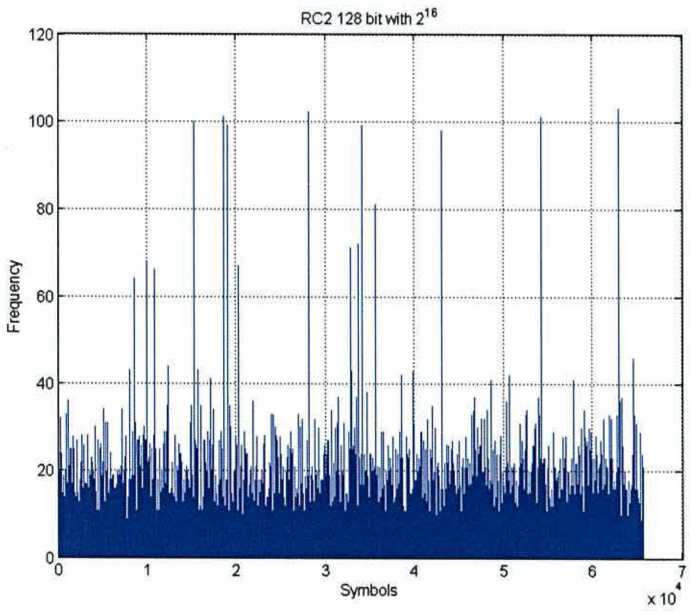


(b) DES-64bits

Fig. B.5: Histograms for AES and DES algorithms with ECB mode using 16-bit codes.

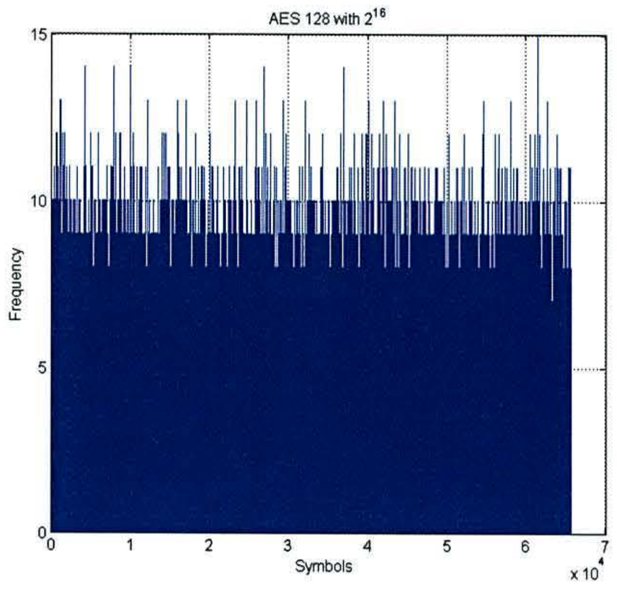


(a) IDEA-128bits

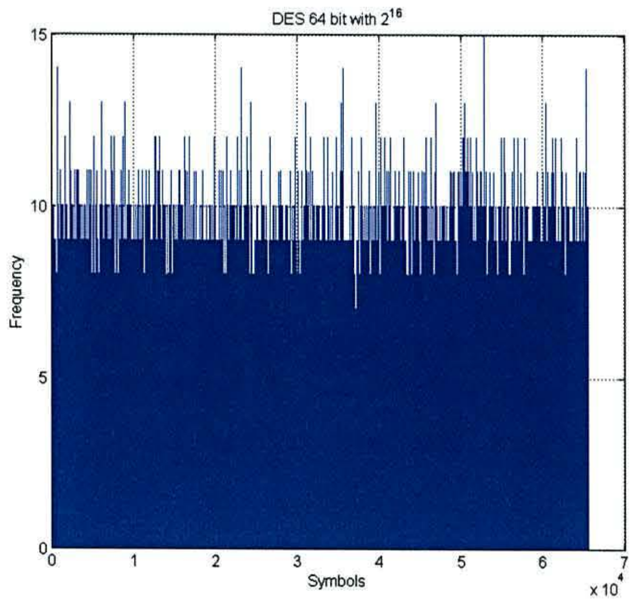


(b) RC2-128bits

Fig. B.6: Histograms for IDEA and RC2 algorithms with ECB mode using 16-bit codes.

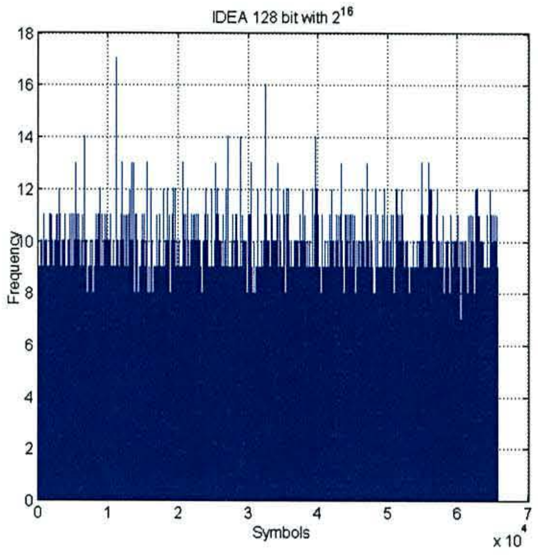


(a) AES-128bits

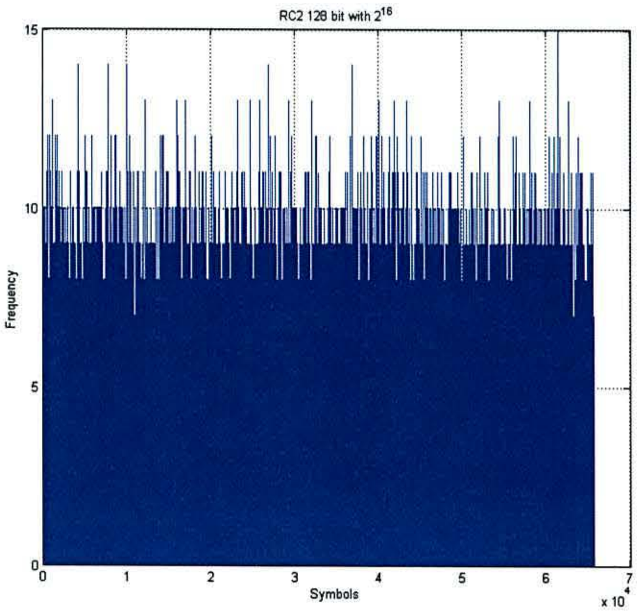


(b) DES-64bits

Fig. B.7: Histograms for AES and DES algorithms with CBC mode using 16-bit codes.



(a) IDEA-128bits



(b) RC2-128bits

Fig. B.8: Histograms for IDEA and RC2 algorithms with CBC mode using 16-bit codes.

References

[1] Z. Linxian, "Reseach of Image Encryption Algorithm Based on S-DES," *2012 International Conference on Computer Science and Electronics Engineering*, vol. 1, pp. 184–187, 2012.

[2] Z. B. LLC, "Data Encryption," 2012.

[3] L. Lan, "The AES Encryption And Decryption Realization Based On FPGA," *Computational Intelligence and Security (CIS), 2011 Seventh International Conference*, pp. 603–607, 2011.

[4] S. Rizvi, S. Hussain, and N. Wadhwa, "PerformanceAnalysis of AES and TwoFish Encryption Schemes," *International Conference, Communication Systems and Network Technologies (CSNT)*, pp. 76 – 79, 2011.

[5] T. Nie, C. Song, and X. Zhi, "Performance Evaluation of DES and Blowfish Algorithms," *Biomedical Engineering and Computer Science (ICBECS), 2010 International Conference*, pp. 1 – 4, 2010.

[6] A. Mohamed, G. Zaibi, and A. Kachouri, " Implementation of RC5 and RC6 Block Ciphers on Digital Images," *Systems, Signals and Devices (SSD), 2011 8th International Multi-Conference*, pp. 1–6, 2011.

[7] K. G. N and D. V. Ramaswamy, "Encryption Quality Analysis and Security Evaluation of CAST-128Algorithm and its Modified Version using Digital Images," *International Journal of Network Security & Its Applications (IJNSA)*, vol. 1, pp. 20–33, 2009.

[8] "Pattern Recognition." Accessed: 0712/2012.

[9] Y. W. Lim, "Efficient 8-cycle DES Implementation," in *Second IEEE Asia Pacific Conference on ASICs, 2000. AP-ASIC 2000.pp 175 - 178*, 2000.

[10] NBS FIPS PUB, "Data Encryption Standard DES," *FIPS PUB 46-3,U.S. DEPARTMENT OF COMMERCE/National Institute of Standards and Technology*, 1999.

[11] J. Rejeb and V. Ramaswamy, "Efficient Rijndael Implementation for High-Speed Optical Networks," *10th International Conference Telecommunications, ICT 2003.*, vol. 1, pp. 641 – 645, 2003.

[12] S. F. Mare, M. Vladutiu, and L. Prodan, "Secret Data Communication System using Steganography, AES and RSA," *IEEE 17th International Symposium for Design and Technology in Electronic Packaging (SIITME)*, pp. 339 – 344, 20-23 Oct. 2011.

[13] J.-S. Coron, "What Is Cryptography?," *Security & Privacy, IEEE*, vol. 4, pp. 70 –73, 2006.

[14] A. M. Eskicioglu and L. Litwin, "Cryptography," vol. 20, pp. 36 – 38, 2001.

[15] H. Delfs and H. Knebl, *Introduction to Cryptography Principles and Applications* . Berlin Heidelberg New York: Springer-Verlag 2007, 2007.

[16] M. Umaparvathi and D. Varughese, "Evaluation of Symmetric Encryption Algorithms for MANETs," *2010 IEEE International Conference. Computational Intelligence and Computing Research (ICCIC),*, pp. 1 – 3, 28-29 Dec. 2010.

[17] Z. Yun-peng, L. Xia, and W. Qiang, "Asymmetric Cryptography Algorithm with Chinese Remainder Theorem," *Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference*, pp. 450 – 454, 27-29 May 2011.

[18] M. S. Rhee and B. Lee, *Information security and cryptology - ICISC 2006:.* 9th international conference, Busan, Korea, November 30 - December 1, 2006 ; proceedings (Google eBook), 2007.

[19] D. Lee, "Hash Function Vulnerability Index and Hash Chain Attacks ," *Secure Network Protocols, 2007. NPSec 2007. 3rd IEEE Workshop*, pp. 1 – 6, 2007.

[20] Q. Yu, C. N. Zhang, and X. Huang, "An RC4-Based Hash Function for Ultra-Low Power Devices," *Computer Engineering and Technology (ICCET), 2010 2nd International Conference*, vol. 1, pp. 323 –328, 2010.

[21] S. Singh, "The Code Book:The Secret History of Codes and Code-breaking (Google eBook)," 2010.

[22] S. Singh, *The Cracking Code Book: How to Make It, Break It, Hack It, Crack It.* HarperCollinsChildren'sBooks, 2009.

[23] B. A. Forouzan, *Cryptography and Network Security.* McGraw-Hill Higher Education, 2008.

[24] A. Mousa and A. Hamad, "Evaluation of the RC4 Algorithm for Data Encryption," *International Journal of Computer Science & Applications*, vol. 2, pp. 44–56, 2006.

[25] M. Peyraviar and D. Coppersmith, "A Structured Symmetric-Key Block Cipher," *Computers & Security*, vol. 18, pp. 134–1 47, 1999.

[26] G.-H. Kim, J.-N. Kim, and G.-Y. Cho, "Symmetry Structured SPN Block Cipher Algorithm," *11th International Conference, Advanced Communication Technology, 2009. ICACT 2009.*, vol. 03, pp. 1777 – 1780, 2009.

[27] H. M. Heys, "Information Leakage of Feistel Ciphers," *Information Theory, IEEE Transactions*, vol. 47 , Issue:1, pp. 23 – 35, 2001.

[28] H. Mohamed, A. Kader, and M. M. Hadhoud, "Performance Evaluation of Symmetric Encryption Algorithms," *Journal of Computer Science*, vol. 8, pp. 280–286, 2008.

[29] J. M. Granado-Criado, M. A. Vega-rodríguez, J. M. Sánchez-pérez, and J. A. Gómez-pulido, "A Dynamically and Partially Reconfigurable Implementation of the IDEA Algorithm Using FPGAs and Handel-C," *Journal of Universal Computer Science*, vol. 13, pp. 407–418, 2007.

[30] T. Teerakanok and S. Kamolphiwong, "Accelerating Asymmetric-key Cryptography using Parallel-key Cryptographic Algorithm PCA," in *6th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, 2009. ECTI-CON 2009.*, pp. 812 – 815, 6-9 May 2009.

[31] V. Pachghare, *Cryptography and Information Security*. 30/01/2010, 2010.

[32] J. LANO, "Cryptanalysis and Design of Synchronous Stream Ciphers," *PhD thesis*, 2006.

[33] S.-J. Han, H.-S. Oh, and J. Park., "The improved data encryption standard DES algorithm," *IEEE 4th International Symposium,Spread Spectrum Techniques and Applications Proceedings, 1996.*, vol. 3, pp. 1310 – 1314, 1996.

[34] J. O. Grabbe, "TheDESAlgorithm Illustrated," p. http://orlingrabbe.com/des.htm.

[35] K.-Y. Yuan, J. Chen, and G.-P. Liu, "Design and Implementation of Data Encryption for Networked Control Systems," *Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics San Antonio, TX, USA .*, pp. 2105 – 2109, 2009.

[36] W. Wu, J. Jin, and J. Cheng, "The Research and Design of ATM PIN Pad Based on Triple DES ," *Information and Automation (ICIA), 2011 IEEE International Conference*, pp. 443 – 447, 6-8 June 2011.

[37] M. Matin, M. Hossain, M. Islam, and M. Islam, "Performance Evaluation of Symmetric Encryption Algorithm inMANET and WLAN," 2009.

[38] D. Feldmeier, "AHigh-Speed Software DES Implementation," *Computer Communication I Research Group, Bellcore, Morristown, NJ 07962*, 1989.

[39] B. S. a ndHaris Tauqeer and M. S. Ilyas, "Hardware Implementation of DES Encryption Cracker," *Engineering Sciences and Technology, SCONEST 2005. Student Conference*, pp. 1 – 4, 27-27 Aug. 2005.

[40] H. Eberle and H. Eberle, "AHigh-speed DES Implementation for Network Applications," 1992.

[41] Federal Information Processing Standards Publication 46-2, "DES Encryption Standard DES," *FIPS PUB 46-3,U.S. DEPARTMENT OF COMMERCE/National Institute of Standards and Technology*, pp. http://www.itl.nist.gov/fipspubs/fip46–2.htm, 1993.

[42] J. Qing-feng and Q. Shui-sheng, "A New Image Encryption Scheme Based on DES Algorithm and Chua's Circuit," *Imaging Systems and Techniques, 2009. IST '09. IEEE International Workshop*, pp. 168–172, 2009.

[43] L. Zhi, "Reseach of Image Encryption Algorithm Based on S-DES," *Computer Science and Electronics Engineering (ICCSEE), 2012 International Conference*, vol. 1, pp. 184–187, 2012.

[44] D. A. Osvik, *Efficient Implementation of the Data Encryption Standard*. 2003.

[45] S. J.Shepherd, "A high Speed Software Implementation of the Data Encryption Standard," *Cryptography and Computer Security, Electrical Engineering Department, University of Bradford, Bradford, UK*, vol. 14, pp. 349–357, 1995.

[46] M. Fischer, "How to Implement the Data Encryption Standard," 1995.

[47] J. Choa, S. Soekamtoputrab, K. Choib, and J. Moona, "Power Dissipation and area Comparison of 512-bit and 1024-bit key AES," *Computers & Mathematics with Applications*, pp. 1–6, 2012.

[48] Z. Hu, "Progress in the Advanced Encryption Standard," *International Conference on Intelligence Science and Information Engineering*, pp. 345 – 348, 2001.

[49] N. Kosaraju, M. Varanasi, and S. Mohanty, "A High-Performance VLSI architecture for Advanced Encryption Standard AES Algorithm," *19th International Conference, VLSI Design, 2006. Held jointly with 5th International Conference on Embedded Systems and Design.,*, p. 4 pp, 2006.

[50] A. Refik Sever and M. A. Neslin smailoğlu, Yusuf C. Tekmen, "A High Speed ASIC Implementaion of the Rijndael Algorithm," *Circuits and Systems, 2004. ISCAS '04. Proceedings of the 2004 International Symposium*, vol. 2, pp. II – 541–4, 2004.

[51] X. liang Wang, F. hai Xiao, and D. yong Wang, "Application of AES Algorithm in Digital Cnema Projection System based on DaVinci technology," *Information Networking and Automation (ICINA), 2010 International Conference*, pp. V2–24 – V2–27, 2010.

[52] S. Shivkumar and D. G. Umamaheswari, "Performance Comparison of Advanced Encryption Standard (AES) and AES key dependent S-box - Simulation using MATLAB," *Process Automation, Control and Computing (PACC), 2011 International Conference*, pp. 1 – 6, 20-22 July 2011.

[53] X. Zhang and P. K.K, "Implementation Approaches for the Advanced Encryption Standard Algorithm," *Circuits and Systems Magazine, IEEE*, vol. 4, pp. 24 – 46, 2002.

[54] C. Jenkins, M. Schulte, and J. Glossner, "Instruction set Extensions for Triple DES Processing on a Multi-threaded Software-defined Radio Platform ," *Signals, Systems and Computers (ASILOMAR), 2010 Conference Record of the Forty Fourth Asilomar Conference*, pp. 1387 – 1391, 2010.

[55] W. C. Barker, "Informtion Security," *NIST Special Publication 800-67 Version 1.1, Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher*, 19 May 2008.

[56] A. T. Software, "This is a test entry of type @ONLINE," June 1997.

[57] W. Stallings, *Cryptography and Network Security*. Prentice Hall, 2005.

[58] X. Bi, L. Wu, and G. Bai, "Design and FPGA Implementation of 3DES against Power Analysis Attacks for Ie Bankcard," *ASIC, 2009. ASICON '09. IEEE 8th International Conference*, pp. 159–162, 2009.

[59] A. Nadeem and D. M. Y. Javed, "A Performance Comparison of Data Encryption Algorithms," *Information and Communication Technologies, 2005. ICICT 2005. First International Conference*, pp. 84 –89, 2005.

[60] P. Ghosal, M. Biswas, and M. Biswas, "A Compact FPGA Implementation of Triple-DES Encryption System with IP Core Generation and On-Chip Verification," *Proceedings of the 2010 International Conference on Industrial Engineering and Operations Management Dhaka, Bangladesh*, 2010.

[61] W. Guo, Z. Li, Y. Chen, and X. Zhao, "Security Design for Instant Messaging System Based on RSA and Triple DES," *Image Analysis and Signal Processing, 2009. IASP 2009. International Conference*, pp. 415 – 418, 2009.

[62] S. Yang, H. Piao, L. Zhang, and X. Zheng, "An Improved IDEA Algorithm Based on USB Security Key ," *Natural Computation, 2007. ICNC 2007. Third International Conference*, vol. 3, pp. 184–188, 2007.

[63] J. J. G. Savard, "IDEA (International Data Encryption Algorithm)," 2012.

[64] A. Biryukov, J. N. Jr, B. Preneel, and J. Vandewalle, "New Weak-Key Classes of IDEA," in *4th Internation Conference, Appeared in Information and Communications Security, ICICS 2002, Lecuter Notes in Computer Secience 2513.Springer-Verlag.*, pp. 315–326, 2002.

[65] R. Zimmermann, A. Curiger, H. Bonnenberg, H. Kaeslin, N. Felber, and W. Fichtner, "A 177 Mb/s VLSI Implementation of the International Data Encryption Algorithm," *IEEE JOURNAL OF SOLID-STATE CIRCUITS.*, vol. 3, pp. 303–307, 1994.

[66] P. Junod, "Statistical Cryptanalysis of Block Ciphers," 2004.

[67] P.-J. Kang, S.-K. Lee, and H.-Y. Kim, "Study on the Design of MDS-M2 TwofIsh Cryptographic Algorithm adapted to Wireless Communication," *The 8th International Conference. Advanced Communication Technology, 2006. ICACT 2006.*, vol. 1, pp. 4 pp. – 695, 2006.

[68] B. Schneier, J. Kelsey, D. Whiting, D. Wagnery, C. Hall, and N. Ferguson, "On the Twofish Key Schedule," pp. 1–17, 1998.

[69] B. Schneier, J. Kelsey, D. Whitingz, D. Wagner, C. Hall, and N. Ferguson, "Twofish: A 128-Bit Block Cipher," 1998.

[70] J. Gargiulo, "Global Information Assurance Certification Paper,This paper is Taken From the GIAC Ddirectory of Certified Professionals. Reposting is not Permited without Express Written Permission,SANS Institute Author Rebains Full Rights," p. SANS, 2002.

[71] B. Gatliff, "Encrypting Data with the Blowfish algorithm," *http://www.embedded.com/showArticle.jhtml?articleID=12800442,* Jul 15 2003 (11:00 AM).

[72] M. Peyraviana and D. Coppersmithb, "A Structured Symmetric-Key Block Cipher," *Computers & Security,Elsevier Science Ltd*, vol. 18, pp. 134–147, 1999.

[73] N. Palaniswamy, M. Dipesh Dugar, N. Dinesh Kumar Jain, and G. Raaja Sarabhoje, "Enhanced Blowfish Algorithm Using Bitmap Image Pixel Plotting for Security Improvisation," *2nd International Conference, Education Technology and Computer (ICETC),*, pp. V1–533 – V1–538, 2010.

[74] T. Nie and T. Zhang, "A Study of DES and Blowfish Encryption Algorithm," *IEEE Region 10 Conference of TENCON 2009 -*, pp. 1 – 4, 2009.

[75] C. T. R. Hager, S. F. Midkiff, J.-M. Park, and T. L. Martin, "Performance and Energy Efficiency of Block Ciphers in Personal Digital Assistants," *Third IEEE International Conference , Pervasive Computing and Communications,*, pp. 127–136, 2005.

[76] A. J. Marcella and D. Menendez, *Cyber Forensics: a field manual for collecting, examining, and preserving evidence of computer crimes.* CRC Press, 2007.

[77] M. Y. Rhee, *Internet Security Cryptographic Principles, Algorithms and Protocols.* 2003.

[78] O. Elkeelany and A. Olabisi, "Performance Comparisons Design and Implementation of RC5 Symmetric Encryption Core Using Reconfigurable Hardware," *JOURNAL OF COMPUTERS*, vol. 3, pp. 48–55, 2008.

[79] M. Hasan and H. Al-Shalabi, "Modified Cryptanalysis of RC5," *The Internatioanl Arab Jornal of Information Technology*, vol. 4, 2005.

[80] B. Preneel, "Fast Software Encryption," *Second International Workshop, Leuven, Belgium(GoogleeBook)*, vol. 2, 1994.

[81] H. E. din H. Ahmed, H. M. Kalash, and O. S. F. Allah, "Implementation of RC5 Block Cipher Algorithm for Image Cryptosystems," *International Journal of Information Technology*, vol. 3,no.4, pp. 245–250, 2007.

[82] H. E. din H. Ahmed, H. M. Kalash, and O. S. F. Allah, "Encryption Efficiency Analysis and Security Evaluation of RC6 Block Cipher for Digital Images," *International Journal of Computer, Information, and Systems Science, and Engineering*, pp. 33–39, 2007.

[83] W. Stallings, *Cryptography and Network Security.* Pearson Education, 2003.

[84] C. M. Adams, "Constructing Symmetric Ciphers Using the CAST Design Procedure," *Designs, Codes and Cryptography*, vol. 12, pp. 283–316, 1997.

[85] G. Jakimoski and L. Kocarev, "Differential and Linear Probabilities of a Block-Encryption Cipher ," *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions*, vol. 50 Issue: 1, pp. 121 – 123, Jan. 2003.

[86] A.-K. A. Tamimi, "Performance Analysis of Data Encryption Algorithms."

[87] G. Wang and S. Wang, "Improved Differential Cryptanalysis of Serpent," *Computational Intelligence and Security (CIS), 2010 International Conference*, pp. 367 – 371, 11-14 Dec. 2010.

[88] F.-X. B.Collard and J.-J.Quisuater, "Improved and Multiple Linear Cryptanalysis of Reduced Round Serpent," *Information Security and Cryptology , Thired SKLOIS Conferenec, Inscrpt 2007 Xining, China.,Springer*, pp. 51–65, 2007.

[89] B. Najafi, B. Sadeghian, M. Saheb Zamani, and A. Valizadeh, "High Speed Implementation of Serpent Algorithm," *Microelectronics, 2004. ICM 2004 Proceedings. The 16th International Conference*, pp. 718 – 721, 2004.

[90] M. Çakiroğlu, "Software implementation and performance comparison of popular block ciphers on 8-bit low-cost microcontroller," *International Journal of the Physical Sciences*, vol. 5(9), pp. 1338–1343, 2010.

[91] ISO/IEC, "Information technology -Security techniques -Encryption algorithms -Part 2: Asymmetric ciphers ," *Webstore International Electrotechnical Commission*, 2006.

[92] T. Sugawara, N. Homma, T. Aoki, and A. Satoh, "A High-Performance ASIC Implementation of the 64-bit Block Cipher CAST-128," *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium*, pp. 1859–1862, 2007.

[93] K. Boey, Y. Lu, M. O'Neill, and R. Woods, "Differential Power Analysis of CAST-128," *IEEE Annual Symposium on VLSI*, pp. 143 – 148, 2010.

[94] J. Viega, M. Messier, and P. Chandra, *Network Security with OpenSSL*. OŔeilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, 2002.

[95] T. Sugawara, N. Homma, T. Aoki, and A. Satoh, "A High-Performance ASIC Implementation of the 64-bit Block Cipher CAST-128," *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium*, pp. 1859 – 1862, 2007.

[96] H. Adams, C.and Heys, S. Tavares, and M. Wiener, "An Analysis of the CAST-256 Cipher," *Electrical and Computer Engineering, 1999 IEEE Canadian*, vol. 1, pp. 361 – 366, 1999.

[97] C. Adams, "The CAST-256 Encryption Algorithm," June 1999.

[98] M. Riaz and H. Heys, "The FPGA Implementation of the RC6 and CAST-256 Encryption Algorithms," *Electrical and Computer Engineering, 1999 IEEE Canadian Conference*, vol. 1, pp. 367 – 372, 1999.

[99] A. Pestunov, "Differential Cryptanalysis of 24-Round CAST-256," *IEEE Region 8 International Conference. Computational Technologies in Electrical and Electronics Engineering, 2008. SIBIRCON 2008.*, pp. 46 – 49, 21-25 July 2008.

[100] O. Verma, R. Agarwal, D. Dafouti, and S. Tyagi, "Peformance Analysis of Data Encryption Algorithms," *Electronics Computer Technology (ICECT), 2011 3rd International Conference*, vol. 5, pp. 399 – 403, 2011.

[101] N. Couture and K. B. Kent, "The Effectiveness of Brute Force Attacks on RC4," *Second Annual Conference. Communication Networks and Services Research, 2004. Proceedings.*, pp. 333 – 336, 19-21 May 2004.

[102] B. Crainicu and B. LászlóIantovics, "Cryptanalysis of KSAm-like Algorithms," *First International Conference on Complexity and Intelligence of the Artificial and Natural Complex Systems. Medical Applications of the Complex Systems. Biomedical Computing*, pp. 130 – 148, 2008.

[103] P. Prasithsangaree and P. Krishnamurthy, "Analysis of Energy Consumption of RC4 and AES Algorithms in Wireless LANs," *'03. IEEE ,Global Telecommunications Conference, 2003. GLOBECOM*, vol. 3, pp. 1445 – 1449, 2003.

[104] Y. Yao, J. Chong, and W. Xingwei, "Enhancing RC4 algorithm for WLAN WEP Protocol," *Control and Decision Conference (CCDC), 2010 Chinese*, pp. 3623 – 3627, 2010.

[105] A. M. Riad, A. R. Shehata, E. K. Hamdy, M. H. Abou-Alsouad, and T. R. Ibrahim, "Evaluation of the RC4 Algorithms a Solution for Converged Networ," *Journal of ELECTRICAL ENGINEERING*, vol. 3, pp. 155–160, 2009.

[106] J. Xie and X. Pan, "An Improved RC4 Stream Cipher," *International Conforence on Computer Application and System Modeling (ICCASM 2010)*, vol. 7, pp. 156 –159, 2010.

[107] S. H. M. Kwok and E. Y. Lam, "Effective Uses of FPGAs for Brute-Force Attack on RC4 Ciphers," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions*, vol. 16 , Issue:8, pp. 1096 – 1100, 2008.

[108] H. Zhang and X. Wang, "Cryptanalysis of Stream Cipher Grain Family." Cryptology ePrint Archive, Report 2009/109, 2009. `http://eprint.iacr.org/`.

[109] M. Afzal and A. Masood, "Algebraic Cryptanalysis of A NLFSR Based Stream Cipher," *ICTTA 2008. 3rd International Conference, Information and Communication Technologies: From Theory to Applications*, pp. 1 – 6, 2008.

[110] S. S. Mansouri and E. Dubrova, "Improved Hardware Implementation of the Grain Stream Cipher," *13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools*, pp. 433 – 440, 2010.

[111] M. A. M. Hell, T. Johansson, and W. Meier, "A New Version of Grain-128 with Authentication," Presented at SKEW 2011, available via skew2011.mat.dtu.dk/ proceedings/.

[112] J. Yan and H. Heys, "Hardware Implementation of the Salsa20 and Phelix Stream Ciphers," *Electrical and Computer Engineering, CCECE 2007. Canadian Conference*, pp. 1125 – 1128, 2007.

[113] L. Henzen, F. Carbognani, N. Felber, and W. Fichtner, "VLSI Hardware Evaluation of the Stream Ciphers Salsa20 and ChaCha, and the Compression Function Rumba," *2008 International Conference on Signals, Circuits and Systems*, pp. 1–5, 2008.

[114] G. Meiser, T. Eisenbarth, K. Lemke-Rust, and C. Paar, "Efficient Implementation of eSTREAM Ciphers on 8-bit AVR Microcontrollers," *International Symposium, Industrial Embedded Systems, 2008. SIES 2008.*, pp. 58 – 66, 2008.

[115] L. Henzen, F. Carbognani, N. Felber, and W. Fichtner, "VLSI Hardware Evaluation of the Stream Ciphers Salsa20 and ChaCha, and the Compression Function Rumba," *2nd International Conference, Signals, Circuits and Systems, SCS 2008.*, pp. 1 – 5, 2008.

[116] J. H. Park, Hsiao-Hwa, and M. Atiquzzaman, "AComparative Analysis of HC-128 and Rabbit Encryption Schemes for Pervasive Computing in WAN Environment," *Advances in Information Security and Assurance(Springer), Third International Conferenec and Workshops,*, pp. 682–691, 2009.

[117] Y. Liu and T. Qin, "TheKey and IV Setup of the Stream Ciphers HC-256 and HC-128," *09. International Conference, Networks Security, Wireless Communications and Trusted Computing, 2009. NSWCTC '*, vol. 2, pp. 430 – 433, 2009.

[118] E. Zenner, "A Cache Timing Analysis of HC-256," *15th Annual International Workshop, SAC 2008, Selected Areas in Cryptography*, pp. 199–213, 2008.

[119] G. Meiser, T. Eisenbarth, K. Lemke-Rust, and C. Paar, "Efficient Implementation of eSTREAM Ciphers on 8-bit AVR Microcontrollers," *Industrial Embedded Systems, 2008. SIES 2008. International Symposium*, pp. 58 – 66, 2008.

[120] K. Chain, "The Study and Security Analysis of HC Stream Cipher," *Journal of Convergence Information Technology,*, vol. 6, pp. 439–454, 2011.

[121] H. Wu, "A New Stream Cipher HC-256," pp. 81–85, 1995.

[122] B. Zoltak, "VMPC One-Way Functiion and Stream Cipher," *Fast Software Encryp-tion, FSE 2004, LNCS 3017, Springer-Verlag*, pp. 210–225, 2004.

[123] Y. Tsunoo, T. Saito, H. Kubo, M. Shigeri, T. Suzaki, and T. Kawabata, "The Most Efficient Distinguishing Attack on VMPC and RC4A," 2005.

[124] L. Chen and R. Zhang, "A Fast Encryption Mode for Block Cipher with Integrity Authentication," *IEEE International Conference , Service Operations and Logistics, and Informatics, 2008. IEEE/SOLI 2008.*, pp. 573 – 576, 2008.

[125] W. Trappe and L. C.Washington, *Introduction to Cryptography: With Coding Theory*. 2006.

[126] J. Dj.Golić, "How to Construct Cryptographic Primitives from Stream Cipers," *Computers & Security*, vol. 20, No.1, pp. 79–89, 2001.

[127] R. Doomun, J. Doma, and S. Tengur, "AES-CBC Software Execution Optimization," *Information Technology,2008.ITSim 2008.International Symposium*, pp. 1–8, 2008.

[128] M. Dworkin, "Recommendation for Block Cipher Modes of Operation," *NIST Special Publication 800-38A 2001 Edition*, 2001.

[129] H. M. Heys, "Analysis of the Statistical Cipher Feedback Mode of Block Ciphers," *IEEE Transactions on Computers*, vol. 1, pp. 77–92, 2003.

[130] W.-E. Ghnaim, N. Ghali, and A. Hassanien, "Known-ciphertext Cryptanalysis Approach for the Data Encryption Standard Technique," *Computer Information Systems and Industrial Management Applications (CISIM), 2010 International Conference*, pp. 600 – 603, 2010.

[131] L. Keliher, H. Meijer, and S. Tavares, "Provable Security of Substitution-Permutation Encryption Networks Against Linear Cryptanalysis," *Electrical and Computer Engineering, 2000 Canadian Conference*, vol. 1, pp. 37 – 42, 2000.

[132] A. Bechtsoudis and N. Sklavos, "Side Channel Attacks Cryptanalysis against Block Ciphers Based on FPGA Devices," *VLSI (ISVLSI), 2010 IEEE Computer Society Annual Symposium*, pp. 460 – 461, 5-7 July 2010.

[133] J. Kelsey, B. Schneier, D. Wagner, and C. Hall, "Side Channel Cryptanalysis of Product Ciphers," *Journal of Computer Security*, vol. 8 Issue 2,3, August 2000.

[134] P. Junod, "On the Complexity of Matsuis Attack," in *in Selected Areas in Cryptography, SAC 2001*, pp. 199–211, Springer-Verlag, 2001.

[135] A. Kahate, *Cryptography and Network Security*. Tata McGraw-Hill Comapny Limited, 2008.

[136] A. Biryukov and E. Kushilevitz, "From Differential Cryptanalysis toCiphertext-Oly Attacks," *H.Krawczyk(Ed):CRYPTO'98,LNCS 1462*, pp. 77–88, 1998.

[137] W. Abd-Elmonim, N. Ghali, Hassanien, and A. A.E. ; Abraham, "Known-plaintext attack of DES-16 using Particle Swarm Optimization," *Third World Congress on Nature and Biologically Inspired Computing (NaBIC)*, pp. 12–16, 2011.

[138] M. Ahmad, "Cryptanalysis of Chaos Based Secure Satellite Imagery Cryptosystem," *Contemporary Computing , 4th International Conference, IC3 2011, Noida, India*, pp. 81–91, 2011.

[139] J. Daemen, M. Lamberger, N. Pramstaller, V. Rijmen, and F. Vercauteren, "Computational Aspects Of The Expected Differential Probability Of 4-Round AES And AES-like Ciphers," *Mathematics Subject Classification, Springer*, pp. 85–104, 2009.

[140] Z. Chen, A. Youssef, and S. Tavares, "Differential-like Cryptanalysis of a Class of Substitution-Permutation Networks ," *Electrical and Computer Engineering, 1998. IEEE Canadian Conference*, vol. 1, pp. 433 – 436, 24-28 May 1998.

[141] A. G. Bafghi, R. Safabakhsh, and B. Sadeghiyan, "Finding the Differential Characteristics of Block ciphers with Neural networks," *Information Sciences*, vol. 178, pp. 3118–3132, 2008.

[142] A. Biryukov and E. Kushilevitz, " From Differential Cryptanalysis to Ciphertext-only Attacks," in *Advances in Cryptology-CRYPTO'98*, pp. 72–88, 1998.

[143] E. Biham and Adi, "Differential Cryptanalysis of DES-like Cryptosystems ," *Journal of Cryptology*, vol. 4, pp. 3–72, 1991.

[144] H. Zhihua, Q. Zhongping, and H. Haiqing, "Impossible Differential-Algebraic cryptanalysis of Serpent," *MINES '09. International Conference. Multimedia Information Networking and Security, 2009.*, vol. 2, pp. 353 – 357, 18-20 Nov. 2009.

[145] M. Masoumi and S. Mohammadi, "A New and Efficient Approach to Protect AES Against Differential Power Analysis," *Internet Security (WorldCIS), 2011 World Congress*, pp. 59 – 66, 21-23 Feb. 2011.

[146] W. Shahzad, A. B. Siddiqui, and F. A. Khan, "Cryptanalysis of Four-Rounded DES using Binary Particle Swarm Optimization," *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference*, pp. 2161–2166, July 2009.

[147] G. Piret and F.-X. Standaert, " Provable Security Of Block Ciphers Against Linear Cryptanalysis: A Mission Impossible? An experimental review of the practical security approach and the key equivalence hypothesis in linear cryptanalysis," *Springer*, vol. 2009, pp. 225–338.

[148] M. Matsui, "New Structure of Block Ciphers with Provable Security against Differential and Linear Cryptanalysis," *in Fast Software Encryption*, pp. 205–218, 1996.

[149] S. Landau, "Technical Opinion : Designing Cryptography for the New Century," *Communications of the ACM*, vol. 43 Issue 5, pp. 115–120, May 2000.

[150] A. Albassal and A.-M. Wahdan, "Neural Network Based Cryptanalysis of a Feistel Type Block Cipher," *Electrical, Electronic and Computer Engineering, 2004. ICEEC '04. 2004 International Conference*, pp. 231 – 237, 2004.

[151] H. M. Heys, "A Tutorial on Linear and Differential Cryptanalysis," *Cryptologia*, pp. 189–221, 2002.

[152] M. Matsui, "On Correlation Between the Order of S-boxes and the Strenght of DES," *in the proceedings of Eurocrypt 1994, Lecture Notes in Computer Science*, vol. 950, pp. 366–375, May,1994.

[153] H. M. Heyst and S. E. TavaresS, "The Design of Substitution-Permutation Networks Resistant to Differential and Linear Cryptanalysis," *,ACM, Proceedings of the 2nd ACM Conference on Computer and communications security,* 1994.

[154] H. Bizaki, S. Mansoori, and A. Falahati, "Linear Cryptanalysis on Second Round Mini-AES," *2006. ICTTA '06. 2nd Information and Communication Technologies,*, vol. 1, pp. 1958 – 1962, 2006.

[155] A. Jain, R. Duin, and J. Mao, "Statistical Pattern Recognition: A Review," *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, vol. 22, NO. 1, pp. 4–37, Jaunary 2000.

[156] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, ch. Chapter 1. Academic Press, 2006.

[157] M. Friedman and A. Kandel, *Introduction to Pattern Recognition: Statistical, Structural, Neural, and Fuzzy Logic Approaches Front Cover*. World Scientific, 1999.

[158] A. Jain, R. Duin, and J. Mao, "Statistical Pattern Recognition: A review," *IEEE Transactions, Pattern Analysis and Machine Intelligence,*, vol. 22, pp. 4 – 37, 2000.

[159] A. R.Webb, *Statitistical Pattern Recognation*. John Wiley & Sons, 9 Sep 2002.

[160] X. Hui and L. Yan, "Fault Diagnosis for Variable-Air-Volume Systems Using Fuzzy Neural Networks," *Computer Science & Education, 2009. ICCSE '09. 4th International Conference*, pp. 183–188, 2009.

[161] B. W. W. C. X. Ding, "Advanced MDS Based localization Algorithm for Location Based Services in Wireless Sensor Network ," *Ubiquitous Positioning Indoor Navigation and Location Based Service (UPINLBS), 2010*, pp. 1 – 8, 2010.

[162] S. France and J. Carroll, "Two-Way Multidimensional Scaling: A Review ," *IEEE Transactions Systems, Man, and Cybernetics, Part C: Applications and Reviews,*, vol. 41 Issue:5, pp. 644 – 661, Sept. 2011.

[163] L. Toa, X. Shuai, C. Haiyong, and X. Hexu, "Reserach On Improve Multidimensional Scaling Localization Algorithm For Wirless Sensor Network," *Optoelectronics and Image Processing (ICOIP), 2010 International Conference*, vol. 2, pp. 240 – 243, 2010.

[164] J.-F. Meullenet, R. Xiong, and C. J. Findlay, *Multivariate and Probabilistic Analyses of Sensory Science Problems*. Blackwell publiishing, 2007.

[165] A. A. Ahmed, Y. Shang, and H. Shi, "Variants of Multidimensional Scaling for Node Localization," *11th International Conference on Parallel and Distributed Systems, 2005. Proceedings.*, vol. 1, pp. 140 – 146, 20-22 July 2005.

[166] T. Yang, J. Liu, L. Mcmillan, and W. Wang, "A Fast Approximation to Multidimensional Scaling," in *Proceedings of the ECCV Workshop on Computation Intensive Methods for Computer Vision (CIMCV*, 2006.

[167] H. Junfeng, C. Jun, Z. Yafeng, and M. Xue, "A MDS-Based Localization Algorithm for Large-Scale Wireless Sensor Network," *2010 International Conference On Computer Design AndA ppliations (ICCDA 2010)*, vol. 2, pp. 566–570, 2010.

[168] I. Borg and P. J. F. Groenen, *Modern Multidimensional Scaling : Theory and Applications*. Springer Science +Business Media, Inc., 2005.

[169] A. Delorme, "Statistical Methods," *Swartz Center for Computational Neuroscience, INC, University of San Diego California, CA92093-0961, La Jolla, USA. Email: arno@salk.edu.*, pp. 1–23.

[170] L. Moore and B. Ray, "Statistical Methods for Sensitivity and Performance Analysis in Computer Experiments ," *Simulation Conference Proceedings, 1999 Winter*, vol. 1, pp. 486 – 491, 1999.

[171] X. Xie, J. W. K. Ho, C. Murphy, G. Kaiser, B. Xu, and T. Y. Chen, "Testing and Validating Machine Learning Classifiers by Metamorphic Testing.," *Journal of Systems and Software*, vol. 84, issue 4, pp. 544–558, 2010.

[172] N. J.Nilsson and R. Laboratory, *Introduction to Machine Learning*. TextBook, November 3, 1998.

[173] M. Xue and C. Zhu, "A Study and Application on Machine Learning of Artificial Intellligence," *Artificial Intelligence, 2009. JCAI '09. International Joint Conference*, pp. 272 – 274, 25-26 April 2009.

[174] I. H. Witten and E. Frank, *Data Mining : Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2005.

[175] I. H. Witten, E. Frank, L. Trigg, M. Hall, G. Holmes, and S. J. Cunningham, "Weka: Practical Machine Learning Tools and Techniques with Java Implementations," 2006.

[176] R. R. Bouckaert, E. Frank, M. A. Hall, G. Holmes, B. Pfahringer, P. Reutemann, and I. H.Witten, "WEKA-Experiences with a Java Open-Source Project," *The Journal of Machine Learning Research*, pp. 2533–2541, 2010.

[177] Z. Markov and I. Russell, "An Introduction to the WEKA Data Mining System," *ITiCSE06,Bologna, Italy. ACM*, 2006.

[178] M. H. E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA Data Mining Software: An Update," *SIGKDD Explorations Newsletter*, vol. 11 Issue 1, 2009.

[179] I. H. Witten, E. Frank, L. Trigg, M. Hall, G. Holmes, and S. J. Cunningham, "Weka Practical Machine Learning Tools and Techniques with Java Implementations," 1999.

[180] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *SIGKDD Explor. Newsl.*, vol. 11, pp. 10–18, Nov. 2009.

[181] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, SanFrancisco, 2005.

[182] N. D. Marom, L. Rokach, and A. Shmilovici, "Using the Confusion Matrix for Improving Ensemble Classifiers," *IEEE 26-th Convention of Electrical and Electronics Engineers in Israel*, pp. 000555 – 000559, 2010.

[183] E. Baykan, M. Henzinger, L. Marian, and I. Weber, "A Comprehensive Study of Features and Algorithms for URL-Based Topic Classification," *Transactions on the Web (TWEB)*, vol. 5 Issue 3, 2011.

[184] K. P. Murphy, "Naive Bayes classifiers," 2006.

[185] P. Fernandes, L. Lopes, and D. D. A. Ruiz, "The Impact of Random Samples in Ensemble Classifiers," *SAC '10: Proceedings of the 2010 ACM Symposium on Applied Computing*, 2010. Publisher: ACM.

[186] L. Jiang, H. Zhang, and Z. Cai, "A Novel Bayes Model: Hidden Naive Bayes," *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*, vol. 21, NO. 10, pp. 1361–1370, October 2009.

[187] T. Dong, W. Shang, and H. Zhu, "Naive Bayeian Classifier Based on the Improved Feature Weighting Algorithm ," *Advanced Research on Computer Science and Information Engineering: International Conference, CSIE 2011, Zhengzhou, China, May 21-22, 2011. Proceedings, Part 1*, pp. 142–147, 2011.

[188] Y. Wang, J. Hodges, and B. Tang, "Classification of Web Documents Using a Naive Bayes Method," *15th IEEE International Conference. Tools with Artificial Intelligence, 2003. Proceedings.*, pp. 560 – 564, 3-5 Nov. 2003.

[189] J. Cheng and R. Greiner, "Comparing Bayesian Network Classifiers," pp. 101–108, Morgan Kaufmann Publishers, 1999.

[190] R. R. Bouckaert, "Naive Bayes Classification That Perform Well with Continuous Variables," *17th Australian Joint Conference on Artificial Intelligence Caris, Australia advances in artificial intelligence : 1*, pp. 1089–1094, 2004.

[191] M. N. Murty and V. S. Devi, *Pattern Recognition: An Algorithmic Approach (Undergraduate Topics in Computer Science)*. British Library Cataloguing in Publication Data, Springer, 2011.

[192] S.-B. Kim, K.-S. Han, H.-C. Rim, and S. H. Myaeng, "Some Effective Techniques for Naive Bayes Text Classification ," *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*, vol. 18, pp. 1457–1465, 11, NOVEMBER 2006.

[193] H. J. Kim and J. Chang, "Integrating Incremental Feature Weighting into Naive Bayes Text Classifier ," *Proceedings of the Sixth International Conference on Machine Learning and Cybernetics, Hong Kong,*, vol. 2, pp. 1137 – 1143, 19-22 August 2007.

[194] J. C. Cortizo, J. I. Gir'aldez, and M. C. Gaya, "Wrapping the Naive Bayes Classifier to Relax the Effect of Dependences," in *IDEAL'07*, pp. 229–239, 2007.

[195] A. Cufoglu, M. Lohi, and K. Madani, "Classification accuracy performance of Naïve Bayesian (NB), Bayesian Networks (BN), Lazy Learning of Bayesian Rules (LBR) and Instance-Based Learner (IB1) - comparative study ," *Computer Engineering & Systems, 2008. ICCES 2008. International Conference*, pp. 210 – 215, 25-27 Nov. 2008.

[196] M. Lu, K. Hu, Y. Wu, Y. Lu, and L. Zhou, "SECTCS: Towards Improving VSM and Naive Bayesian classifier," *Systems, Man and Cybernetics, 2002 IEEE International Conference*, vol. 5, p. 5, 6-9 Oct. 2002.

[197] L. I. Kunchevaa and J. J. R. íguezb, "Classifier Ensembles for f MRI Data Analysis: An Experiment," *Magnetic Resonance Imaging*, vol. 28, pp. 583–593, 2010.

[198] W. Ding, S. Yu, Q. Wang, J. Yu, and Q. Guo, "A Novel Naive Bayesian Text Classifier," *Information Processing (ISIP), 2008 International Symposiums on Digital Object Identifier*, pp. 78–82, 2008.

[199] X. Zhang, "Improving Svm Learning Accuracy with Adaboost ," *Natural Computation, 2008. ICNC '08. Fourth International*, vol. 3, pp. 221 – 225, 2008.

[200] P. Maji, "Mutual Information-Based Supervised Attribute Clustering for Microarray Sample Classification," *Knowledge and Data Engineering, IEEE Transactions*, vol. 24 , Issue:1, pp. 127 – 140, Jan. 2012.

[201] J. Manikandan and B. Venkataramani, "Design of a Modified One-Against-All SVM Classifier," *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*, pp. 1869 –1874, 2009.

[202] A. Este, F. Gringoli, and L. Salgarelli, "On-line SVM traffic classification," *2011 7th International, Wireless Communications and Mobile Computing Conference (IWCMC)*, pp. 1778–1783, 2011.

[203] C. GU, S. ZHANG, and H. HUANG, "Online Internet Traffic Classification Based on Proximal SVM," *Journal of Computational Information Systems*, pp. 2078–2086, 2011.

[204] P.-Y. Zhao and Y.-S. Ding, "Using a Fuzzy Support Vector Machine Classifier to Predict Interactions of Membrane Protein," *3rd International Conference. Bioinformatics and Biomedical Engineering , 2009. ICBBE 2009.*, pp. 1 – 4, 2009.

[205] I. M. de Diego, J. M.Moguerza, and A. M. noz, "Combining Kernel Information for Support Vector Classification," *5th international workshop, Multiple classifier systems: , MCS 2004, Cagliari, Italy, June 2004 : proceedings*, vol. 5, pp. 102–111, 2004.

[206] M. Ektefa, S. Memar, F. Sidi, and L. Affendey, "Intrusion Detection using Data Mining Techniques ," *Information Retrieval & Knowledge Management, (CAMP), 2010 International Conference*, pp. 200–203, 17-18 March 2010.

[207] S. Abe, *Support Vector Machines for Pattern Classification.* Birtish Library Catakoguing in Publication Data, 2010.

[208] V. N. Vapni, A. B. Labs, and N. Holmdel, *The Nature of Statistical Learning Theory* . Springer-Verlag New York, Inc. New York, NY, USA ©1995, 1999.

[209] A.-J. Annema, K. Hoen, and H. Wallinga, "Learning Behavior and Temporary Minima of Two-Layer Neural Networks," *CONTRIBUTED ARTICLE,,* vol. 7, pp. 1387–1404, 1994.

[210] J. Wang and Z. Xub, "New Study on Neural Networks: The Essential Order of ApproximationI," *Neural Networks*, vol. 23,Issue 5, pp. 618–624, 2010.

[211] M. Kordos, "Analysis of MLP Based Hierarchical Phoneme Posterior Probability Estimator," *PhD Thesis*, p. 2005.

[212] C. Stergiou and D. Siganos, "Neural Networks," ac 14Mar 2012.

[213] S. Osowski, F. Siwek, and T. Markiewicz, "MLP and SVM Networks-a Comparative Study," *Proceedings of the 6th Signal Processing Symposium-NORSIG 2004, Espoo, Finland*, vol. 37-40, 2004.

[214] L. Holmström, P. Koistinen, J. Laaksonen, and E. Oja, "Neural and Statistical Classifiers-Taxonomy and Two Case Studies," *IEEE TRANSACTIONS ON NEURAL NETWORKS*, vol. VOL. 8, NO. 1,, pp. 5–17, JANUARY 1997.

[215] A. Frank and A. Asuncion, "UCI machine learning repository," 2010.

[216] G. P. Zhang, "Neural Networks for Classification: A Survey," *IEEE Transactions on Systems , Man, and Cyberntics-PART : Applications and Reviews*, vol. VOL. 30, NO. 4, pp. 451–462, NOVEMBER 2000.

[217] G. Shmueli, N. R. Patel, and P. C. Bruce, *Data Mining for Business Intelligence: Concepts, Techniques, and Applications in Microsoft Office Excel with XLMiner.* John Wiley & Sons. Inc,. Hoboken , New Jersey, 2010.

[218] L. Breiman., "Bagging predictors. Machine Learning," vol. 24(2), pp. 123–140, 1996.

[219] D. W. Aha and D. Kibler, "Instance-Based Learning Algorithms," in *Machine Learning*, pp. 37–66, 1991.

[220] F. Zaman and H. Hirose, "A Robust Bagging Method Using Median as a Combination Rule," *IEEE 8th International Conference on Computer and Information Technology Workshops*, pp. 55 – 60, 2008.

[221] D. Opitz, "Bagging Classifiers," 8 1999.

[222] Q. He, F.-Z. Zhuang, X.-R. Zhao, and Z.-Z. Shi, "Enhanced Algorithm Performance for Classification Based on Hyper Surface using Bagging and Adaboost ," *Sixth International Conference on Machine Learning and Cybernetics, Hong Kong,*, vol. 6, pp. 3624 – 3629, 19-22 August 2007.

[223] S. Ruggieri, "Efficient C4.5 [classification algorithm] ," *IEEE Transactions on Knoweldge and Data Engineering, vol. 14, No.2, 2002.*, vol. 14,no.2, pp. 438 – 444, 2002.

[224] M. Last, A. Kandel, and H. Bunke, *Artificial Intelligence Methods In Software Testing (Google eBook).* World Scientific, 2004.

[225] H. Wang, P. Li, and T. Zhang, "Histogram feature-based Fisher linear discriminant for face detection," *Neural Comput & Applic*, vol. 17, pp. 49–58, 2008.

[226] J. Rodriguez, L. Kuncheva, and C. Alonso, "Rotation Forest: A New Classifier Ensemble Method," *Pattern Analysis and Machine Intelligence, IEEE Transactions*, vol. 28, pp. 1619 – 1630, Oct. 2006.

[227] Y. Freund and R. E. Schapire, "A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting," 1997.

[228] X. Wang, C. Wu, C. Zheng, and W. Wang, "Improved Algorithm for Adaboost with SVM Base Classifiers ," *Cognitive Informatics, 2006. ICCI 2006. 5th IEEE International Conference*, vol. 2, pp. 948 – 952, 2006.

[229] Y. Freund and R. E. Schapire, "A Short Introduction to Boosting," *Journal of Japanese Society for Artificial Intelligence*, vol. 14, pp. 771–780, 1999.

[230] P. Aby, A. Jose, L. Dinu, J. John, and G. Sabarinath, "Implementation and Optimization of Embedded Face Detection System ," *Signal Processing, Communication, Computing and Networking Technologies (ICSCCN), 2011 International Conference*, pp. 250 – 253, 21-22 July 2011.

[231] S. Shan, P. Yang, X. Chen, and W. Gao, "Adaboost Gabor Fisher Classifier for FaceRecognition," in *Proc. IEEE Int. Workshop Analysis and Modeling of Faces and Gestures, 2005*, pp. 278–291, 2005.

[232] J. Rodríguez, L. Kuncheva, and C. Alonso, "Rotation Forest and Random Oracles: Two Classifier Ensemble Methods ," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, pp. 1619–1630., 2006.

[233] N. Abu-halaweh and R. Harrison, "Practical fuzzy decision trees," *Computational Intelligence and Data Mining, 2009. CIDM '09. IEEE Symposium*, pp. 211–216, 2009.

[234] Z. Xiaoliang, W. Jian, Y. Hongcan, and W. Shangzhuo, "Research and application of the improved algorithm C4.5 on Decision tree," *Test and Measurement , 2009, ICTM'09. International Conference*, vol. 2, pp. 184–187, 2009.

[235] M. N. Anyanwu and S. G. Shiva, "Comparative Analysis of Serial Decision Tree Classification Algorithms," *IEEE ASSP Magazine*, vol. 22, pp. 230–239, 1993.

[236] G. Stiglic and P. Kokol, "Effectiveness of Rotation Forest in Metalearning Based Gene Expression Classification," *Twentieth IEEE International Symposium on Computer-Based Medical Systems, CBMS '07.*, pp. 243 – 250, 2007.

[237] S. B. Kotsiantis, "Supervised Machine Learning: A Review of Classification Techniques," *informatica, An International Jornal of Computing and Information*, vol. 31, issue 3, pp. 249–268, 2007.

[238] A. Dileep and C. Sekhar, "Identification of Block Ciphers using Support Vector Machines," *IJCNN '06. International Joint Conference ,Neural Networks, 2006.*, pp. 2696 – 2701, 30 October 2006.

[239] R. Spillman, M. Janssen, B. Nelson, and M. Kepner, "Use of A Genetic Algorithm in the Cryptanalysis of Simple Substitution Ciphers," *Cryptologia*, vol. 17 Issue 1, pp. 31–44, Jan. 1993.

[240] A. Cufoglu, M. Lohi, and K. Madani, "A Comparative Study of Selected Classifiers with Classification Accuracy in User Profiling," *Computer Science and Information Engineering, 2009 WRI World Congress*, vol. 3, pp. 708 – 712, 24 July 2009.

[241] B.-J. Wang, H.-J. Cao, Y.-H. Wang, and H.-G. Zhang, "RANDOM NUMBER GENERATOR OF BP NEURAL NETWORK BASED ON SHA-2 (512)," *Proceedings of the Sixth International Conference on Machine Learning and Cybernetics, Hong Kong*, vol. 5, pp. 2708–2712, 2007.

[242] J. J. M. Chan, B. Sharma, J. Lv, G. Thomas, R. Thulasiramand, and P. Thulasiraman, "True random number generator using GPUs and Histogram equalization techniques," *2011 IEEE International Conference on High Performance Computing and Communications*, pp. 161–170, 2011.

[243] T. Matthews, "Suggestion for Random Number Generation in Software," tech. rep., An RSA Data Security Engineering Report, 1995. accessed 15-01-2013.

[244] V. Mises and M. V. Lambalgen, "Definition of Random Sequences Reconsidered," *Journal of Sybolic Logic*, vol. 52,Issue 3, pp. 725–755, 1987.

[245] F. Zafar, M. Olano, and A. Curtis, "GPU Random Numbers via the Tiny Encryption Algorithm," *High Performance Graphics (2010)*, pp. 133–141, 2010.

[246] D. E. Holcomb, W. P. Burleson, and K. Fu, "Power-Up SRAM State as an Identifying Fingerprint and Source of True Random Numbers," *IEEE TRANSACTIONS ON COMPUTERS*, vol. 58, NO. 9, pp. 1198–1120, 2009.

[247] J.-M. Tsai and J. Tzeng, "AUDIO RANDOM NUMBER GENERATOR AND ITS APPLICATION ," *Proceedings of the 2011 International Conference on Machine Learning and Cybernetics, Guilin*, vol. 4, pp. 1678–1683, 2011.

[248] J. Tzeng, I.-T. Chen, and J.-M. Tsai, "Random Number Generator Designed by the Divergence of Scaling Functions," *Intelligent Information Hiding and Multimedia Signal Processing, 2009. IIH-MSP '09. Fifth International*, pp. 1038–1041, 2009.

[249] RANDOM.ORG, "What's This Fuss About true Randomness." Valid XHTML 1.0 Transitional | Valid CSS.

[250] D. Hussain, "An Analasis of Stream-Based & Paralle Approaches for Statistical Analysis of Random Number Generator Output," Master's thesis, School of Computer Sciences, Bangor University, 2012.

[251] J. Soto, "Statistical Testing of Random Number Generators ," in *Proceedings of the 22nd National Information Systems Security Conference*, 1999.

[252] M. Haahr, "Random." Last accessed 2nd January 2013.

[253] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, and S. Vo, " Statistical Test Suite For Random And Pseudorandom Number Generators For Cryptographic Applications. Statistical Test Suite For Random And Pseudorandom Number Generators For Cryptographic Applications," *NIST Special Publication 800-22*, 2001.

[254] J. Lan, W. L. Goh, Z. H. Kong, and K. S. Yeo, "A Random Number Generator for Low Power Cryptographic Application ," *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium*, pp. 328–331, 2010.

[255] B. F. J. Gravetter and L. B. Wallnau, "Statistics for the Behavioral Sciences," 2009.

[256] L. Cohen, L. Manion, D. K. Morrison, and K. R. B. Morrison, "Research Methods Education." GoogelBook.

[257] D. Salomon, "Data Compression: The Complete Reference," 2006. Googelbook.

[258] J. Cleary, W. Teahan, and I. Witten, "Unbounded Length Contexts for PPM," *Data Compression Conference, 1995. DCC '95. Proceedings*, pp. 52–61, 1995.

[259] A. S. vs. API Developer Documentation, "Interface ThreadMXBean," *JavaTM 2 Platform Standard Ed. 5.0.* accessed 16-01-2013.

[260] N. Kofahi, T. Al-Somani, and K. Al-Zamil, "Performance Evaluation of three Encryption/Decryption Algorithms," *Sixth International Conference,Networked Computing and Advanced Information Management (NCM), 2010*, vol. 2, pp. 790–793, 2003.

[261] D. Hook, *Beginning Cryptography in Java (Programmer to Programmer)* . Wiley, 19 Aug 2005.

[262] G. Contributor, "Master the basics of Java Cryptography Extension (JCE)." http://www.techrepublic.com/article/master-the-basics-of-java-cryptography-extension-jce/1046088, October 14, 2003, 7:57pm PDT. Accessed (02/01/2012).

[263] "JavaTM Cryptography Extension (JCE) Reference Guide," 2003, 2010. Last modified: 10 Jan 2002.

[264] T. Winters, T. C. C. operative Ltd, and G. Hook, "The Legion of the Bouncy Castle." Java and JCE are registered trademarks of Oracle ®.

[265] H. D. Ltd., "What is Base 64 Encoded Data?," 2012. Last Updated : Friday, 12-Oct-2012.

[266] O. Java, "Java Cryptography Extension Reference Guide ." Copyright © 2003, 2010 Oracle and/or its affiliates. All rights reserved., 2010. Last modified: 10 Jan 2002.

[267] W. Dai, "Crypto++ Library 5.6.1." http://www.cryptopp.com/, 9 2010. Last modified: 8/9/2010.

[268] W. R. Yount, *Research design and statistical analysis for Christian ministry*. 2006.

[269] B. Ryabko, V. Stognienko, and Y. Shokin, "A new test for randomness and its application to some cryptographic problems," *Journal of Statistical Planning and Inference*, vol. 123, pp. 365–376, 2004.

[270] K. Wongpanya, K. Sripimanwat, and K. Jenjerapongvej, "Simplification of Frequency Test for Random Number Generation Based on Chi-Square," *The Fourth Advanced International Conference on Telecommunications*, 2008.

[271] A. Sayyed and S. Agarwal, "PPM Revisited with New Idea on Escape Probability Estimation," *International Conference on Computational Intelligence and Multimedia Applications,ICCIMA*, vol. 4, pp. 152–156, 2007.

[272] N. D. Marom, L. Rokach, and A. Shmilovici, "Using the Confusion Matrix for Improving Ensemble Classifiers," *2010 IEEE 26-th Convention of Electrical and Electronics Engineers in Israel*, pp. 000555–000559, 2010.

[273] L. Chen and H. Tang, "Improved Computation of Beliefs based on Confusion Matrix for Combining Multiple Classifiers," *ELECTRONICS LETTERS*, vol. 40 No. 4, pp. 1–2, 2004.

[274] I. Kukenys, W. N. Browne, and M. Zhang, "Confusion Matrices for Improving Performance of Feature PatternClassifier Systems," in *Proceedings of the 13th annual conference companion on Genetic and evolutionary computation*, GECCO '11, (New York, NY, USA), pp. 181–182, ACM, 2011.

[275] D. Bowes, T. Hall, and D. Gray, "Comparing the performance of fault prediction models which report multiple performance measures: recomputing the confusion matrix," *Proceedings of the 8th International Conference on Predictive Models in Software Engineering*, pp. 109–118, 2012.

[276] A. Borkar, R. Kshirsagar, and M. Vyawahare, "FPGA Implementation of AES Algorithm," *Electronics Computer Technology (ICECT), 2011 3rd International Conference*, vol. 3, pp. 401–405, 2011.

[277] W. Diffie and M. Hellman, "New directions in cryptography ," *Information Theory, IEEE Transactions*, vol. 22, Issue: 6, pp. 644 – 654, 1976.

[278] R. A. R. Manjula, "Identification of Encryption Algorithm Using Decision Tree," *First International Conference on Computer Science and Information Technology, CCSIT 2011, Proceedings, Part III, CCIS 133.*, pp. 237–246, 2011.

[279] S. Dean and B. Illowsky, "Descriptive Statistics: Histogram," 2011. Last edited by Susan Dean on Aug 11, 2011 7:15 pm GMT-5.

[280] F. Buccafurri and G. Lax, "Fast range query estimation by N-level tree histograms," *Data & Knowledge Engineering*, vol. 51, Issue 2, pp. 257–275, November 2004.