A Genetic Algorithmic Approach to the Differential & Linear Cryptanalysis

By

Mete EMİNAĞAOĞLU

A Dissertation Submitted to the
Graduate School in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

Department: Computer Engineering
Major: Computer Software

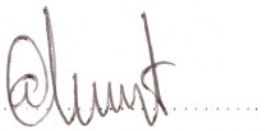
İzmir Institute of Technology

İzmir, Turkey

June, 1999

We approve the thesis of **Mete EMİNAĞAOĞLU**

Date of Signature

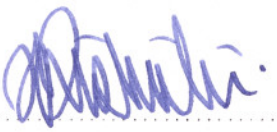


Assist. Prof. Dr. Ahmet KOLTUKSUZ

Supervisor

Department of Computer Engineering

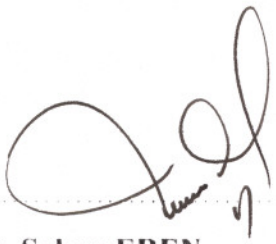
22.06.1999



Prof. Dr. Halis PÜSKÜLCÜ

Department of Computer Engineering

22.06.1999

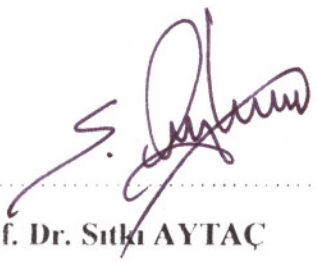


Prof. Dr. Şaban EREN

Ege University

Department of Computer Engineering

22.06.1999



Prof. Dr. Sıtkı AYTAÇ

Head of Department

22.06.1999

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my advisor *Asst. Prof. Ahmet KOLTUKSUZ, Phd.*, for his enduring support, advice, encouragement and supervision throughout the hard years of this study.

I do thank to all the academic staff of my department who have contributed to the achievement of this thesis.

I also owe a special thanks to my dear fiancee, *Miss Neslihan PİRİMOĞLU*, for her support, encouragement and patience.

ABSTRACT

The two most well known and recently developed methods in cryptanalysis of DES and DES-like symmetric block ciphers are differential and linear cryptanalysis. But these cryptanalytic attacks need to be improved due to the computational performance and storage capacity problems. On the other hand, genetic algorithms can be a good solution in cases where the optimum value or near-optimum solutions are sought in complex systems or for non-linear problems. This is a valid situation for the cryptanalysis case where DES and DES-like ciphers are non-linear in structure making differential and linear cryptanalysis a complex system with a very large search landscape and extreme amount of conditional and probabilistic candidates for the key being sought.

In this study, a new and promising method with better performance is to be developed for differential / linear cryptanalysis of DES and similar symmetric cryptosystems exploiting genetic algorithms' broadened search and optimum finding capacity.

ÖZ

DES ve DES benzeri simetrik blok şifrelerin kriptanalizinde yakın zamanda geliştirilen ve en iyi bilinen iki yöntem diferansiyel ve lineer kriptanalizdir. Fakat, büyük miktarda yer gereksinimi ve hesaplama performansındaki düşüklükler gibi nedenlerden ötürü bu kriptanalitik saldırıların geliştirilmesine ihtiyaç vardır. Öte yandan genetik algoritmaların, optimum değerin veya optimuma yakınsayan çözümlerin arandığı kompleks sistemler ya da lineer olmayan problemler için iyi bir çözüm yöntemi olduğu bilinmektedir. Bu durum aynı zamanda kriptanaliz uygulamaları için de söz konusudur; şöyle ki, DES ve DES benzeri şifrelerin lineer olmayan yapılarından ötürü lineer ve diferansiyel kriptanaliz yöntemleri kompleks bir sisteme dönüşmekte, oldukça büyük tarama alanları kapsamında aranan anahtar için çok sayıda durumsal ve belirli olasılıkta aday değerler bulunmaktadır.

Bu çalışmada, genetik algoritmaların arama ve optimum sonucu bulma gücünden yararlanılarak DES ve benzeri simetrik şifre sistemlerinin diferansiyel / lineer kriptanalizinde daha başarılı ve etkili sonuçlar sağlayan yeni bir yöntem geliştirilmesi amaçlanmıştır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

1.1 Motivation

In today' s cryptology, new techniques have been developed and yet being developed involving cryptanalysis of DES and other symmetric ciphers. All such methods' aim is to find out a complete weakness or a big security hole in these cryptosystems in order to prove that both in theory and practice, these systems are insecure and must be replaced with entirely different and enhanced models. Differential and linear cryptanalysis are two effective methods generated for the cryptanalysis of DES and DES-like ciphers. However, these methods are not very practical and their performances must be improved in some way or another. For instance, it' s proven that differential cryptanalysis of DES requires at least 2^{47} chosen plaintexts and 2^{37} computational complexity; linear cryptanalysis of DES needs at least 2^{43} - 2^{47} known plaintexts and equivalent computational complexity. Although these values are proven to be better than brute-force attacks or other cryptanalytic attacks against DES, they are still far from providing a feasible computation time as well as being impractical and bringing storage overhead whenever the plaintext / ciphertext data requirements are in concern. In order to implement these attacks successfully and break DES in acceptable time limits, gigantic computing power and storage is required and this platform cannot be obtained easily. Thus, results with much better performance values and more realistic requirements are still being awaited reluctantly for the differential and linear cryptanalysis.

Besides increasing the cracking speed and decreasing the storage consumption, it' s still argued in common that with the aid of an improved variant of either differential or linear cryptanalysis, a proof of entire weakness in the design of DES and similar symmetric block ciphers might be achieved. This is, in fact, the main goal of all the cryptanalysts involved in breaking DES and similar block ciphers.

On the other hand, genetic algorithms (GA' s) are well known for their high performance in searching and optimum-finding capacities, especially in probabilistic problems, complex systems and in various conditions where good and robust estimations and predictions are necessary. Especially, whenever the best possible solution or a set of solution candidates are sought for complex or non-linear problems while providing high performance within feasible time limits, GA' s are proven to be a good alternative where all the other alternative methods, algorithms might be proven to be inefficient, or even might not exist. It' s also sometimes possible that the optimum value or the exact solution being searched lies within a very large exploration space with a lot of possible candidates or pseudo-optimum values which makes the standard search algorithms such as simulated-annealing, and probabilistic estimation methods both inefficient and unreliable. In such cases, GA is taken into account as a new promising alternative. GA' s sometimes outperform all the other heuristic and probabilistic methods and artificial intelligence algorithms in problems like best-path, travelling salesman, minimum cost - maximum throughput or maximum gain, load balancing, stock market modelling, cost estimations amongst various applications and so on. Most of these problems resemble the cryptanalysis phenomenon in structure,

where the best solution or a set of best possible solutions are required within the best seek-time and highest efficiency with the lowest cost. In all types of cryptanalytic attacks, the exact key or candidate key values or best possible plaintext data are being sought where no formal solution or direct formulation for the solution of the problem exists. Even, there are millions or billions of possible values or combinations for the searched key value and no efficient probabilistic method is in hand for a strong symmetric block cipher algorithms such as DES. The differential and linear cryptanalysis exploits the measurements of probabilities and conditional probabilities of possible key / subkey values, but the resulting search space is again so large that makes the implementations usually impractical and inefficient. This space should be narrowed significantly and the seek time and storage requirements must be greatly decreased with enhanced probabilistic methods and search algorithms which; GA' s seem to be the one of the best choices. Thus, implementing a new model with genetic algorithms for the cryptanalytic searches and embedding it into the differential / linear cryptanalysis; the performance and efficiency of these cryptanalytic methods might be improved.

In fact this is a new idea that has never been tried or no similar study has been done before, at least at the time of writing this thesis; it has been first brought up and recommended by my advisor, *Asst. Prof. Ahmet KOLTUKSUZ, Ph. D.*, Genetic algorithms were used only in a few applications among cryptanalysis, such as breaking a Rotor Cipher or simple substitution ciphers. Yet, no search has been done involving cryptanalysis of DES or similar symmetric block ciphers with GA as well as implementing GA in differential / linear cryptanalysis. Even constructing a hypothetical model for the GA usage in differential / linear cryptanalysis will be a study that has never been carried out before. If any of the existent ones can be used or a new GA model can be derived and applied to the cryptanalysis of symmetric block ciphers, this will probably open new extensions in cryptanalytic applications. The nature of the problem faced in differential and linear cryptanalysis seems more or less related to the GA-based approaches. However, the problem is very hard to implement with a suitable GA model, and applying this to the standard cryptanalytic algorithms and hence deriving any successful or unsuccessful results is noticeably difficult. But against all odds, dealing with a difficult cryptological problem from a very different point of view and trying to bring new visions to both GA and cryptanalysis has been a strong motivation and desire for me to work on this subject.

1.2 Genetic Algorithms & Differential / Linear Cryptanalysis

Genetic Algorithm, (GA) is an alternative and competitive method in software technology to the neural networks and various artificial intelligence (AI) algorithms, or as some do say so, an AI variant. It derives its roots from the fundamental theorems and models of biology, genetics and genetic engineering. The implementations of these models, facts and functions to the computer algorithms / software has founded a new era.

GA' s are especially proven to be useful in solving problems where straightforward software mechanisms, linear algorithms, any kind of formulations or any direct solution to the problem via computers are impossible or too inefficient. For example, no mathematical formulation or linear algorithm is found for the load

balancing problem in networks, but GA' s, as well as neural networks, can be applied to the system and can enable the solution of this problem accurately and reliably. The only problem with GA' s is that, a suitable GA model cannot be driven for some types of non-linear or complex problems. In addition, it' s very difficult sometimes to establish an efficient GA model or implementation for some of the search and optimum-finding problems. For the algorithm designer, this is the most important and the hardest fact that must be dealt with. Another fact is that since GA' s are usually changeable due to the problem they are applied, no or very few theoretical foundations, standards, formal models or basic structures exist. GA' s are proven to be much more implementational rather than theoretical, so one cannot deduce which GA model might be best or more useful for the problem involved, or whether any of the GA' s is applicable or not until he / she makes trials with each of the alternative GA models. This trial / error approach may be considerably exhaustive and time-consuming sometimes and might leave the designer or programmer with unsatisfactory results.

On the other hand, the flexibility of GA both in design and application imposes a good mark for software engineers. Also, giving the designer great freedom to extend his GA model without limits and to provide new alternative techniques, models, functions for any part of a GA, there always exists the hope of a better GA solution with a leap in performance.

The design, structure and implementation of genetic algorithms will be discussed and analyzed in detail later in Chapter 4, but a simple definition of GA is included here. The construction of a GA for any problem is as follows; First, the problem in question must be defined with all its variables, constants, functions, etc.; where this problem can be any scientific, industrial, application-based, or something else. Second, all the parameters and portions of the problem must be redefined as a GA model components, thus the real model of the problem must be mapped into a GA-system model. Third, the components of the GA model must be chosen among several alternatives, such as which genotype, phenotype implementation will be used for the parameters, which methods will be used in selecting, searching mechanisms, which probabilistic functions might be exploited, and what kind of fitness function shall be used, etc. If necessary, new mechanisms, new genotype-phenotype models will have to be derived by the designer. And the last stage is applying this GA model into the software and making tests until obtaining the sufficient results. Throughout these tests, any necessary changes are possible in the previously formed GA model, even that the whole GA implementation for the problem might be redesigned and rebuilt.

Sometimes, early built GA models with implementations in several computer languages, GA libraries or GA tools might be useful for the solution of the problem. But if this problem is not similar to the ones in those tools or libraries, then everything must be developed by the designer or programmer. This was also true for this project where no toolbox, library or generated applications was available for a GA model applied to cryptanalysis. It should be noted that even some of the prototypes in GA libraries such as chromosome and gene representation and selecting mechanisms are the same with the ones I used, I preferred coding and declaring everything by myself throughout this study.

The cryptanalysis of symmetric block ciphers has been in question since the invention and usage of DES and similar cryptosystems. The common goal is to derive an attack methodology faster and more efficient than brute-force attack. The two of these cryptanalytic attacks are differential and linear cryptanalysis which were invented in the early 90' s and still being used. These attacks are proven to be theoretically successful whereas found to be impractical and infeasible in application; thus they need some improvements.

Differential cryptanalysis is a statistical attack method used in cryptanalysis that can be applied to any iterated mapping, therefore to any symmetric block cipher with several rounds. This method analyzes the effect of particular differences in plaintext pairs on the differences of the resultant ciphertext pairs derived by the symmetric encryption algorithm. These differences are used to assign probabilities to the possible keys and to deduce the most probable key. This method is usually applied to many plaintext / ciphertext pairs having the same particular differences under the same initial encryption key being sought. In other words, differential cryptanalysis looks for pairs of ciphertexts whose corresponding plaintexts have particular differences and analyzes the evolution of these differential values while the plaintexts propagate through the rounds of the iterated cipher algorithm. For DES and DES-like symmetric block ciphers, the difference is achieved by the XOR operation where the differential values are the resultant fixed XOR outputs of the two randomly or specifically chosen plaintext pairs. This attack is mostly used with chosen plaintexts even though can be applied to known plaintexts, because the performance degrades significantly with the known plaintext usage.

Although the differential cryptanalytic attack was proven to be effective against some product ciphers like FEAL, REDOC-II, Lucifer, LOKI both in theory and practice; this technique still needs some strict improvements against DES and its enhanced variants.

After the invention of differential cryptanalysis, again in the early 90' s another cryptanalytic method was generated known as linear cryptanalysis. In the details and implementation, linear cryptanalysis is proven to be quite different from the differential one; however, it' s also shown that they are very similar in the structural level. Linear cryptanalysis can be defined as the attack method which studies the statistical relations between bits of plaintext / ciphertext pairs and the keys used in their encryption. These relations thereafter are used to predict the bit values of the key with calculated probabilities derived from known plaintext / ciphertext pairs. Expressing it in another way, linear cryptanalysis uses linear approximations to describe the action of a symmetric block cipher. For DES and DES-like ciphers, this method works by XORing some of the plaintext bits, XORing some of the corresponding ciphertext bits, and then XORing the outputs of these two, a single bit value will be generated that will be also the XOR of some of the key bits. This is the linear approximation which will have a calculated probability. Thus, by analyzing these probabilities, strong guesses can be made about the key bits. This method requires known plaintext and their corresponding ciphertext data rather than chosen plaintext that' s necessary for differential cryptanalysis.

Linear cryptanalysis is proven to be performing better than differential cryptanalysis against DES and similar symmetric block ciphers. However, this attack is successful only in theory and yet needs improvements and enhancements to make it practical and efficient in real life.

It should be noted that differential and linear cryptanalysis will be dealt in more detail and more technically with mathematical notations, formulations, etc. in Chapter 3.

Both differential and linear cryptanalysis make use of statistical data, probabilistic measures and require efficient search algorithms and better estimators in comparisons and selections. The search speed of both methods must be increased, the exploration space for the key value must be narrowed while improving the differential analysis and strengthening the linear approximations. These might be achieved by embedding special-purpose genetic algorithms with some special fitness and selection functions, estimators, etc. into the differential and linear cryptanalysis schemes.

1.3 Scope and Structure

This study aims to give a brief perspective of; structure, properties and cryptanalysis of symmetric block ciphers (taking DES as the base model), differential and linear cryptanalysis, genetic algorithms as well as discussing how genetic algorithm techniques could be embedded into cryptanalytic attacks or how genetic algorithms might be implemented for the usage of cryptanalysis. The applications and implementations in this study are limited to a case study of a simple block cipher' s cryptanalysis and the usage of genetic algorithms (GA' s) in that case. This is due to the shortcomings in the proposed project, lack of necessary equipment, data, hardware, time problems, etc. which made the observation or tests on differential or linear cryptanalysis of DES or a similar complex cipher and the genetic algorithmic approaches to these attacks, impractical and inapplicable. Therefore, the analyzes involved in differential / linear cryptanalysis with GA' s and hypothetical models related with this concept, assumptions and possible outcomes of such models are submitted from a theoretical standpoint in this thesis. Implementations, tests, and results providing any precise conclusions with the approval or disapproval of; genetic algorithms' usage in differential / linear cryptanalytic attacks against DES and DES-like ciphers is far beyond the scope of this study.

The structure of this thesis is as follows;

In Chapter 1, a brief introduction is made where the reasons and the urge for achieving this study are mentioned, an introduction to the concepts related with this study including the basic terms is given. What' s included in this project and yet, what are beyond this project are explained shortly.

A brief introduction and analysis of cryptanalysis is made in Chapter 2. The general concepts and terms in cryptology as well as cryptanalysis are focused on. Then, a detailed analysis of symmetric block ciphers is made where DES is taken as the pivotal model. The design, theory, implementation, measured security performances and cryptanalytic attacks achieved so far against DES are all explained in several

sections of Chapter 2 so as to give a brief information to the readers about symmetric cryptosystems and their cryptanalysis.

Chapter 3 introduces differential and linear cryptanalysis. The basic concepts, mathematical foundations, the design and basic structure, the algorithms and methods are given. Both these cryptanalytic methods are explained in detail with their theoretical and practical aspects. Several examples of differential and linear cryptanalytic attacks are given with sample case studies previously attained among various symmetric block ciphers, mainly focusing on DES.

In Chapter 4, genetic algorithms are introduced. The nature of genetic algorithms, their basic structure, components, models are explained briefly both in theory and practice with examples. The application areas of genetic algorithms are given at the end of the chapter.

In Chapter 5, previous studies that have been achieved so far concerning cryptanalysis of cryptosystems with genetic algorithms are given in the first section. Then, the proposed model for differential / linear cryptanalysis using genetic algorithms, that was expected to be used in this study, is given. The difficulties, drawbacks and infeasibilities making this model impractical and inapplicable are explained with reasons as well as recommendations for the similar future studies.

Chapter 6 deals with a simple case study that has been carried out. The definition of the model, its implementation and application to several versions of a simple symmetric block cipher that is designed and used, are all explained. The results derived aftermath and the consequences of this test are included in the Results & Discussion section.

Chapter 7 is a general conclusion involving differential / linear cryptanalysis with genetic algorithms, the model proposed in Chapter 5 and the results achieved in Chapter 6. The necessary remarks, advices and assumptions are made for the future studies.

Some mathematical models, functions, algebraic concepts and formulations related with cryptology and cryptanalysis can be found in Appendix A.

In Appendix B, a sample study of the avalanche-effect of 56-bit DES in ECB mode is given which I achieved during the analysis of DES.

Chapter 2

CRYPTANALYSIS

2.1 Introduction to Cryptanalysis

Cryptanalysis is the science of recovering the plaintext of a message without access to the key.[1] Also cryptanalysis deals with all kinds of methodologies, techniques that aim:

- to achieve the key(s) used in any kind of encryption,
- to get or infer the algorithm used in encryption-decryption,
- to deduce the meaning of ciphertext messages,
- to find weaknesses in encryption algorithms.[2]

In any kind of secure system the basic purpose is to encrypt a message or data (that' s referred as plaintext) into a secret form (known as ciphertext) so that no one but the authorized or aimed receiver could get it and read it correctly after a decryption process. In a more terminological sense, the practice of using encryption to conceal text[3], henceforth the achievement of hidden writing is defined as cryptography. There are various topics in cryptography but the most common and essential points are; the encryption / decryption algorithm, kind of the cipher used, and the key(s) used (if necessary) whilst encryption / decryption.

The person involved in cryptanalysis is called as cryptanalyst. Thus a cryptanalyst' s aim is to break the cipher, to retrieve information from hidden messages or data, and to achieve the key used in the cryptosystem within acceptable time limits, money and energy costs. In fact, a cryptanalyst is an interceptor, he or she can be any legal person as a scientist or researcher, but on the other hand, a cryptanalyst can be an unauthorized and illegal person filled with criminal or immoral desires. A cryptanalyst can use any kind of theoretical, mathematical, heuristic, randomized, etc. methods and strategies whenever necessary; he / she should exploit from any tool, aid, method, data on behalf of himself / herself in order to achieve a successful cryptanalytic attack. The strategies used vary according to the structures of encryption schemes and the availability and the nature of the information.

It must be stressed that the language used in cryptography, the algorithm or the cipher used in encryption, the type, size of the key used in encryption (or whether it' s a keyless system), the amount of available ciphertext or plaintext / ciphertext paired data, the syntactic and semantic pre-knowledge about data, etc. are very essential points and guiding light for a cryptanalyst. Since cryptography blends several area of mathematics: number theory, complexity theory, information theory, probability theory, abstract algebra and formal analysis[4], a cryptanalyst should be good at all these

---

[1] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 5.
[2] Charles P. Pfleeger, *Security in Computing*, pp. 24-25.
[3] *ibid*, p. 24.
[4] Bruce Schneier, "Why Cryptography is harder than it looks", CounterPlane Systems, Technical Report, p. 4, 1997.

and also should have a good knowledge of advanced statistics, calculus, computer sciences, linguistic sciences, social sciences. It should also be noted that there are several important issues which assures a cryptanalysis' success. These are:

- Achievement of some acceptable amount or all of the key, plaintext, or the encipherment algorithm reliably and exactly,
- Fulfilment of the money, time, and effort by the value of the data or information retrieved,
- Retrieval of data within a time limit that doesn' t exceed the useful time of the data or information.

2.2 Cryptanalytic Attack Types

An attempted cryptanalysis is called an attack.[5] There are various kinds of cryptanalytic attacks and most of these attacks all rely on the assumption that the cryptanalyst has complete knowledge of the encryption algorithm used.[6] Some of the cryptanalytic attacks are listed as below which the first five are the most well known ones and they are listed in order of cryptanalytic effectiveness from the weakest to the strongest:

- **Brute-force attack:** This is the most difficult case for a cryptanalyst where he or she has only the ciphertext, and no information is available for the plaintext or the key; even any guesses for the encryption algorithm, data, key is hard to make. The attack method relies on chance where all the possible keys are tried until the successful guess is made. Thus the search time is very long and the methodology is very weak, resulting with NP complexity. For example, for a cipher with a key of *k* bit length, 2^{k} or 2^{k-1} trials are required to recover the messages or to find out the key itself. Sometimes, brute-force attack is referred as a special case of known-plaintext attack[7] where a few or yet a single plaintext / ciphertext pair is used during the trial of each key value.

- **Ciphertext-only attack:**[8-9] The cryptanalyst has the ciphertext of several messages and also might have a-priori knowledge about the key structure or the encryption system. The cryptanalyst tries to retrieve the plaintext of the messages or aims to get the key(s) used in the encryption so as to decrypt the messages encrypted with the same key(s). This can be abbreviated as below (where P_i stands for any plaintext message, C_i is for ciphertext, k is the key used in encryption and decryption, E_k denotes the encryption process or algorithm, and D_k is for decryption, conversely.);

  Given: $C_1 = E_k(P_1), C_2 = E_k(P_2), \ldots, C_n = E_k(P_n)$
  Recover: Either $P_1, P_2, \ldots, P_n$, or k, or an algorithm to
  get P_{n+1} from $C_{n+1} = E_k(P_{n+1})$

---

[5] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 5.
[6] *ibid*, p. 5.
[7] *ibid*, p. 152.
[8] *ibid*, pp. 5-6.
[9] Charles P. Pfleeger, *Security in Computing*, p. 69.

- **Known-plaintext attack:**[10]-[11] Not only the ciphertext, but also its related plaintext form of several messages are all available to the cryptanalyst. The purpose in this type of attack is to get or deduce the key(s) used in encryption, or an algorithm to decrypt any new messages encrypted with the same key(s) with the aid of these plaintext / ciphertext pairs in hand. This attack type is especially useful in cryptanalysis of symmetric cryptosystems like FEAL, DES, etc. and is more effective in linear cryptanalysis of such ciphers.

  Given: P_1, $C_1 = E_k(P_1)$; P_2, $C_2 = E_k(P_2)$; ...; P_n, $C_n = E_k(P_n)$
  Recover: Either k, or an algorithm to get P_{n+1} from $C_{n+1} = E_k(P_{n+1})$

- **Chosen-plaintext attack:**[12]-[13] This is a similar but more advantageous attack method than the previous one where the cryptanalyst has plaintext / ciphertext of several messages and also, has the chance of choosing the plaintext that will be encrypted. Thus, an improved and more powerful analysis of the cryptosystem is possible by the chance of choosing since the cryptanalyst is able to test the plaintext / ciphertext pairs which have special structure enhancing the cryptanalysis. By this way, cryptanalysis can be focused on the plaintext / ciphertext pairs that could possibly give more information about the attacked key(s). The purpose is to deduce the key(s) used in the cryptosystem or an algorithm to decrypt the incoming messages encrypted with the same key(s). As similar to the previous one, this attack type is usually applied to cryptanalysis of symmetric cryptosystems but has much better performance with the usage of differential cryptanalysis rather than linear.

  Given: P_1, $C_1 = E_k(P_1)$; P_2, $C_2 = E_k(P_2)$; ...; P_n, $C_n = E_k(P_n)$,
  where P_1, P_2, ..., P_n are chosen
  Recover: Either k, or an algorithm to get P_{n+1} from $C_{n+1} = E_k(P_{n+1})$

- **Adaptive-chosen plaintext attack:**[14] This is a special case of chosen-plaintext attack. Again, cryptanalyst has plaintext / ciphertext pairs of several messages, and has the chance of choosing the plaintexts but this time, he / she can change his / her choice with respect to the previous encryption results. Thus, in an adaptive manner more optimized choices on the plaintext blocks can be made where a smaller block of plaintext (compared to the previous method) can be chosen and the next block can be selected based on the results of the first block' s encryption, and so on. Again, the aim is to recover the key(s) or to decrypt any messages under the same cryptosystem and with the same key(s).

  Given: P_1, $C_1 = E_k(P_1)$, P_1 is chosen;
  P_2, $C_2 = E_k(P_2)$, P_2 is chosen after checking P_1, C_1;
  ...;
  P_n, $C_n = E_k(P_n)$, P_n is chosen after checking P_{n-1}, C_{n-1}, ..., P_1, C_1;

  Recover: Either k, or an algorithm to get P_{n+1} from $C_{n+1} = E_k(P_{n+1})$

---

[10] Charles P. Pfleeger, *Security in Computing*, p. 69.
[11] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 6.
[12] Charles P. Pfleeger, *Security in Computing*, p. 69.
[13] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 6.
[14] *ibid*, p. 6.

- **Chosen-ciphertext attack:**[15] The cryptanalyst has the chance of choosing different ciphertexts to be decrypted and accessing to the decrypted plaintexts of these. He or she has (and should have) more knowledge or access to the cryptosystem that' s attacked and this is a more powerful attack than the previous ones considering test time and accuracy. As an example, if the cryptanalyst has access to a tamperproof box that enables automatic decryption, he / she can make this type of attack possible. The purpose in this attack is to deduce or get the key(s) used in the cryptosystem. It should be noted that this type of attack is primarily used in asymmetric cryptosystems with public keys but it' s also sometimes effective in symmetric algorithms or cryptosystems. It should also be stressed that sometimes this attack type might fail or cause erroneous results if, in a cryptosystem or algorithm two or more distinct keys can produce the same ciphertext as a result of encrypting different plaintexts.[16] Henceforth, when the cryptanalyst decrypts using this chosen ciphertext he / she can yield to the wrong plaintext and get a wrong key in the end.

  Given: C_1, $P_1 = D_k(C_1)$, C_2, $P_2 = D_k(C_2)$, ..., C_n, $P_n = D_k(C_n)$
  Recover: k

- **Chosen-key attack:**[17] In fact, this attack does not claim that the cryptanalyst has the chance of choosing the keys but rather, it means that the attack is carried out by the help of some knowledge about the relationship between different keys. By the way, this attack is also referred as related-key cryptanalysis[18] where the differences between the keys are examined and the cryptanalyst chooses a relationship between a pair of keys, but does not know the keys themselves. This method is especially effective and successful among DES and DES-like symmetrical ciphers (ie. this attack can break a modified DES variant using 2^{17} chosen-key chosen plaintexts or 2^{33} chosen-key known plaintexts.[19]) whereas it is considered as being strange, obscure and not very practical.[20]

2.3 Cryptosystems

2.3.1 Definition

Since, throughout this chapter it' s mainly focused on cryptanalysis and its applications on the most well known ciphers such as DES, before going into further details a short and simple overview of cryptosystems, the basic terminology and some basic facts considering cryptosystems are introduced in this section. Cryptosystem can be defined as a system both for encryption and decryption. More precisely, a cryptosystem is an algorithm to devise secure information transfer, data hiding, secure messages, etc., plus all possible plaintexts, ciphertexts and key(s) (if necessary).[21] All

---

[15] Bruce Schneier, *Applied Cryptography - Second Edition*, pp. 6-7.
[16] Charles P. Pfleeger, *Security in Computing*, p. 70.
[17] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 7.
[18] *ibid*, p. 290.
[19] *ibid*, p. 290.
[20] *ibid*, p. 7.
[21] *ibid*, p. 4.

of the important aspects (which are explained in the previous sections) considering security and cryptography are implemented and used in a cryptosystem.

Cryptographic systems or cryptosystems, namely, must provide all three objectives of information security: confidentiality, integrity, and availability. Confidentiality simply means concealment of data from unauthorized parties and assurance of secrecy and hiding of information. Integrity means that data can be modified only by authorized parties and assurance of impossibility of any kind of modification otherwise. Availability means that the system still functions efficiently after security precautions or measures are in place; also, the assurance of not preventing authorized parties from accessing data or system objects that they already have legitimate access.[22]-[23] In addition, confidentiality and integrity can also be sub-classified into five groups: confidentiality, user authentication (assurance that any party is really himself / herself as identified), data origin authentication (assurance of the source of a message), data integrity, non-repudiation (the receiver of a transaction is able to demonstrate to a third party that the claimed sender did indeed send the transaction).[24]

As far as it' s known, there are two basic types of cryptosystems; Symmetric and Asymmetric cryptosystems, namely. Sometimes, they can be referred as algorithms, considering key-based algorithms; Symmetric and Asymmetric (public-key) algorithms.[25] Since both of them are implemented with the usage of key(s) and since the distinction between them lies behind the usage and structure of key(s), thus the algorithms, sometimes these cryptosystems can be called as algorithms.

2.3.2 Symmetric Cryptosystems

In key-based cryptosystems, if the key used for encryption and decryption is the same then this is called as symmetric cryptosystem. The encryption key can be deduced from the decryption key or vice versa; and since there's a single key for both encryption and decryption process, these systems are also referred as secret-key, single-key, one-key algorithms or cryptosystems.[26] Also since there' s a single key, this must be in hands of both the sender and the receiver which brings the necessity of; secure key transfer, agreement protocols for the key usage and transfer, authentication and proof of key validness, etc.[27]-[28]-[29] In applications where a limited number of users exist, symmetric key cryptosystem can be considered feasible. However, in large networks with a lot of users distributed over a wide area, key distribution becomes a problem in symmetric cryptosystems. Each individual in a network should have a unique and different key to communicate with each other person. To establish this, a tremendous number of keys must be produced and stored securely. For instance, a

---

[22] Certicom Inc., "An Introduction to Information Security", Certicom Whitepaper, p. 2, March 1997.
[23] Charles P. Pfleeger, *Security in Computing*, pp. 4-6.
[24] Certicom Inc., "An Introduction to Information Security", Certicom Whitepaper, p. 3, March 1997.
[25] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 4.
[26] *ibid*, p. 4.
[27] *ibid*, pp. 4, 21-31, 47-74, 216-220.
[28] Aviel D. Rubin, Peter Honeyman, "Formal Methods for the Analysis of Authentication Protocols", Technical Report, Center for Information Technology Integration, pp. 1-17, 1993.
[29] Charles P. Pfleeger, *Security in Computing*, pp. 129-164, 381-385.

system with 1000 users would require approximately 500,000 keys to be exchanged and maintained securely; in addition, exchanging and managing such a large number of keys is very costly and difficult causing too much overhead, and sometimes might be impossible at all.[30]

However, symmetric cryptosystems require less computational effort and time than asymmetric cryptosystems considering encryption and decryption. Also, the key sizes used are comparatively much smaller than asymmetric cryptosystems while achieving equal levels of security. Symmetric cryptosystems are commonly accepted and used when speed is an important criterion in the security implementation. They are widely used all around the world and accepted more conventional than asymmetric cipher models. Symmetric algorithms and cryptosystems are considered more efficient than asymmetric ones (not thoroughly, but by most of the authorities and security experts) and heavily used in applications of encrypting data (computer files, mail, passwords, credit and bank cards, documents, software, etc.).[31]

When someone wishes to send a secret message to another user, he / she encrypts the message with a key K and a symmetric encryption algorithm used with that key. Then the encrypted secret message or data is sent to the other side. Also, the receiver has the same key K and the decryption algorithm which enables him / her to decrypt the message and get the plaintext. Both sides use the same private key K which must be agreed on and be known to no one but the sender / receiver couple as well as the cryptographic protocols; and this must be established in a very trusted and secret manner before each session begins.

Encryption & decryption in a symmetric algorithm is denoted as below and also shown in Figure 2.1.

$$\textit{Encryption:}\ E_k(P) = C$$
$$\textit{Decryption:}\ D_k(C) = P$$

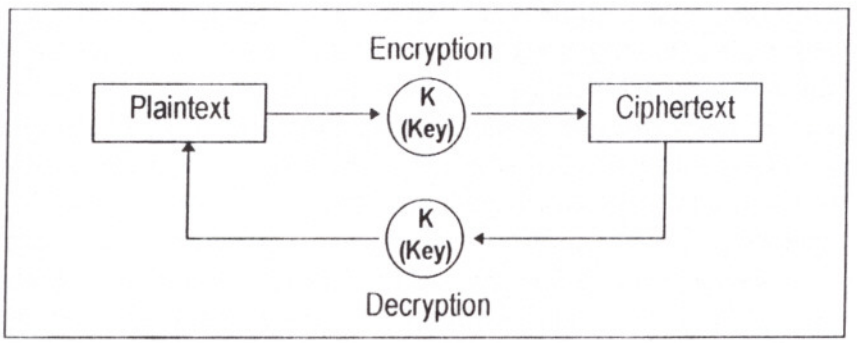


Figure 2.1 Encryption & Decryption in a Symmetric Cryptosystem.

[30] Certicom Inc., "An Introduction to Information Security", Certicom Whitepaper, p. 4, March 1997.
[31] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 216.

In general, symmetric cryptosystems are divided into two categories: Block Ciphers (algorithms) and Stream Ciphers (algorithms).[32] If each time a single bit or byte of a plaintext (sometimes one 32-bit word)[33] is processed and encrypted, then this is referred as stream ciphers. In stream ciphers, one character of plaintext is immediately converted into ciphertext where the conversion or transformation mechanism depends only on the character (bit or byte), the key, and control information of the encryption algorithm.[34]

On the other hand, if a group of characters (bits or bytes) is operated and encrypted at a time, it is called as block ciphers or block algorithms. The group of bits or bytes is named as blocks where the block size used in a cryptosystem need not have any particular relationship with the size of a character.[35] It should be noted that the block size can be of any fixed value in an encryption mechanism, as well as variable-sized blocks can be used. For today' s computer algorithms, blocks of 64-bit length are commonly used and considered as conventional and compact enough to be workable;[36] but other sizes are also valid.

Some symmetric cryptosystems are designed to operate in stream modes, others in block; but also some algorithms can be accomplished to process in both modes alternatively as seen in DES. (These alternative operation modes will be explained in section 2.4.2.) But it should be stressed that DES and some other well known symmetric cryptosystems are usually operated in block cipher modes; thus considering differential and linear cryptanalysis, block algorithms are concerned in this study. In general, whatever the operation mode of a symmetric cipher, it can be concluded that the basic theory and implementation is based on substitution / transposition operations (shift registers, inverters, etc.), special function operators (ie. S-Boxes), expansion operators and the key.

Regarding security considerations block ciphers have some advantages over stream ciphers; block ciphers are more difficult to break than stream ciphers due to diffusion and they are immune to insertions.[37] But on the other hand, they are slower during encryption when compared to stream ciphers and there' s the problem of error propagation; even an error in one bit or byte of a block will affect all the subsequent characters in that block. Considering stream ciphers, they are faster in encryption than block modes and there' s low error propagation; an error in a bit or byte doesn' t affect the other bits or bytes due to the immediate detection of error as a result of singularity in transformation. By contrast, stream ciphers do have some disadvantages; they are more vulnerable because of low diffusion and they are prone to malicious insertions or modifications because of the separate encryption of each character[38], a case surely not seen in block ciphers.

---

[32] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 4.
[33] *ibid*, p. 189.
[34] Charles P. Pfleeger, *Security in Computing*, p. 60.
[35] *ibid*, p. 61.
[36] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 4.
[37] Charles P. Pfleeger, *Security in Computing*, p. 63.
[38] *ibid*, pp. 60-61.

The definition summarizing the basic difference between stream and block ciphers is given by *Rainer Rueppel*[39] stating that;

Block ciphers operate on data with a fixed transformation on large blocks of plaintext data; stream ciphers operate with a time-varying transformation on individual plaintext digits.

It' s also accepted that in the real world applications, block ciphers seem to be more general (more widely accepted and has four different alternative modes) and stream ciphers yield to be analyzed more easily in mathematical terms. It' s usually accepted and also proven that stream ciphers do have a lower linear complexity, thus less secure than block ciphers.[40] Also stream ciphers are vulnerable to correlation attacks.[41] On the other hand, differential and linear cryptanalysis methods are especially generated for block ciphers and very successful attacks have been made to some of the block symmetric cryptosystems, but not to all.[42] It should be noted that a block cipher's vulnerability to cryptanalytic attacks or its strength lies behind the design of the S-Boxes, diffusion / confusion quality in the functions producing the cipher, key length, existence of weak keys, possession of the group structure.[43] Rather than being theoretical, when applications in real world is considered the basic difference between stream and block ciphers is concerned to be the nature of the implementation. Due to their design and algorithmic structure, block ciphers are easier to implement in software; whereas stream ciphers are considered to be more suitable for hardware implementations.[44]

There are various types of symmetric ciphers developed early and some of them are still in use today. Most well known ones are DES and variants of DES, FEAL, LOKI, GOST, CAST, FEISTEL, RC4, RC5, Khufu, Khafre, IDEA, SAFER, Blowfish, SEAL, REDOC, WAKE.[45]

2.4 DES

Since this research is involved in cryptanalysis of symmetric ciphers, and since DES is today' s one of the most well known and commonly used symmetric cryptosystem which also stands as a basic model for the other symmetric ciphers, it will be focused on in more detail throughout this section. This is also necessary due to the fact that DES was chosen as the target system in most of the linear and differential cryptanalytic studies and attacks that have been carried out, thus it would be better for the readers to have a pre-knowledge about DES before going into further details of linear / differential cryptanalysis.

---

[39] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 210.
[40] *ibid*, p. 380.
[41] *ibid*, p. 380.
[42] *ibid*, pp. 346-349.
[43] *ibid*, pp. 346-351.
[44] *ibid*, p. 211.
[45] *ibid*, pp. 265-351, 397-404.

2.4.1 Definition & Background

The Data Encryption Standard, namely DES, is a cryptosystem and algorithm developed for the U.S. government previously and has been in common use all around the world afterwards. DES has been officially accepted as a cryptographic standard both in the U.S. and abroad which many hardware and software systems have been implemented as well as the new and improved versions of the algorithm such as Triple-DES, Fenced-DES, etc. However, as a controversy its security and reliability has been questioned so far. Considering several successful cryptanalytic attacks and discovery of some security flaws; at least the standard 56-bit DES and other old versions are doubted today and not trusted for the future use.[46]-[47]-[48]

In the late 1960' s IBM began a research project in computer cryptography which concluded in 1971 with the production of a symmetric block cipher named as LUCIFER. Meanwhile, because of the great demands and needs, the National Bureau of Standards (NBS), which is today known as the National Institute of Standards and Technology (NIST), was trying to construct the standards for encryption systems, computer and communication security. In 1973, NBS defined the criteria for such encryption systems and issued a call for a single public encryption algorithm which could satisfy all the criteria proposed, which could be tested and publicly certified, and which could become as a standard. The specified criteria were (which are also valid today) as follows:

- The algorithm must provide a high level of security.
- It must be completely specified and easy to understand.
- The security of the algorithm must reside in the key, thus the algorithm itself must provide the security; the security should not depend on the secrecy of the algorithm like a black box.
- The algorithm must be available to all users.
- It must be adaptable for use in diverse applications.
- It must be economical whenever implementation in electronic devices is necessary.
- It must be efficient to use.
- It must be open to validation.
- It must be exportable.

After the second call from NBS in 1974, IBM started a new project for the proposed algorithm which is based on formerly developed Lucifer. Later on, IBM announced the new algorithm named as DEA (Data Encryption Algorithm). After the negotiations between NBS and IBM, and after some changes, tests and adaptations, NBS approved the algorithm as a standard and released it for public security usage renowned as DES. It must be noted that until 1994 DES was only validated and formally accepted as a standard for hardware and firmware implementations; after that NIST also validated its software implementations.[49]-[50]

---

[46] Bruce Schneier, *Applied Cryptography - Second Edition*, pp. 265-270.
[47] Charles P. Pfleeger, *Security in Computing*, p. 106.
[48] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, pp. 1-3, 7-9.
[49] Bruce Schneier, *Applied Cryptography - Second Edition*, pp. 265-270.
[50] Charles P. Pfleeger, *Security in Computing*, pp. 106-107.

2.4.2 Algorithm and Basic Structure

DES is a symmetric block cipher and algorithm which is based on the combination of the two most fundamental and typical methods of encryption: substitution and permutation (transposition). The algorithm gets its strength from repeated application of these two techniques as a total of 16 cycles (rounds) and the usage of a 64-bit key which is decomposed into subkeys through each of these cycles so that during each cycle, encryption is achieved with a new key value. Although the plaintext is encrypted in 64-bit blocks and key length is known to be 64 bits, in fact the key is 56 bits long where 8 bits are discarded for parity checking. Since DES is classified as a symmetric cryptosystem, the same algorithm and the same key is used both in encryption and decryption except some slight differences in the key schedule.[51]-[52] The substitutions in DES provide confusion by substituting some bit patterns for others in each block of data. On the other hand, the permutations provide diffusion by reordering the bits. Thus, the plaintext is operated on a substitution and then a transposition through each round in an iterative manner which results with an encrypted 64-bit block after 16 rounds. It must be mentioned here that confusion means output bits (ciphertext) have no obvious relationship to the input bits (plaintext), no or very little correlation should be observed between them. Diffusion means any change in a single bit of plaintext should affect many parts or bits of the ciphertext, in other words, diffusion is the characteristic of distributing the information from single plaintext letters over the entire output.[53]

The encryption process in DES algorithm is done as follows. The first 64-bit portion of the plaintext is taken as input and each time, the next 64-bit block is processed for encryption until the end of plaintext is reached. If the last block of plaintext falls behind the 64-bit length, the algorithm pads the empty block with the key. Also, the 64-bit key chosen previously is shortened to 56 bits by dropping the 8^{th}, 16^{th}, 24^{th}, ... 64^{th}. An initial permutation (IP) is applied on the 64-bit plaintext and then this block is divided into a left half and a right half, each 32 bits in length. Then some mixed operations named as function f is applied on the right and left halves interchangeably for each of the 16 rounds where the data is combined with the key. After the 16^{th} cycle, the right and left halves are joined and a final permutation which is known as inverse of the initial permutation (IP^{-1}) is applied. This is the last stage of the algorithm where the same processes are executed for each of the 64-bit plaintext blocks.[54]-[55] The whole DES encryption process is also demonstrated in the Figure 2.2.

In each of the 16 rounds, the key bits are shifted and then 48 bits are selected from 56 bits. Then these shifted bits are permuted by a simple permutation operation. The right half of the data is expanded from 32 to 48 bits by a special operation known as expansion operation. These 48 bits of data is XORed (denoted by $\oplus$) with the shifted and permuted choice of 48-bit key. The output is processed through eight S-Boxes and 32 new bits are produced. These 32 bits are finally permuted with a P-Box. These four operations compose the function f. The output from P-Box is XORed with

---

[51] Charles P. Pfleeger, *Security in Computing*, p. 107.
[52] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 270.
[53] Charles P. Pfleeger, *Security in Computing*, pp. 64-66, 109.
[54] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 270.
[55] Charles P. Pfleeger, *Security in Computing*, p. 109.

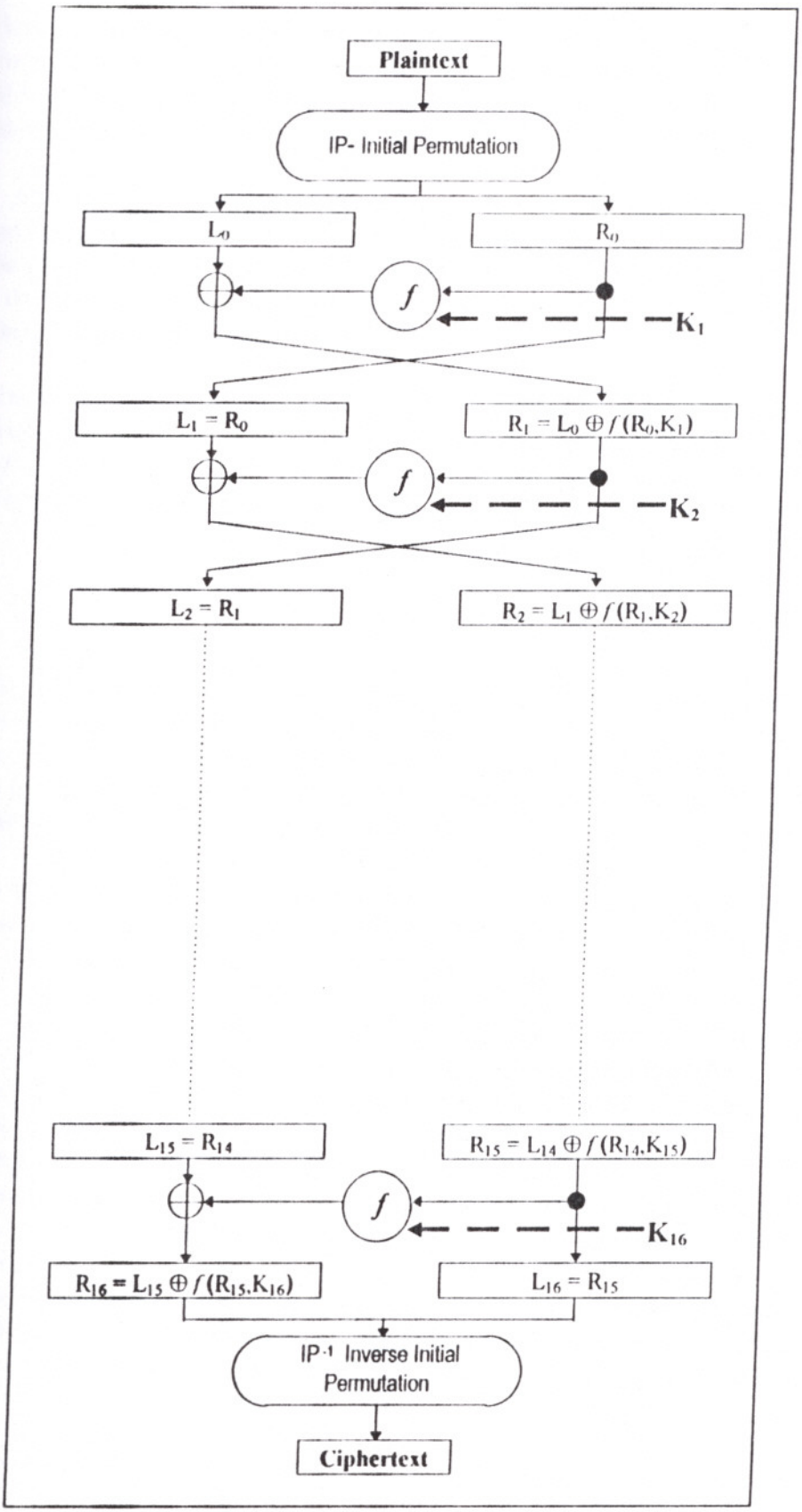


Figure 2.2 Complete representation of a 16-round DES Algorithm.[56]

[56] Bruce Schneier, ***Applied Cryptography*** - *Second Edition*, p. 271.

the left half which makes the new right half input for the next round where the old right half becomes the new left right for the next round. All of these operations can also be analyzed from Figure 2.3. This operation is repeated 16 times which makes up the 16-round standard DES.[57-58-59]

After the 16th cycle, the left and right halves are combined together and processed through the inverse initial permutation to produce the 64-bit cipher block. It should be noted that the left and right halves are not exchanged after the 16th round as seen in the previous 15 rounds; instead the concatenated left and right blocks are directly input to the final permutation IP^{-1}.[60]

The operation performed in each round can also be denoted as below, where K_i is abbreviated for the 48-bit key in round i, L_i and R_i are the left and right halves of round i, L_{i-1} and R_{i-1} are the left and right halves from the previous round and f is the function executed in each cycle;

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

The initial permutation IP transposes the input block by changing the positions of the bits which is shown in Table 2.1. For instance, after the initial permutation the 58th bit moves to the bit position 1, bit 34 moves to position 4, bit 7 moves to 64, etc. Similarly, IP^{-1} does just the same operation inversely in order to get back the bits to their initial positions. The initial permutation and corresponding final permutation do not affect DES' s security, even these operations are ignored in cryptanalytic attacks since they do not have any cryptanalytic significance.[61] Most people suggest that their primary purpose is to make it less complex and less difficult to load plaintext and ciphertext data into a DES chip in byte-sized pieces.[62]

As previously mentioned, the initial DES key is first reduced from 64 to 56 bits ignoring every eighth bit so as to use as parity bit. While discarding these bits, the positions of the key bits are also interchanged and this operation is named as key permutation or key transformation which is given in Table 2.2. After this process, a different 48-bit subkey is generated for each of the 16 cycles. In other words, the keys K_i used in each round i are also called as the subkeys since they are all calculated from the 56-bit key at the very beginning. After the key scheduling operation, a new subkey value is derived for that round.[63] This key scheduling operation is composed of two processes; key shifting and compression permutation. First, the 56-bit key is broken up into two 28-bit halves. Then, these halves are independently and circularly shifted left by 1 or 2 bits, depending on the round. This shifting operation is also given in Table 2.3.

---

[57] Bruce Schneier, *Applied Cryptography - Second Edition*, pp. 110-111.
[58] *ibid*, p. 270.
[59] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, pp. 12-14, 155-156.
[60] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 277.
[61] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, p. 12.
[62] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 271.
[63] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, p. 12.

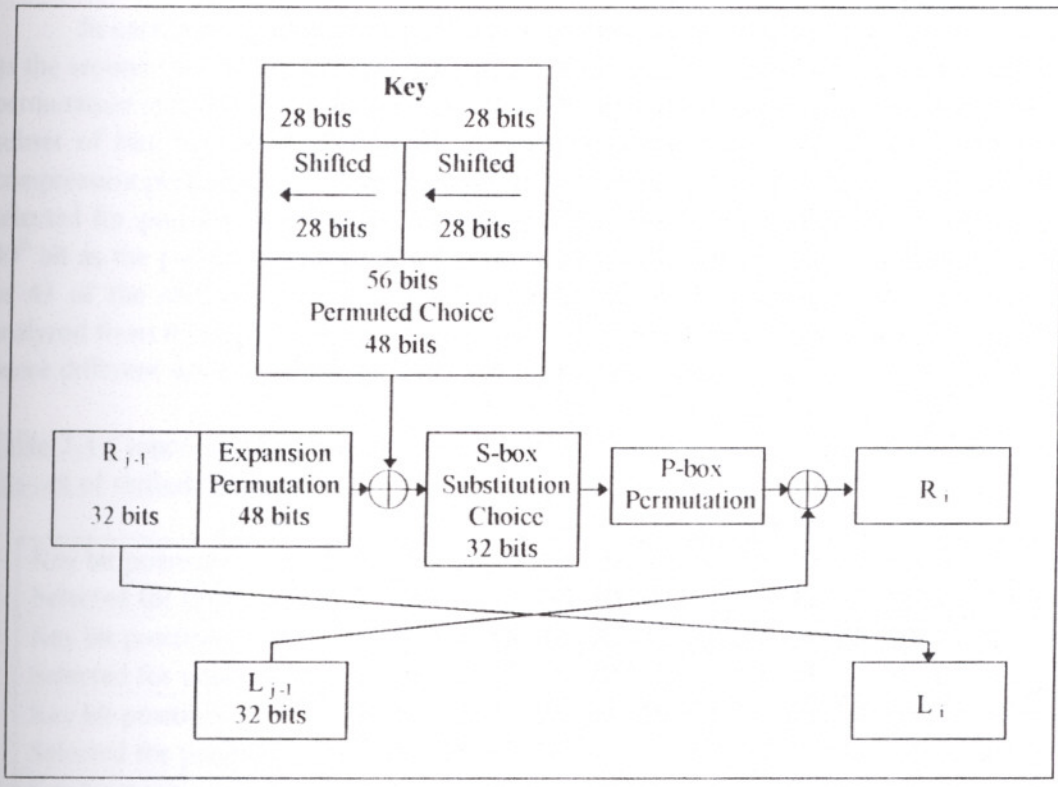


Figure 2.3 One Round of DES with 56-bit key.[64]

It should be noted that, in the Figure 2.3, the compression permutation operation is denoted as 56 bits → Permuted Choice → 48 bits.

Table 2.1 Initial Permutation.[65]

58, 50, 42, 34, 26, 18, 10, 2, 60, 52, 44, 36, 28, 20, 12, 4, 62, 54, 46, 38, 30, 22, 14, 6, 64, 56, 48, 40, 32, 24, 16, 8, 57, 49, 41, 33, 25, 17, 9, 1, 59, 51, 43, 35, 27, 19, 11, 3, 61, 53, 45, 37, 29, 21, 13, 5, 63, 55, 47, 39, 31, 23, 15, 7

Table 2.2 Key Permutation.[66]

57, 49, 41, 33, 25, 17, 9, 1, 58, 50, 42, 34, 26, 18, 10, 2, 59, 51, 43, 35, 27, 19, 11, 3, 60, 52, 44, 36, 63, 55, 47, 39, 31, 23, 15, 7, 62, 54, 46, 38, 30, 22, 14, 6, 61, 53, 45, 37, 29, 21, 13, 5, 28, 20, 12, 4

Table 2.3 Number of Key Bits shifted in each Round.[67]

Round	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Number of Bits Shifted	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

[64] Charles P. Pfleeger, *Security in Computing*, p. 111.
[65] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 272.
[66] *ibid*, p. 273.
[67] *ibid*, p. 273.

In each round, after shifting the bits another permutation process is carried out as the second part of the key transformation. 48 out of the 56 bits are selected and the permutation operation is processed on these 48 bits named as permuted choice. Since a subset of bits are selected as well as permuting the order, this is also named as compression permutation.[68] For example; the bit in the position 1 of the shifted key is selected for position 5 as output, or similarly 56th bit of the shifted key becomes the 40th bit as the permuted choice. On the other hand, the bits in positions such as 9, 38, or 43 of the shifted key are not selected for the 48-bit output. This can also be analyzed from the Table 2.4. Due to the shifting operation, different subset of key bits, hence different subkey values used in each of the 16 rounds.

Table 2.4 Compression Permutation (Permuted Choice which selects the Subkeys' 48 bits out of shifted 56 bits).[69]

Key bit position	1	2	3	4	5	6	7	8	**9**	10	11	12	13	14
Selected for position	5	24	7	16	6	10	20	18		12	3	15	23	1
Key bit position	15	16	17	**18**	19	20	21	**22**	23	24	**25**	26	27	28
Selected for position	9	19	2		14	22	11		13	4		17	21	8
Key bit position	29	30	31	32	33	34	**35**	36	37	**38**	39	40	41	42
Selected for position	47	31	27	48	35	41		46	28		39	32	25	44
Key bit position	**43**	44	45	46	47	48	49	50	51	52	53	**54**	55	56
Selected for position		37	34	43	29	36	38	45	33	26	42		30	40

For each round, the right half of the data, R_i, is expanded from 32 to 48 bits by an operation known as expansion permutation. Not only the order of the bits are changed in order to perform transformation, but also some chosen bits are repeatedly used so as to achieve expansion. For instance, the 3rd bit in coming from the right half as the input becomes the 4th bit in the output block; the bit in position 1 in the input block becomes the 2nd and 48th bits in the output. These can be checked from Table 2.5.

Expansion permutation is designed and used in DES for two basic purposes: First one is to make the right half size equivalent to the key for the XOR operation, and the second is to provide a longer result that can be compressed later during the substitution.[70]-[71] However, these are technical considerations which have no relation with security enhancements or cryptographic purposes. But it does have also an aim for cryptographic improvement. By the expansion permutation, one bit is enabled to affect two substitutions, thus the dependency of the output bits on the input bits spreads faster. This is known as the avalanche effect which is a desirable property of any encryption algorithm. DES exhibits this property so that every bit of the ciphertext depend on every bit of the plaintext and the key which provides the condition that a small change either in the key or the plaintext does produce a great change in the ciphertext, in other words, high diffusion. Expansion permutation enables the

[68] Bruce Schneier, *Applied Cryptography - Second Edition*, pp. 272-273.
[69] Charles P. Pfleeger, *Security in Computing*, p. 113.
[70] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 273.
[71] Charles P. Pfleeger, *Security in Computing*, p. 111.

avalanche effect to occur very rapidly.[72] During this research, some work has been carried out related with the avalanche effect and it' s given in Appendix B.

Table 2.5 Expansion Permutation.[73]

32,	1,	2,	3,	4,	5,	4,	5,	6,	7,	8,	9,
8,	9,	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,	1

After the compressed key is XORed with the expanded right half block, the 48-bit output goes through a substitution operation. All the substitutions in this stage are carried out by eight substitution boxes, named as S-boxes. In fact, it' s assumed that the S-boxes are the core of the DES algorithm, their uniqueness in design, high confusion performance and their non-linearity give DES its cryptographic strength and make DES secure.[74]-[75] The S-boxes operate as follows: Each of the eight S-boxes has a 6-bit input and 4-bit output where each S-box is different from the others. The 48-bit input block is divided into eight sub-blocks of 6 bits in size. Each of these 6-bit sub-blocks is input to a separate S-box such that the first sub-block goes to S-box 1, the second is input to S-box 2, and so on. In other words, the first 6 bits of the 48-bit block is processed through S-box 1, the bits from 7 to 12 is input to S-box 2, etc. After the substitution operation through S-boxes, each of the eight S-boxes produce an output of 4 bits, totally making 32 bits. This 32-bit output is then forwarded to the P-Box for permutation. These can also be checked out from Figure 2.4.

The process inside the S-boxes and their structure can be described simply here. Each S-box is a table made up of 4 rows and 16 columns. Each entry in the box is a 4-bit number ranging from 0 to 15. In Table 2.6, eight S-boxes with all their output entries are given.[76] Since there are 64 entries in each box, some numbers are used more than once in the S-box. The 6-bit input value give the row and column value of the entry for which will be chosen as the 4-bit output value. For example, if the 6 input bits to an S-box are denoted as $b_1\ b_2\ b_3\ b_4\ b_5\ b_6$, then the left-most and right-most bits, b_1 and b_6 respectively, are combined to form a 2-bit number which is used to index the rows (0 to 3) in S-box. Similarly, the other four bits in the middle, $b_2\ b_3\ b_4\ b_5$ are combined to form the 4-bit number so as to index the columns ranging from 0 to 15. This can also be described by the following example:

If the input to the S-box 1 is 101110, then this input implies the entry which is placed in the 2nd row (10) and 7th column (0111), and for S-box 1, this is the value 11 (in decimal), hence the 4-bit output will be 1011.

---

[72] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 273.
[73] *ibid*, p. 275.
[74] *ibid*, p. 275.
[75] Charles P. Pfleeger, *Security in Computing*, p. 117.
[76] *ibid*, p. 115.

Table 2.6 S-Boxes for DES.

S-Box 1

Row	Column 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S-Box 2

Row	Column 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	12	15
3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S-Box 3

Row	Column 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S-Box 4

Row	Column 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S-Box 5

Row	Column 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S-Box 6

Row	Column 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S-Box 7

Row	Column 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S-Box 8

Row	Column 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

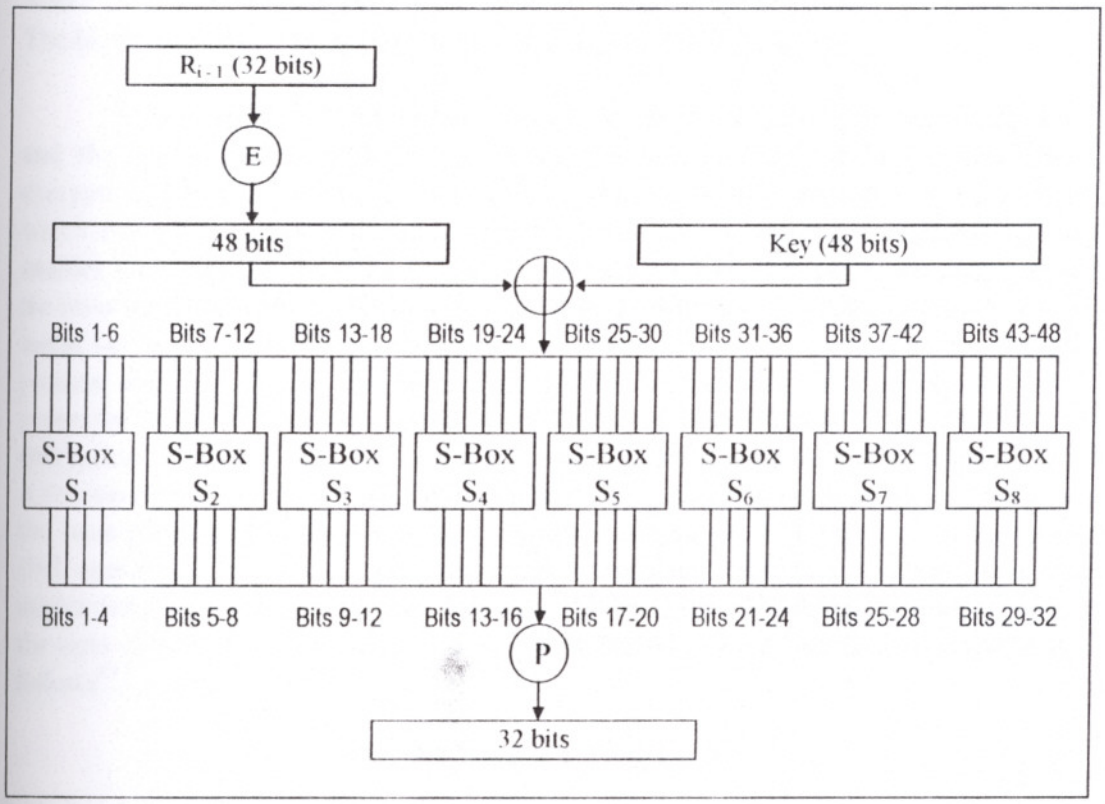


Figure 2.4 S-Boxes from S_1 to S_8 where each one operating on 6-bit input blocks and producing 4-bit output blocks.[77]-[78]

After the S-box substitution, the 32-bit output is processed through a permutation box known as P-box. The operation is also named as straight permutation[79] because there' s no bit discarding or bit expansion as seen in the previous permutation operations. In P-box permutation, each input bit is mapped to an output bit position producing a total of 32-bit output from 32 bits of input. For example, bit 16 of the input moves to position 1 in the output, while bit 1 of the input block moves to 9th position in the output block. The permutation process inside the P-Box is simply demonstrated in Table 2.7.

Table 2.7 P-Box Permutation.[80]

16, 7, 20, 21, 29, 12, 28, 17, 1, 15, 23, 26, 5, 18, 31, 10, 2, 8, 24, 14, 32, 27, 3, 9, 19, 13, 30, 6, 22, 11, 4, 25

As a final operation, the 32-bit output of the P-box is XORed with the left half of the initial 64-bit block. The output 64-bit block becomes the new right half for the next round where the initial 64-bit right half block is switched to be the new left half for the next round. Thus, the next round begins with these new right and left blocks

[77] Charles P. Pfleeger, *Security in Computing*, p. 114.
[78] William Stallings, *Network and Internetwork Security - Principles and Practice*, p. 47.
[79] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 275.
[80] *ibid*, p.277.

where all of these operations are repeated 16 times as the 16 rounds or cycles of DES. The block exchange can also be observed from the Figure 2.2.

As well as encryption, the decryption in standard DES is achieved in 16 rounds and the operations throughout the decryption process are similar to the ones in encryption, but in an inverse manner. This is due to the symmetrical structure of DES which enables the same algorithm work both for encryption and decryption and also enables the usage of the same key, thus the subkeys.[81]-[82] But since, the decryption is the inverse of encryption, this time the ciphertext 64-bit block is taken as input, and the initial key is taken as K_{16}. Thus, using K_{16} in the first cycle, and repeating the same process with key K_{15} in the second round, K_{14} in the third, and so on, the similar processes are iterated for 16 rounds with the same algorithms and operations seen in encryption. Finally, K_1 is used in the 16th round and the original 64-bit plaintext block is recovered aftermath. All the functions and the operations are exactly the same with the ones processed in the encryption except some slight differences such as the key shift operation. The basic logic while generating each subkey is the same and again implemented in a circular manner, however this time the shift in each round is made in the right direction and the number of key bits shifted in each round of decryption is as follows[83];

0 1 2 2 2 2 2 2 1 2 2 2 2 2 2 1

It can also be compared with the shift process during the encryption procedure which was given in Table 2.3. It should be noted that in the decryption process, the same initial permutation and its inverse, IP and IP^{-1} respectively, are used which were also used in the encryption algorithm. As seen in the encryption, the left and right 32-bit halves L_i and R_i are exchanged after each round i.

In order to explain it better and in a simple manner, the decryption process is formulated as below which can also be considered as the theoretical proof showing that the decryption process is the inverse of the encryption. In fact, this property is based on the symmetrical structure of the functions and operations used and the special nature of XOR operation.

From the 16th round of encryption it' s known that;

$$L_{16} = R_{15}$$
$$R_{16} = L_{15} \oplus f(R_{15}, K_{16})$$

Then, by the property of XOR, it can be deduced that;

$$L_{15} = R_{16} \oplus f(R_{15}, K_{16})$$

[81] Bruce Schneier, *Applied Cryptography - Second Edition*, p.277.
[82] Charles P. Pfleeger, *Security in Computing*, p. 116.
[83] *ibid*, p. 277.

Thus,

$$L_{15} = R_{16} \oplus f\left(L_{16}, K_{16}\right).$$

So, since L_{16} , K_{16} and R_{16} already in hand, for the beginning of 1st round of decryption, L_{15} and R_{15} can be produced easily as well as the subkey K_{15} (shifting K_{16}). Hence, in a similar manner, L_{14} and R_{14} can be obtained in the 2nd round, and this can be iterated for 16 rounds until L_0 and R_0 are obtained. Thus, after the IP^{-1} filter the plaintext is retrieved. These can be summarized in a generic form for each and every round as follows, where $j = 17\text{-}i$ and i stands for the i[th] round of decryption (conversely, j was the j^{th} round of encryption);

$$L_{j-1} = R_j \oplus f\left(L_j, K_j\right)$$
$$R_{j-1} = L_j$$
$$K_{j-1} = Shift\left(K_j\right)$$

2.4.3 Operation Modes of DES

For the ease of use, and for the security enhancement in applications and implementations, any one of the four common operation modes is used within the DES algorithm. In fact, these four modes of operation are also used in some other block or stream cipher symmetric encryption algorithms. They are namely; ECB - Electronic Codebook Mode, CBC - Cipher Block Chaining Mode, CFB - Cipher Feedback Mode, OFB - Output Feedback Mode. Among those, ECB and CBC are designed and used for block ciphers, whereas CFB and OFB are stream cipher modes in nature, but they can also be adapted and used for block cipher algorithms so as to make those encryption algorithms process in streams of bits or bytes.[84]

Because of its simplicity, ECB is mostly used in off-the-shelf commercial software products as well as in the studies throughout this thesis, but it' s not recommended for highly-confidential purposes since ECB is the most vulnerable mode to cryptanalytic attacks[85] and provides the lowest avalanche effect amongst all the modes. The *ANSI banking standards* specify ECB and CBC for encryption, CBC and n-bit CFB for authentication.[86] Some security specialists suggest that ECB is only suitable for secure transmission and encryption of small amounts of data (ie. a key); CBC is convenient for block ciphers and secure encryption of long sized data; CFB would rather be chosen for authentication and also general-purpose stream-oriented data transmission; OFB is best suited for stream-oriented data transmission over noisy channels such as satellite communication.[87]

Besides the four modes used in DES, there are other operation modes for stream and block ciphers, such as Counter Mode, Block Chaining Mode, Plaintext

---

[84] Bruce Schneier, *Applied Cryptography - Second Edition*, pp. 189-205.
[85] *ibid*, pp. 277-278.
[86] *ibid*, p. 277.
[87] William Stallings, *Network and Internetwork Security - Principles and Practice*, p. 58.

Block Chaining, Plaintext Feedback Mode, Cipher Block Chaining with checksum, etc.[88] These types of modes will not be discussed since they are not involved in this study, but the four modes applied to DES will be taken into spotlight.

- **Electronic Codebook Mode - ECB**

The simplest mode among all the modes is proven to be the electronic codebook mode where plaintext is divided into 64-bit blocks and is encrypted in 64-bit blocks at a time where each block of plaintext is encrypted using the same initial key. Thus, there' s a unique ciphertext block for each corresponding plaintext block. This is the easiest mode in implementation and usage both for software and hardware platforms. Since each block is encrypted independently, there' s no necessity of encrypting a file or data linearly or in any definite sequence. In other words, during encryption (and also decryption), no plaintext block is dependent on the previous plaintext or ciphertext blocks. This is especially useful in encrypting random-access files, databases and in parallel implementations with multiple processors (all the processors can encrypt or decrypt the blocks of the whole data independent of each other and without considering the order).[89]

Decryption in ECB is processed in the same way as encryption where each block of ciphertext decrypted at a time independently using the same key. The encryption and decryption with DES in ECB mode is shown in Figure 2.5.

It's stressed that any bit errors in the ciphertext block will cause the corresponding plaintext block to be decrypted incorrectly but this is accepted as an advantage for ECB mode, since errors in the ciphertext block only affect that corresponding plaintext block so that the other plaintext blocks can be decrypted correctly and the rest of the plaintext data can be obtained without errors. However, if a bit or some bits are lost or added in the ciphertext block, this will cause all the succeeding plaintext blocks to be decrypted incorrectly as well as that corresponding plaintext block[90] (ie. if the first bit of the ciphertext is removed as a result of damage, then since the size of the ciphertext is changed, all the ciphertext / plaintext blocks will be arranged incorrectly during decryption and all the plaintext data retrieved starting from the first bit will be wrong). This is considered as a disadvantage for ECB.

As mentioned before, the main disadvantage of ECB is its low diffusion and vulnerability to some cryptanalytic attacks. Since there's a unique ciphertext for each plaintext under the same key, a cryptanalyst can generate a codebook with several plaintext / ciphertext pairs. Thus, for the incoming encrypted messages, he / she can deduce the plaintext block whenever a ciphertext block which also exists in the codebook is met. With the help of some statistical analyses, all of the original plaintext message can be recovered and if necessary, the key can also be retrieved. Because of these shortcomings, ECB is not accepted as a secure method for lengthy messages and is only used for encrypting small amounts of data, eg. a key.[91]

---

[88] Bruce Schneier, *Applied Cryptography - Second Edition*, pp. 205-208.
[89] *ibid*, p. 190.
[90] *ibid*, p. 190.
[91] William Stallings, *Network and Internetwork Security - Principles and Practice*, p. 59.

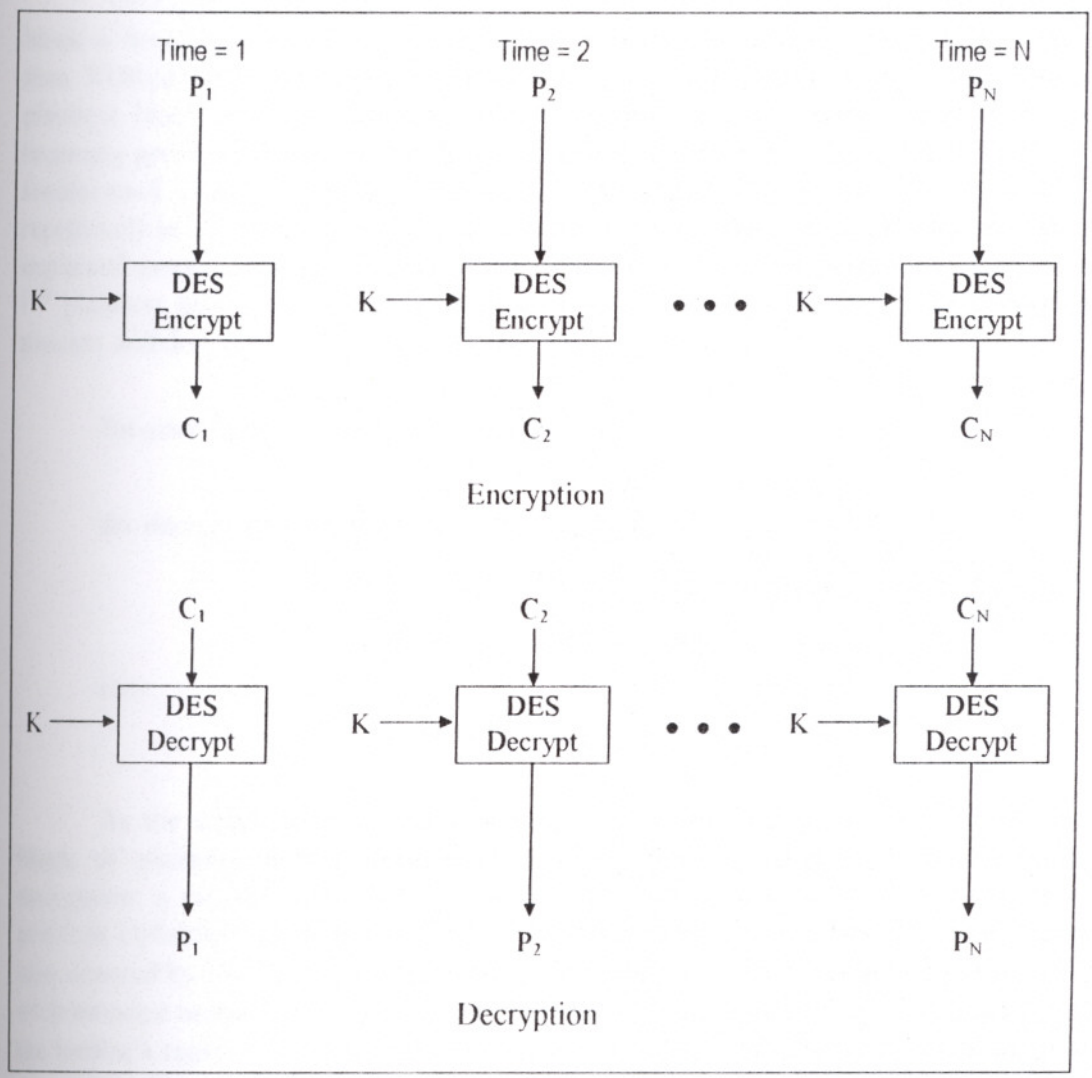


Figure 2.5 Electronic Codebook Mode.[92]

- **Cipher Block Chaining Mode - CBC**

CBC mode was designed to overcome the security flaws faced in ECB and to increase the diffusion rate and avalanche effect in the ciphertext data. The basic goal was to implement a cryptographic mode that would enable the production of different ciphertext blocks from the same plaintext block whenever met more than once in the same data input. This goal was achieved by using a simple feedback mechanism known as chaining. The scheme in CBC can be described simply as follows; each time, the plaintext block is first XORed with the previous ciphertext block that is stored in a feedback register and the result is encrypted with the key afterwards. This is processed repetitively until the end of input data. Thus, each ciphertext block is not only dependent on the corresponding plaintext block but also on all of the other previous plaintext blocks.[93] CBC mode algorithm also works on input data in definite sizes of block portions.

[92] William Stallings, *Network and Internetwork Security - Principles and Practice*, p. 59.
[93] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 193.

Decryption process is also proven to be straightforward where each ciphertext block is decrypted with the same key used in encryption and the decrypted output is then XORed with the previous ciphertext block. The result is the corresponding plaintext block for that ciphertext block. Again, for each block's decryption, the incoming previous ciphertext block is stored in the feedback register (also referred as feedforward register for the decryption stage) and the process is carried out repetitively in sequence until the last ciphertext block's decryption. These can also be explained simply using formulas as below; where C_i is the i th ciphertext block, P_i is the i th plaintext block, E_K is the encryption process with key K, D_K is the decryption process with key K.

$$\text{for encryption}: \quad C_i = E_K\left(C_{i-1} \oplus P_i\right)$$

$$\text{for decryption}: \quad D_K\left(C_i\right) = D_K\left(E_K\left(C_{i-1} \oplus P_i\right)\right)$$

$$D_K\left(C_i\right) = C_{i-1} \oplus P_i \qquad \text{since,} \quad D_K\left(E_K\left(f(x)\right)\right) = f(x)$$

$$C_{i-1} \oplus D_K\left(C_i\right) = C_{i-1} \oplus C_{i-1} \oplus P_i$$

thus,

$$P_i = C_{i-1} \oplus D_K\left(C_i\right)$$

As the whole scheme looks simple, it' s worth to mention that for the first block of plaintext during encryption and for the first ciphertext block during decryption, a random data block is put into the feedback register since there' s no previous ciphertext block in hand. This random data is named as initialization vector, also denoted by IV. IV can be assigned to any random value; since it' s used to make each message unique and to enable the encryption / decryption of the first block using the feedback register, it has no special meaning or value.[94] IV is considered as an initial parameter value throughout the CBC algorithm. In addition, it' s proven that IV need not have to be kept secret or be protected; it' s referred as a dummy ciphertext block.[95] CBC mode can also be checked out from Figure 2.6.

CBC is accepted as an appropriate mode for encrypting data longer than 64 bits, especially proven to be more secure than ECB mode.[96] The diffusion in the ciphertexts and the avalanche effect is much higher than ECB which is also an advantage for CBC. Another advantage of CBC is its potential usage for authentication.[97]

On the other hand, a single bit error in any plaintext block affects not only that ciphertext block but also all the other subsequent ciphertext blocks. But after decryption, due to the CBC mode' s structure, only the plaintext block with that incorrect bit will again be recovered with a single bit error while all the other plaintext blocks being retrieved without errors. As a result, it can be concluded that in CBC mode, errors in the plaintext has a significant effect on the ciphertext whereas the

---

[94] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 194.
[95] *ibid*, p. 194.
[96] William Stallings, *Network and Internetwork Security - Principles and Practice*, p. 59, 61.
[97] *ibid*, p. 61.

decrypted plaintext will only consist of errors coming from the original plaintext, thus plaintext errors (if few in amount) doesn' t cause too much trouble.[98]

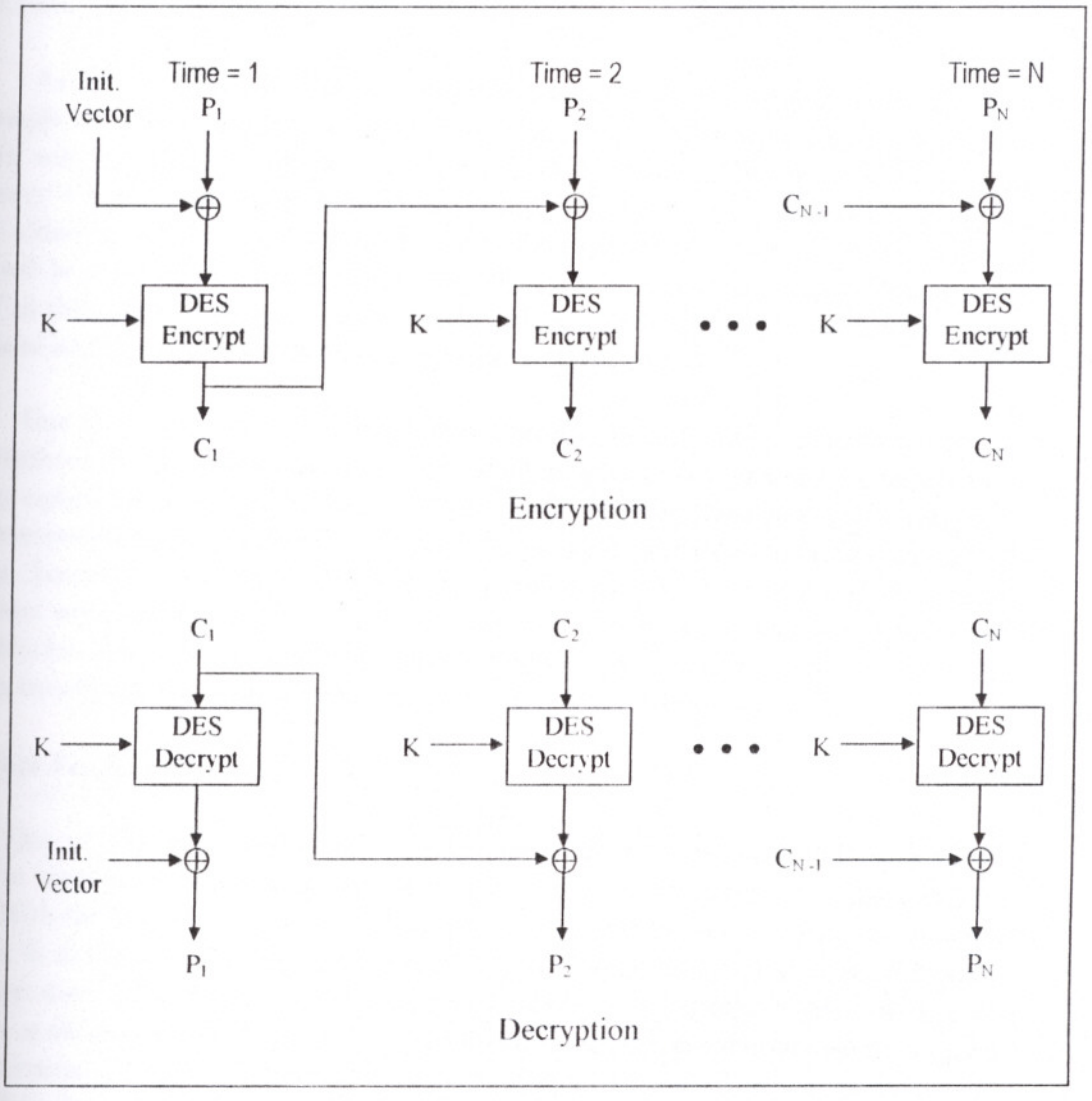


Figure 2.6 Cipher Block Chaining Mode.[99]

More or less, in CBC the real problem is considered to be the ciphertext errors. For instance, only a single bit error in any of the ciphertext blocks due to transmission signal noises or an intentional eavesdropper attack will have a much higher effect and damage on the recovered plaintext compared to the previous error type. A single bit error in ciphertext block C_n will cause the corresponding plaintext block P_n to be entirely incorrect plus 1-bit error in the subsequent block P_{n+1}. This can be considered as a disadvantage for CBC mode, since its structure allows *error extension* problem whenever ciphertext errors occur. If, there are single bit errors in each of the ciphertext blocks, this will end up with a complete garbage of plaintext filled with errors. However, it must be stressed that the errors in a ciphertext block affects two plaintext blocks: the corresponding plaintext block and the following one. Thus, the other plaintext blocks are not affected with the errors in that ciphertext block and can be

[98] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 195.
[99] William Stallings, *Network and Internetwork Security - Principles and Practice*, p. 60.

recovered correctly. This property is known as *self-recovering* and it enables the CBC mode to recover from bit errors quickly. As a conclusion, it' s stated that in CBC mode, ciphertext errors are more effective and more spreadable than ECB mode, but error detection is possible.[100]

As seen in the ECB, fabrication in the ciphertext is also troublesome for CBC. If a single bit is lost or added to any of the ciphertext blocks, then all the subsequent blocks will be affected and all the corresponding plaintext blocks will be entirely incorrect. For example, if the first bit is deleted from the first ciphertext block or one bit is added to the first position of the first ciphertext block, then all of the plaintext data will be affected and the complete plaintext will be a garbage where almost every bit of all the plaintext blocks will be wrong. Thus, synchronization problem is a great overhead and threat to the reliability and usage of CBC mode.[101]

Due to its structure, CBC mode has some other potential problems which can be considered as other disadvantages. First of all, a cryptanalyst can make changes on a chosen ciphertext block so that he / she can easily observe the controlled changes in the corresponding plaintext blocks. Thus, CBC mode is vulnerable to chosen ciphertext attacks. Secondly, an intruder can add extra ciphertext blocks to the end of original ciphertext message and in some cases the receiver might not detect those additional fraud blocks. Thus, to overcome this problem, some extra precautions must be taken such as structuring the original plaintext before the encryption process.[102]

- **Cipher Feedback Mode - CFB**

One of the most well known modes which can be implemented for a block cipher to work as a self-synchronizing stream cipher is the cipher feedback (CFB) mode. With the usage of CFB, block encryption algorithms such as DES can process the data in streams of bits or bytes one at a time rather than fixed sizes of blocks. Implementation of a block cipher in stream mode is especially needed in secure network applications and in real-time operations.[103] (ie, when each character is typed from a terminal, it must be immediately encrypted and transmitted to the receiver host, and must be decrypted whenever it' s received with no time or data loss.)

In CFB mode, the length of data encrypted & decrypted at a time unit can be chosen any length, ranging from 1-bit to *j*-bits. But, in most ciphers like DES, 8-bit CFB is often chosen because applications require 1 character transmission at a time which is 8 bits in length, thus if more than 8 bits are used during encryption transmission capacity would be wasted and if a size less than 8-bit is chosen this would bring overhead in design and also would result with performance degradation.[104]

The operation principle of a *j*-bit CFB mode working on a 64-bit block cipher will be explained here which is also represented in the Figure 2.7. As similar to CBC, an initialization vector (IV) is used which is 64 bits in length, equivalent to block size.

---

[100] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 196.
[101] *ibid*, p. 196.
[102] *ibid*, p. 196.
[103] *ibid*, p. 200.
[104] William Stallings, *Network and Internetwork Security - Principles and Practice*, p. 61.

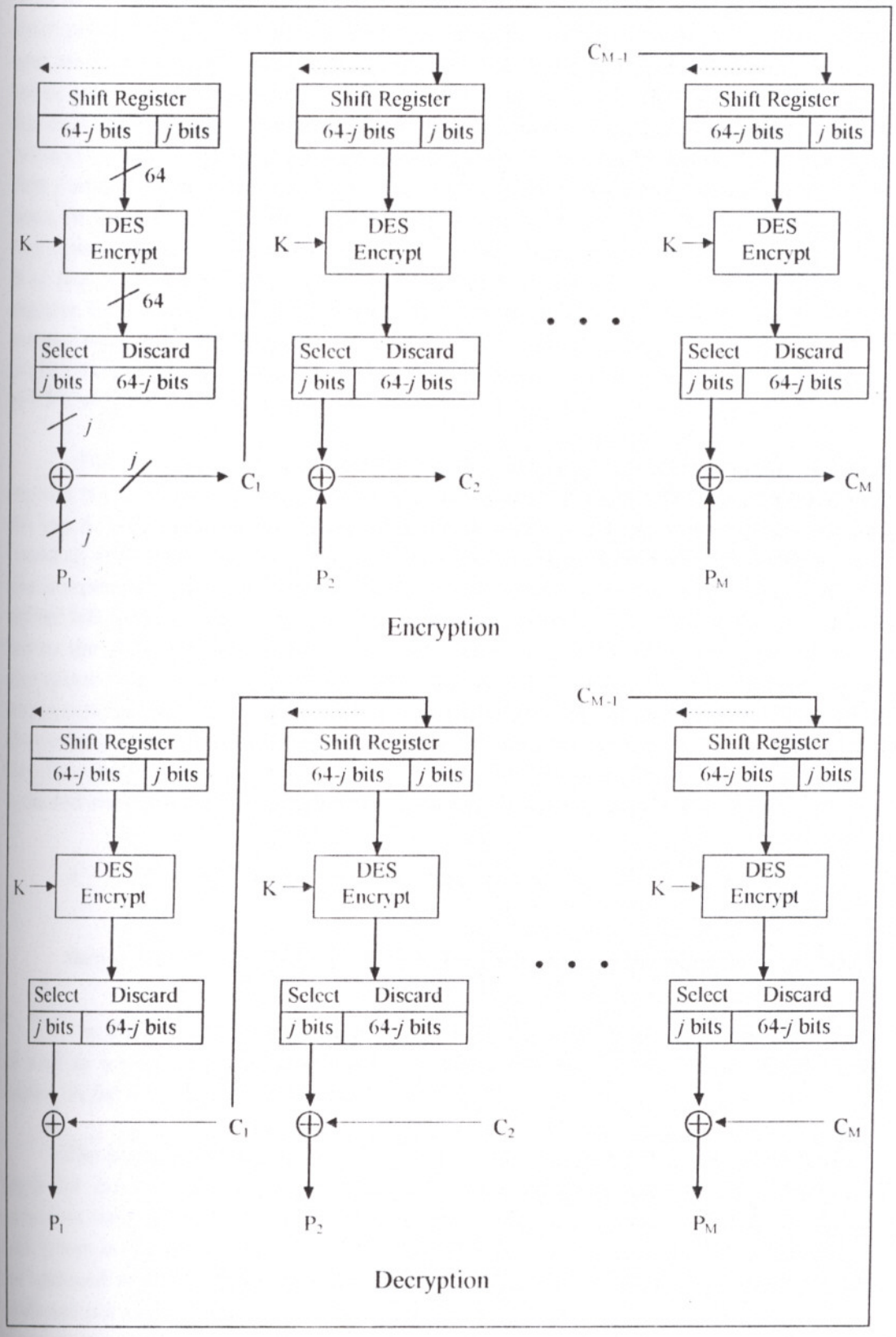


Figure 2.7 J-Bit Cipher Feedback Mode.[105]

[105] William Stallings, *Network and Internetwork Security - Principles and Practice*, p. 62.

This IV is loaded into a shift register as an initial operation both during encryption and decryption. The IV need not be secret, as in CBC, but the IV must be a unique value and must be changed within every message. This is an essential requirement for CFB mode (this was not necessary in CBC mode); because unless the IV is not kept unique for each message, the cryptanalyst will have a chance to recover the plaintext.[106] The left-most j bits of the IV is encrypted with the key K and this result is XORed with the first j bits of the plaintext message. The output is the first j-bit ciphertext which is also input to the shift register for the next round. The shift register is shifted left j bits and this ciphertext output is put to the end of the shift register as the right-most j bits. This is in fact, a simple queue mechanism which also changes the value of IV in the shift register before each round. The same encryption process is repeated in an iterative manner until the end of plaintext is reached. Each ciphertext bit is dependent on all the previous ciphertext bits and hence plaintext characters, which was also a valid property in CBC mode.[107]

The decryption is very similar to the encryption process. Again, the shift register is initialized with the same IV value, and the left-most j bits are encrypted with the key K and the result is XORed with the first ciphertext bits coming from the first round of encryption. As a result, the first j bits of the plaintext will be retrieved, and this is repeatedly processed until the last ciphertext unit. Also, for each round, j bits are shifted left from the left-most part and j bits of ciphertext from the previous round are fed to the shift register as the right-most j bits which is exactly the same as in the encryption stage. It must be stressed that in CFB mode, during decryption the encryption function of the cipher is used instead of the decryption function. This can be shown by a mathematical notation as follows, where for each round i, and for each j bits, P_i is the j-bit unit of plaintext and C_i is the j-bit unit of ciphertext for i^{th} round, IV_j is the left-most j-bit of the initialization vector, K is the key used in the cipher;

if $\quad C_i = P_i \oplus E_K\left(IV_j\right)$, for encryption

then $\quad P_i = C_i \oplus E_K\left(IV_j\right)$, is true for decryption by the transitivity of XOR.

The CFB mode could be implemented for any type of block cipher algorithms as well as 64-bit block ciphers like DES. The differences in block size would only impact on the size changes of the IV.

The plaintext errors are also faced as a problem in CFB mode where even a single bit error in the plaintext affects the corresponding and all the subsequent ciphertext data. However, the error is reversed after decryption, and only the bit(s) with errors in the plaintext is decrypted with errors while all the subsequent plaintexts are retrieved without errors. The plaintext error effects are similar to the ones in CBC mode and not considered as an important problem.[108]

---

[106] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 201.
[107] *ibid*, p. 200.
[108] *ibid*, p. 201.

Because of the structure of CFB mode, ciphertext errors are accepted as an important problem and disadvantage. A single bit error in the ciphertext not only causes the corresponding plaintext bit to be recovered incorrectly but also affects a large part of the subsequent plaintext data. Since, that incorrect bit in the ciphertext is also forwarded into the shift register, it will make the shift register contain a wrong value and all the subsequent ciphertexts will be produced and decrypted incorrectly until that bit is carried off from the shift register. For instance, in 8-bit CFB mode, a single bit error causes a total of 9 bytes of plaintext to be decrypted entirely wrong. It is proven that for a *k*-bit block cipher using *j*-bit CFB mode, a single bit error affects the decryption of the current and subsequent k / j-1 blocks.[109]

A side-effect of ciphertext error is another problem which is that if an intruder knows the plaintext message in any transmission, he / she can change bits in a chosen ciphertext block so that it could be decrypted to whatever he / she wishes on the receiving end afterwards. The changes might not be detected and the message could be accepted valid by the receiver.

Another disadvantage of CFB mode comes from the structure of self-synchronizing stream ciphers. It' s proven that self-synchronizing stream ciphers are vulnerable to playback attacks.[110] If any eavesdropper records some of the ciphertext bits, then he / she can put these data into the new ciphertext streams generated where the receiver cannot recognize that the data being decrypted is not the new and the actual one, but instead some older replayed data. This technique is known as the *playback attack* and since CFB mode is a self-synchronizing stream cipher, the playback attack can be considered as a potential threat for this mode.

One of the significant advantages of CFB mode when compared to ECB and CBC is its resistivity to synchronization problems due to its self-recovering structure. Any bit loss or extra bit addition to the ciphertext will affect only some portion of the plaintext but not all the subsequent plaintext data. In addition, since CFB is self-synchronizing, the errors can be detected and recovered during decryption.[111]

CFB mode is used both for encrypting messages of any length with high level of security and for authentication purposes.[112] CFB mode is mostly preferred in stream-oriented data transmission of any desired size. Like CBC, CFB mode provides high level of diffusion among the ciphertext data and high avalanche effect.

- **Output Feedback Mode - OFB**

The output feedback mode (OFB) is another method of implementing a block cipher so as to work as a synchronous stream cipher and it' s very similar to the CFB mode. The only difference in the algorithm is that, in OFB the *j-bit* output of the encryption function is fed back to the right-most positions of the shift register whereas in CFB the ciphertext output was to be the feedback for the shift register.[113]

---

[109] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 201.
[110] *ibid*, p. 199.
[111] *ibid*, pp. 201-202.
[112] William Stallings, *Network and Internetwork Security - Principles and Practice*, p. 62.
[113] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 203.

As seen in CFB, the decryption is the reverse of encryption algorithm in OFB mode. Again, for the decryption process, the block encryption mode is used. All the other mechanisms and processes are just the same as in CFB mode for both encryption and decryption. This can be deduced from the Figure 2.8 where a *j-bit* OFB mode is shown. As in CFB mode, the length of the shift register and *j* can be of any length, but for DES and DES-like cryptosystems, the shift register is 64 bits and *j* is 8 bits in length for the conventionality and performance considerations.

It should also be stressed that both OFB and CFB modes allow encryption with a variety of block sizes, the block size can be chosen any length if necessary.[114]

In OFB mode, there's a term named as *internal feedback* and it comes from the fact that; since the selected output of the shift register which is previously encrypted with key K is fed back into the shift register for the next round, the feedback mechanism is independent of the plaintext and ciphertext streams.[115] This property of OFB mode brings a unique feature which does not exist for CFB; most of the encryption process can be carried out independently and offline, even before the plaintext is retrieved. Only using the shift register, the encryption can be processed for each and every shift register block; and when the plaintext data is acquired finally, it can be XORed with the encrypted output from the shift register. This is also true for the decryption algorithm, where the shift register can be pre-processed and encrypted independently from the ciphertext blocks, and XORed with the ciphertext streams afterwards.

The algorithm for a *j-bit* OFB can also be described as follows; where for each round i, and for each j bits, P_i is the j-bit unit of plaintext and C_i is the j-bit unit of ciphertext for i^{th} round, S_i is the state vector or the output of the shift register, K is the key used in the cipher:

for encryption: $C_i = P_i \oplus S_i$, where
$S_i = E_K(S_{i-1})$

for decryption: $P_i = C_i \oplus S_i$, again where
$S_i = E_K(S_{i-1})$

The S_i' s for each round are the outputs of shift registers and they are independent of plaintexts and ciphertexts.

In OFB, the initialization vector IV is used which must be initially loaded into the shift register before the first round. For each message, this IV must be a unique value, as seen in CFB, but IV doesn' t have to be kept secret in OFB mode.[116]

---

[114] Eli Biham, "Cryptanalysis of Multiple Modes of Operation", Journal of Cryptology, vol.11 Number 1, p. 45, 1998.
[115] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 203.
[116] *ibid*, p. 204.

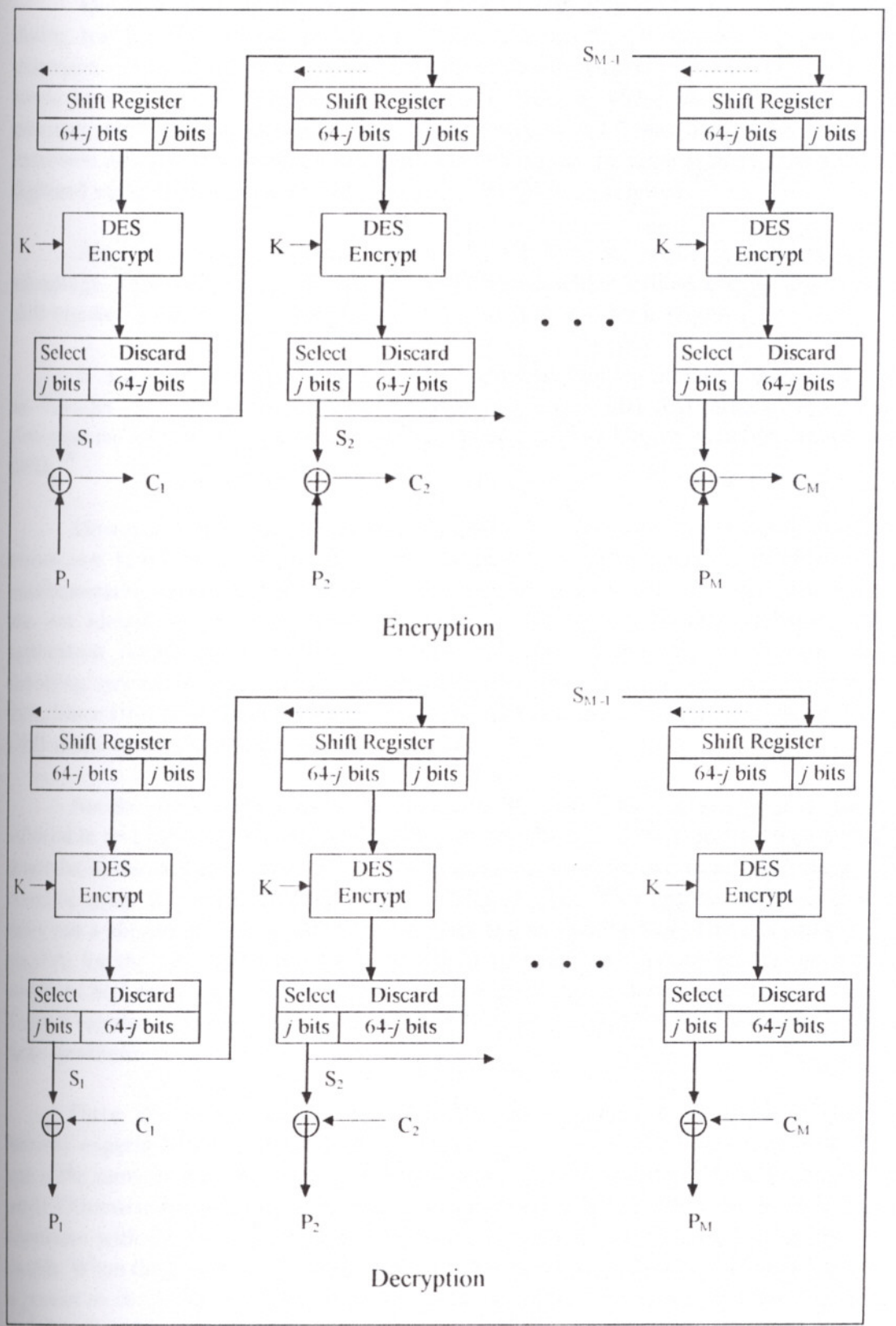


Figure 2.8 J-Bit Output Feedback Mode.[117]

[117] William Stallings, *Network and Internetwork Security - Principles and Practice*, p. 63.

One of the significant advantages of OFB mode is that bit errors that occurred during transmission do not propagate.[118] In other words, OFB mode has no error extension. Thus, if a single bit error occurs in the ciphertext C_i, this only causes a single bit error in the corresponding recovered plaintext where the other subsequent plaintext units are not corrupted. Due to its structure, CFB has the trouble of error extension and bit errors might propagate; for this reason, in some applications such as digitized signal, voice, image, video transmissions OFB is preferred.[119]

Also, the internal feedback property of OFB mode can be considered as an advantage, especially for some specific applications where offline encryption of the shift register is necessary or pre-processing of the shift register is required.

It' s also proven that in OFB mode, a chosen plaintext attack does not enable an intruder to gain more information than a known plaintext attack, thus the performance of chosen plaintext attack is no better than known plaintext attack in OFB.[120]

However, OFB has some disadvantages. For instance, the synchronization errors are fatal and OFB mode does not have the self-recovering property for synchronization errors. If the shift registers on the encryption and the decryption ends are not identical, then the recovered plaintext will be a complete garbage. Any application implemented with OFB mode must have additional mechanisms for detecting synchronization loss and for regaining synchronization as well as recovery of data. Since CFB is self-recovering by its nature, it' s considered a better algorithm than OFB when synchronization problem is the case.

Another disadvantage of OFB comes from the fact that it' s proven to be more vulnerable to a message-stream modification attack than CFB.[121] Since, a change in a single bit in the ciphertext only affects the corresponding plaintext, controlled changes could be made to the selected ciphertext / plaintext pairs. This enables an attacker to carry out a chosen-ciphertext attack and deduce the encryption key. This also makes it possible for the eavesdropper to alter the ciphertext so that it could be decrypted to whatever he / she wishes on the receiving end without being detected by the receiver. These were also considered as a problem for CFB mode, but in OFB, it' s proven to be more troublesome.

There also exists another security problem whenever OFB mode is used. Security experts advise that OFB should be used or chosen only when the feedback size is the same as the block size. (ie. for 64-bit DES, only 64-bit OFB mode must be used) Otherwise, it will hopefully breed security flaws. Since OFB mode XORs a keystream with the text block, this keystream may repeat after a period of cycles or rounds. When the feedback size is equivalent to the block size (*k-bits*), the block cipher is proven to show the characteristics of a permutation of *k-bit* values and the average cycle length is 2^k - 1. For a 64-bit block, this makes 2^{64} - 1, which is considerably a big

---

[118] William Stallings, *Network and Internetwork Security - Principles and Practice*, p. 63.
[119] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 204.
[120] Eli Biham. "Cryptanalysis of Multiple Modes of Operation", Journal of Cryptology, vol.11 Number 1, p. 45, 1998.
[121] William Stallings, *Network and Internetwork Security - Principles and Practice*, pp. 63-64.

value. But when the feedback size is less than block size k, the average cycle length drops to around $2^{k/2}$. Thus, for a 64-bit block cipher, this is approximately 2^{32}, a value not long and big enough.[122]

OFB is mostly useful in applications such as stream-oriented transmission of data over noisy channels with a moderate level of security. It' s also proposed that OFB mode is designed to act as a pseudorandom bit generator as well as allowing precomputation of a major part of the encryption process.[123]

2.4.4 Security Strength and Weaknesses of DES

Since its invention and usage, the security of DES has been questioned and has been a common subject of argument in cryptology until today. Several weaknesses have been proven, but no exact proof of a complete weakness or security hole in its design has been laid so far. But due to its structure and especially its key length, security strength of DES is proven to be continually decreasing and it' s shown that DES is computationally not secure today as it has been in 1970's and 80's. This has brought the necessity of upgrading DES to a higher security level, or producing new variants of DES, or inventing new alternatives to it. These will be discussed shortly in section 2.4.5.

2.4.4.1 Security of DES

There has been even no proof of weakness in its inner structure and design. In other words, theoretically DES hasn' t been proven yet to be completely insecure. There are some known weaknesses in its design but these are not considered to be threatening or inevitable problems. In fact, there has been some speculations and arguments on the key length, iterations, design of the S-boxes, existence of a probable security flaw in the algorithm or basic structure of DES as well as some myths claiming that certain trapdoors exist in DES which might have been embedded intentionally by the designers in IBM or by the NSA experts afterwards.[124]-[125] Especially, there has been some arguments that NSA made some changes in the design for the establishment of hidden trapdoors, even a US Congressional inquiry was made which ended with a formal and unclassified report exonerating NSA from any improper involvement in the DES design.[126]-[127]

- **Design of the S-boxes and the Algorithm**

Most of the arguments considering the security of DES was based on the design of the S-boxes, because S-boxes are accepted to be the core of DES security. The non-linearity of S-boxes mostly give the DES its security strength. Several researches and analyses have been carried out and no flaw is revealed so far in the

---

[122] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 205.
[123] Eli Biham, "Cryptanalysis of Multiple Modes of Operation", Journal of Cryptology, vol.11 Number 1, p. 45, 1998.
[124] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 278.
[125] Charles P. Pfleeger, *Security in Computing*, pp. 117.
[126] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 278.
[127] Charles P. Pfleeger, *Security in Computing*, p. 117.

functioning of S-boxes.[128] On the other hand, the design of S-boxes is also analyzed by several researchers and no precise theoretical weakness is found in the design of S-boxes except some oddities and its vulnerability to differential and linear cryptanalytic attacks[129] which showed that S-boxes and hence DES might be susceptible to such cryptanalytic attacks (requiring a great amount of time and storage capacity); but these did not impose an exact weakness in the design of S-boxes that could prove DES to be entirely insecure. Even today, some argue that there might be several trapdoors in S-boxes, embedded by NSA. In 1970's, *The Bell Laboratories* stated that the S-boxes might have trapdoors, but gave no scientific evidence.[130] Also, *The Lexar Corporation*' s report on the S-box analysis was concluded with a remark stating that;

...the problem (of the search for structure in the S-boxes) is complicated by the ability of the human mind to find apparent structure in random data, which is really not structure at all.[131]

However, in that conclusion, it was also stated that;

Structures have been found in DES that were undoubtedly inserted to strengthen the system against certain types of attack. Structures have also been found that appear to weaken the system.[132]

In the meantime, NSA released some important information about several design criteria in S-boxes; which, in a way, indicates the security burdens of S-boxes and how a secure S-box design should be.[133]

- No S-box is a linear or affline function of its input; that is, the four output bits cannot be expressed as a system of linear equations of the six input bits.
- Changing one bit in the input of an S-box results in changing at least two output bits; that is, the S-boxes diffuse their information well throughout their outputs.
- The S-boxes were chosen to minimize the difference between the number of 1's and 0's when any single input bit is held constant. In other words, holding a single bit as 0 or 1 and changing the bits around it should not lead to disproportionately many 0's or 1's in the output.

It should be noted that, after the differential cryptanalysis technique was known in general, IBM published a more detailed and extensive design criteria for S-boxes and P-boxes in 1990' s.[134]

Besides, some analyzes have shown that the design of S-boxes is crucial for DES security since some slight changes could easily cause the security of DES to be weakened considerably. Indeed, it' s proven that the order of the S-boxes or the entries' allocations in these boxes affect the overall security of DES, especially against

---

[128] Charles P. Pfleeger, *Security in Computing*, p. 117.
[129] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 285.
[130] *ibid*, p. 284.
[131] *ibid*, p. 285.
[132] *ibid*, p. 284.
[133] Charles P. Pfleeger, *Security in Computing*, p. 117.
[134] Bruce Schneier, *Applied Cryptography - Second Edition*, pp. 293-294.

differential cryptanalysis. A related comment was made by *Biham* and *Shamir* stating that;

The replacement of the order of the eight DES S-boxes (without changing their value) also makes DES much weaker: DES with 16 rounds of a particular replaced order is breakable in about 2^{38} steps...DES with random S-boxes is shown to be very easy to break. Even a minimal change of one entry of one of the DES S-boxes can make DES easier to break.[135]

As well as S-boxes, all the other parts of DES algorithm has been analyzed extensively and no serious flaw has been detected so far.[136] Some researches aiming theoretical or mathematical proofs of security flaws in the DES structure such as non-randomness or direct statistical dependencies were also unsuccessful.[137]

- **Number of Rounds**

Many analyses have also shown that the number of rounds or iterations being chosen 16 for DES was proven to be adequate and sufficiently secure. Implementations with iterations less than 16 rounds were proven to be much less secure against differential cryptanalysis.[138] Also, it' s proven that increasing the number of rounds to values greater than 16 do not improve the security strength or increase the diffusion level, thus 16 is considered to be an optimum number for DES iterations with sufficient avalanche effect and security.[139]-[140]

- **Algebraic Structure**

Another critical point in the security of DES is that DES is proven to be not a group. The group property of an encryption algorithm can be explained simply as follows; the elements of a group are the ciphertext blocks with each possible key where composition is the group operation. If an algorithm shows a group structure, then multiple encryptions under multiple keys should give exactly the same result with single encryption under a single key. Thus, it' s stressed that if an encryption algorithm forms a group under any combinations of keys, then that algorithm theoretically has a pure weakness in its structure and should not be used. It's also proven that even not having an exact group structure; if an algorithm is fairly close to being a group, then that encryption algorithm is poor in design and might yield to security weaknesses. Several studies and researches were carried out whether DES was having a group structure or closed properties, and none were found. Finally in 1992, it was proven that DES is not a group.[141]-[142]

---

[135] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 296.
[136] Charles P. Pfleeger, *Security in Computing*, p. 117.
[137] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 285.
[138] *ibid*, p. 284.
[139] *ibid*, p. 284.
[140] Charles P. Pfleeger, *Security in Computing*, p. 118.
[141] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 283, 348.
[142] RSA Data Security Inc, "Answers to FAQ About Today's Cryptography", RSA Laboratories paper, p. 64, 70, 1996.

- **Key Length**

One of the main objectives related with security strength of DES is the key length, which has always been a question in common. Originally, DES key was chosen to be 56 bits in length, which became as a standard afterwards. In the 1970' s and early 1980' s 56 bits was computed and accepted as an adequate and sufficiently secure key length for DES. Since a brute-force attack requires 2^{55} - 2^{56} computations to deduce the key, this means testing all the possible key values approximately would take 7.2 * 10^{15} sec, or about 228 million years if each key is tested in 100 ms. Even test time per each key is assumed to be 1 μs, the total search time would take about 2280 years.[143] Considering the computing capacity, hardware / software technology at the 1970' s, these values were the possible limits with feasible money costs. If, only very high expenses were taken into consideration, the total time for exhaustive DES key search would be reduced to days. For instance, in 1977 *Diffie* and *Hellman* suggested that a specially designed parallel DES-cracking machine with 10^6 chips could recover the 56-bit key in 1 day with a cost of $20 million.[144]-[145] In 1981, *Diffie* updated his calculations and predicted a time limit of 2 days within a cost of $50 million.[146] Regarding these costs, at the 80' s, DES was still accepted strongly secure but the security experts asserted that by 1990, DES would become totally insecure and the key length should have been increased.

In fact they were right, thanks to both hardware / software advances in computing technology and new cryptanalytic attack techniques. In 1993 *Michael Wiener* designed a machine that could find the 56-bit DES key by brute force attack in an average of 3.5 hours within a cost of $1 million.[147]-[148] He also reported a design of his own that uses a pipeline architecture to break the key at a speed of 50 million keys per second.[149] As the years went passing by, things got more realistic and more tragic in practice for 56-bit DES. In February 1998, *Distributed.Net* won the *RSA*' s DES Challenge II-1 by cracking in 41 days, but at a very low expense, where only a distributed cracking software on a moderate computer and several thousand computers in contribution via Internet were used.[150] In July 1998, *Electronic Frontier Foundation (EFF)* won the *RSA*' s DES Challenge II-2 in 56 hours using a specially designed machine named "DES Cracker" which cost about $250,000.[151] In January 19, 1999, *Distributed.Net* worked together with the *EFF*' s "Deep Crack," a specially designed supercomputer and a worldwide network of nearly 100,000 PC s on the Internet to crack 56-bit DES, and won *RSA Data Security*'s DES Challenge III in a record-breaking 22 hours and 15 minutes.[152] *EFF*'s "Deep Crack" and the *Distributed.Net* computers' testing capacity was approximately 245 billion keys per second[153], a terrific

---

[143] Charles P. Pfleeger, *Security in Computing*, p. 118.
[144] *ibid*, p. 118.
[145] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 283.
[146] *ibid*, p. 283.
[147] *ibid*, p. 284.
[148] William Stallings, *Network and Internetwork Security - Principles and Practice*, p. 54.
[149] *ibid*, p. 54.
[150] RSA Data Security Inc., RSA '99 Press Release - Internet Document, http://www.rsa.com/pressbox/html/990119-1.html, 1999.
[151] *ibid*.
[152] *ibid*.
[153] *ibid*.

improvement in performance when compared to capacity of 1 million keys per second in the mid-80' s.[154]

As well as advances in exhaustive key search attacks, new methodologies in cryptanalysis made 56-bit DES security more doubtful. In 1990, *Biham* and *Shamir* invented a technique named as differential cryptanalysis, which reduced the 2^{55} complexity of brute force attack to 2^{37}. This proved that 56-bit DES was not secure as it was thought to be in the previous decades, not only in practice in but also in theory. The only infeasibility with differential cryptanalysis is the storage capacity since this method requires 2^{47} plaintext / ciphertext pairs at least. Similarly, another attack by *Matsui* known as linear cryptanalysis needs 2^{43} known plaintexts to crack 56-bit DES. The storage requirements make these types of attacks impractical at the moment, but improvements and some changes will probably decrement these values. Still, some argue that 56-bit key length would be enough for moderate DES security, whereas most of the people believe that 56 bits do imply a weakness rather than a strength and strongly suggest the use of key lengths much bigger than 56, or new alternatives to DES.

2.4.4.2 Weaknesses of DES

As mentioned in section 2.4.4.1, although there are known weaknesses in DES, it' s proven that these are not such serious problems that would endanger the effectiveness and security of the algorithm. Only to give an idea, these weaknesses will be explained shortly in this section.

- **Weak Keys**

In DES, and in also some other symmetric ciphers, due to the structure of the encryption algorithm, some of the key values do not change the ascii values of encrypted blocks, in other words, the encrypted ciphertext block is exactly same of the plaintext block. These key values are named as weak keys.[155] Since in DES encryption, the initial key is split into two halves and each half is independently shifted for each and every round; if all of the bits of each half is 0 or 1 as the initial key value, then the key value used for each of the 16 round will be exactly the same. Thus, XORing the key with plaintext won' t make any changes in the ciphertext. There are 4 known weak keys for 56-bit DES; all 0' s, all 1' s, left half all 1's and right half all 0' s, vice versa (actual initial key values).[156]-[157] If the key k is chosen as one of these four values, then encryption in DES can be denoted as follows;

$$E_k(P_i) = C_i \qquad \text{where} \qquad C_i = P_i \qquad \text{for all } i$$

[154] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 284.
[155] Charles P. Pfleeger, *Security in Computing*, p. 120.
[156] *ibid*, p.120.
[157] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 280.

- **Semi-Weak Keys**

The uniqueness of the keys used in encryption algorithms is considered as an important criterion where each different key value (even a single bit) should produce a different ciphertext from the same plaintext. But in most of the algorithms, some key values don' t satisfy this condition which is also true for DES. In other words, some key values produce exactly the same ciphertext from the same plaintext when 56-bit DES encryption algorithm is used. In cryptology, this phenomenon is named as key clustering where the semi-weak keys are key clusters.[158] The problem can be denoted more mathematically as follows;

for two key values k_1 and k_2 where $k_1 \neq k_2$;

$$C_1 = E_{k_1}(P)$$
$$C_2 = E_{k_2}(P)$$

and hence,

$$C_1 = C_2$$

This implies that k_1 can decrypt any message encrypted under k_2 or vice versa.[159-160] This is due to the fact that if the initial key values are chosen semi-weak keys, then these key values will generate only two different subkeys instead of 16 throughout the 16 iterations.[161] There are 6 semi-weak key pairs known and any of these 12 keys can be avoided during encryption.

In addition to the semi-weak keys, in DES there are 24 pairs of known possibly weak keys. In fact, possibly weak keys are very similar to semi-weak keys, but each of these possibly weak key produce 4 subkeys throughout 16 rounds of DES, each used 4 times.[162] These 48 key values can also be checked and avoided when a key is to be chosen.

Since all of the weak and semi-weak key values are publicized and known, avoiding them while choosing a key will entirely eliminate all possible security problems and vulnerabilities. However, there are totally 64 weak keys among 2^{56} possible key values, and the probability of choosing a weak key is very low ($1 / 2^{50}$); thus this is not considered as a serious weakness and it' s stressed that DES cannot be criticized as a poor encryption algorithm for this. Because of this extremely low probability, it' s also argued that even taking precautions against weak keys and

[158] Charles P. Pfleeger, *Security in Computing*, p. 121.
[159] *ibid*, p. 120.
[160] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 280.
[161] *ibid*, p. 280.
[162] *ibid*, p. 281.

checking them should not be bothered, yet the other security experts do not say so.[163]-[164]

- **Complement Keys**

If a plaintext is encrypted under a key value with the output ciphertext, then the complement of this encryption is the complement of plaintext encrypted under the complementary value of the key resulting with the output ciphertext which is complement of the previous ciphertext.[165]-[166] The complement operation is the bit-wise 1' s complement of the original value, in other words, all the 1' s in the plaintext, key and ciphertext are replaced with 0's and 0' s with 1' s respectively. This can be denoted as follows where ' stands for the complement operation;

$$E_K(P) = C$$

$$E_{K'}(P') = C'$$

This phenomenon occurs in DES and in most of the DES-like symmetric ciphers because in each round, the subkeys are XORed with the right half after the expansion permutation.[167] The problem which the complement keys might cause is that the computational complexity is halved in cryptanalytic attacks. For instance, exploiting this property in a chosen-plaintext attack against DES, a cryptanalyst needs to test half the possible keys, reducing the test key range from 2^{56} to 2^{55}.[168]

The complementary fact is not accepted as a serious problem or threatening weakness, because most messages during encryption do not have corresponding complements. In other words, in a random plaintext message the probability of occurrence of a complement block with the normal block is very low and users can be warned not to use complement keys.[169]-[170]

- **Design Weaknesses**

There are also some oddities or negligible weaknesses in the design of DES algorithm which are discovered by the designers or some researchers. For instance, it' s found out that the expansion permutation EP in DES repeats the first and fourth bits of every 4-bit series by crossing the bits from the neighboring 4-bit series.[171] This slight weakness is shown to be caused by the structure of the expansion permutation, a weak point in the algorithm. Similarly, there are some small weaknesses found out in the design of S-boxes, such as some chosen different inputs to some S-boxes produce the

---

[163] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 281.
[164] RSA Data Security Inc, "Answers to FAQ About Today's Cryptography", RSA Laboratories paper, p.69, 1996.
[165] Charles P. Pfleeger, *Security in Computing*, p. 120.
[166] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 281.
[167] *ibid*, p.281.
[168] *ibid*, p.282.
[169] *ibid*, p.282.
[170] Charles P. Pfleeger, *Security in Computing*, p. 120.
[171] *ibid*, p. 121.

same outputs, or the imbalance in some of the S-boxes' entries. Also it has been found out that by changing some of the chosen bits in only three neighboring S-boxes, it' s possible to get the same output of a single DES round. The last three output bits of the fourth S-box can be obtained in the same way as the first by complementing some of the input bits.[172]-[173]

These kind of weaknesses are not considered as serious problems or harmful vulnerabilities for DES security; however, it' s proven that some of the design weaknesses in S-boxes contribute to the differential and linear cryptanalysis.

2.4.5 Other DES Models

Due to the technological improvements and new cryptanalysis techniques, new alternatives to the standard 56-bit DES having enhanced security and much greater resistivity along with the equivalent efficiency and speed were being searched by the end of 80' s, and still new products are being developed today. Some of these alternatives have sought for increased key length while others made slight or significant changes to the design and structure of DES. Another approach has been the development of new alternative algorithms other than DES. All of these will be discussed throughout this section.

2.4.5.1 Multiple DES

One of the common approaches is increasing the key length of 56-bit DES while making no or very few changes to the algorithm. Using multiple keys and multiple encryptions is one of the solutions, since n number of different keys of 56-bit correspond to a key $n*56$ bits in length, thus an increase in the key size. There are several models of multiple DES, double-DES, triple-DES with two keys, triple-DES with three keys, 3-PEK which some are proven to be inefficient and not secure enough whilst some are considered moderately good but not accepted as a standard in common.

Some of these multiple DES models are summarized in this part, but before going any further, a fact should be noticed. It' s stressed by the cryptographers that while building a new improved system Y instead of the cryptosystem X, there are several common criteria that must be followed. These are:[174]

1. Keys in Y are significantly longer than keys in X.
2. Given an appropriate assumption about the security of X, Y is evidently almost as hard to break as X under any natural attack such as known-plaintext, chosen plaintext, etc.
3. It can be convincingly argued that Y cannot be broken faster by an exhaustive key search, and therefore much stronger than X.

---

[172] Charles P. Pfleeger, *Security in Computing*, p. 121.
[173] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 285.
[174] Ivan B. Damgard and Lars R. Knudsen, "Two-Key Triple Encryption", Journal of Cryptology, vol.11 Number 3, p. 210, 1998.

These are also true for DES. Besides, whenever multiple encryption is the choice, the two well known theorems are taken into consideration, which are also followed in the design of multiple DES models. Theorem 1 is provided by *Even* and *Goldreich* stating that;

A cascade of ciphers is at least as hard to break as any of the component ciphers in attacks where an attacker cannot make use of plaintext statistics.[175]

Theorem 2, asserted by *Maurer* and *Massey* is as follows;

Under any attack, a cascade of ciphers is at least as hard to break as the first cipher.[176]

- **Double-DES**

The simplest choice was double-DES, which was later on proven to be inefficient and far less secure than expected. The basic idea is taking two different 56-bit keys K_1 and K_2 and applying a double encryption with them which would make up an augmented key length of 2*56 = 112 bits.[177] The encryption and decryption in double-DES can be denoted as;

$$C = E_{K_2}\left(E_{K_1}(P)\right), \quad \text{for encryption}$$

$$P = D_{K_1}\left(D_{K_2}(C)\right), \quad \text{for decryption}$$

The overall mechanism is more or less proven to be the same as single DES, the only difference is doubling the encryption / decryption operations and using two initial keys instead of one. Initially, double-DES was thought to be much harder to break than 56-bit DES. In general, if 2^n attempts were required to break a key of length n by brute-force attack, then 2^{2n} trials would have been required to break the doubled key. This would mean 2^{112} trials to break double-DES.[178]-[179] But it was proven later by *Merkle* and *Hellman* that a *meet-in-the-middle-attack*[180]-[181]-[182] (a version of known-plaintext attack) could break double-DES in 2^{n+1} computations rather than 2^{2n}, which showed that double-DES was slightly more secure than 56-bit DES, but far less secure than it was expected. Thus, it was proven that double-DES would be a bad choice for replacing single-DES due to its insufficient efficiency and security, and wasn' t used aftermath.[183]-[184]-[185]-[186]

---

[175] Ivan B. Damgard and Lars R. Knudsen, "Two-Key Triple Encryption", Journal of Cryptology, vol.11 Number 3, p. 211, 1998.
[176] *ibid.* p. 211.
[177] William Stallings, *Network and Internetwork Security - Principles and Practice*, p. 64.
[178] *ibid.* p. 66.
[179] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 358.
[180] *ibid.* p. 358.
[181] William Stallings, *Network and Internetwork Security - Principles and Practice*, p. 66.
[182] Terry Ritter, "2xIsolated DES: Another Weak Two-Level DES Structure", Ritter Software Engineering White Paper, p.2, February 16, 1994.
[183] William Stallings, *Network and Internetwork Security - Principles and Practice*, p. 66.

- **Triple-DES with Three Keys**

Another variant of multiple encryption-based DES is triple-DES with three independent and different initial keys K_1, K_2, K_3, each 56 bits in length. The scheme can be denoted as follows;

$$C = E_{K_3}\left(E_{K_2}\left(E_{K_1}(P)\right)\right), \qquad \text{for encryption}$$

$$P = D_{K_1}\left(D_{K_2}\left(D_{K_3}(C)\right)\right), \qquad \text{for decryption}$$

This triple encryption with three different keys is also referred as DES-EEE3 mode.[187] There are also other modes of triple-DES with three keys such as DES-EDE3 mode[188], *Ellison*' s triple DES-EEE with Tran() function[189], or triple DES-EEE with pseudo-random generator[190], etc. DES-EDE3 mode' s operation can be shown as follows;

$$C = E_{K_3}\left(D_{K_2}\left(E_{K_1}(P)\right)\right), \qquad \text{for encryption}$$

$$P = D_{K_1}\left(E_{K_2}\left(D_{K_3}(C)\right)\right), \qquad \text{for decryption}$$

In fact, DES-EDE3 is very similar to DES-EEE3 except that for the second stage of encryption with the key K_2, decryption is used instead of encryption function, and encryption with K_2 instead of decryption is used in the second stage of decryption process. DES-EDE3 mode is said to be generated for the sake of compatibility with standard 56-bit DES; whenever needed, $K = K_1 = K_2 = K_3$ setting can be done in order to use DES-EDE3 as single DES with a key K.[191]

In theory, using three different keys would make an effect of 56*3=168-bit key length usage. But some analyzes proved that this was not so. It was shown that for n bits of length for each key, a meet-in-the-middle attack would require 2^{2n} trials and 2^n blocks of memory whereas due to $3n$ bits of total key size, the initial estimated trials for brute-force attack had to be 2^{3n} ; thus the security for triple-DES with three keys decreases to 2^{112}, instead of 2^{168} estimated.[192]-[193]-[194] To make matters worse, it' s also

---

[184] Terry Ritter, "The Context of the Fenced DES Design", Ritter Software Engineering White Paper, p.3, June 30, 1994.

[185] Ivan B. Damgard and Lars R. Knudsen, "Two-Key Triple Encryption", Journal of Cryptology, vol.11 Number 3, p. 211, 1998.

[186] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 358.

[187] RSA Data Security Inc, "Answers to FAQ About Today's Cryptography", RSA Laboratories paper, p.71, 1996.

[188] *ibid*, p. 71.

[189] Cryptography FAQ, Cryptosystems Journal, Internet Document, http://ourworld.compuserve.com/homepages/crypto/cryfaq05.htm, p. 4, 1998.

[190] *ibid*, p. 4.

[191] Phillip Rogaway, "The Security of DESX", RSA Laboratories' CryptoBytes, vol.2 Number 2, p. 8, 1996.

[192] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 360.

proven that the *related-key* attack can break a triple encryption with three keys in an approximate time of 2^n, ie, requiring about 2^{56} - 2^{72} computations for DES.[195]-[196]

However, computational resistance of 2^{112} is accepted to be sufficiently high enough considering enhancement of DES security (much more secure than 56-bit DES), thus some advise the usage of triple-DES with three distinctive keys.[197] But when related-key attack possibility is taken into consideration, and regarding the inefficiency and low speed for encryption / decryption, triple-DES with three keys is not favored and not used in common today.[198]-[199]-[200] In fact, it' s shown that triple-DES with three keys is much slower (nearly three times) than 56-bit single DES, both in hardware and software.[201]

- **Triple-DES with Two Keys**

Another variant of multiple DES models is the triple-DES with two keys which encryption of a plaintext block is iterated three times with two keys. The sequence is: Encrypt with the first key, then decrypt with the second key and finally encrypt with the first key again. This is the most commonly used triple-DES with two keys algorithm and it' s also referred as DES-EDE2.[202] There' s also DES-EEE2, but not preferred for the sake of compatibility with standard DES.[203]-[204] DES-EDE2 can be denoted as follows where the two keys are K_1 and K_2;

$$C = E_{K_1}\left(D_{K_2}\left(E_{K_1}(P)\right)\right), \qquad \text{for encryption}$$

$$P = D_{K_1}\left(E_{K_2}\left(D_{K_1}(C)\right)\right), \qquad \text{for decryption}$$

Triple-DES with two keys use two 56-bit keys which make up totally 2*56=112 bits of key length. Thus, in general, if n bits of two keys are used, then the

---

[193] Ivan B. Damgard and Lars R. Knudsen, "Two-Key Triple Encryption", Journal of Cryptology, vol.11 Number 3, p. 212, 1998.

[194] William Stallings, *Network and Internetwork Security - Principles and Practice*, p. 66.

[195] Ivan B. Damgard and Lars R. Knudsen, "Two-Key Triple Encryption", Journal of Cryptology, vol.11 Number 3, p. 212, 1998.

[196] John Kelsey, Bruce Schneier, David Wagner, "Key-Schedule Cryptanalysis of IDEA, G-DES, GOST, Safer and Triple-DES", Advances in Cryptology--CRYPTO '96 Proceedings, p. 249, 1996.

[197] RSA Data Security Inc, "Answers to FAQ About Today's Cryptography", RSA Laboratories paper, p. 72, 1996.

[198] Ivan B. Damgard and Lars R. Knudsen, "Two-Key Triple Encryption", Journal of Cryptology, vol.11 Number 3, p. 212, 1998.

[199] William Stallings, *Network and Internetwork Security - Principles and Practice*, p. 66.

[200] Terry Ritter, "The Context of the Fenced DES Design", Ritter Software Engineering White Paper, p.3, June 30, 1994.

[201] Phillip Rogaway, "The Security of DESX", RSA Laboratories' CryptoBytes, vol.2 Number 2, p. 8, 1996.

[202] RSA Data Security Inc, "Answers to FAQ About Today's Cryptography", RSA Laboratories paper, p. 71, 1996.

[203] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 359.

[204] Ivan B. Damgard and Lars R. Knudsen, "Two-Key Triple Encryption", Journal of Cryptology, vol.11 Number 3, p. 212, 1998.

lower resistivity bound for all the attacks is expected to be 2^{2n} computations, in consequence. For brute-force attack, this is accepted true and it has also been proven that differential cryptanalysis is extremely inefficient for DES-EDE2, requiring 10^{52} computations, which is a much greater value than single DES.[205]

However, it has been proven that this model was less secure than it was thought to be. Several variants of meet-in-the-middle attack have been applied to DES-EDE2 successfully and the keys are broken even faster than exhaustive key search for DES-EDE2, which is 2^{112}.[206] *Merkle* and *Hellman* developed a technique which could break DES-EDE2 within 2^{56} computations and 2^{56} blocks of memory (chosen plaintext/ciphertext pairs).[207]-[208] Later on, *Paul van Oorschot* and *Michael Wiener* improved these values by a known-plaintext attack, which required $2^{120} / p$ computations and p words of memory. In fact, this can be formulated in general as $2^{n+m} / p$ time, where n is the key size of each key and m is the block size in bits, and p is the number of known plaintexts. Hence, for DES-EDE2, $n=56$, $m=64$ and 2^{n+m} is 2^{120}. It should also be noted that if $p > 256$, then this attack is computationally faster than brute-force attack, since $2^{120} / p < 2^{112}$.[209]-[210]-[211] The security levels of several types of multiple encryption methods can be analyzed comparatively from Table 2.8.

Thus, it was proven that triple encryption with two keys was not a good alternative in general for single block ciphers such as 56-bit DES. The security performance was lower than needed, almost the same as single n-bit key cipher while significantly decreasing the encryption / decryption speed. Due to these shortcomings, DES-EDE2 and similar algorithms of triple encryption with two keys are not preferred and used as an alternative to single mode ciphers.[212] It should be noted that there are several variants of triple-DES algorithms with two and three keys aiming to increase efficiency and security of original triple-DES; such as Triple-DES with Inner CBC, Triple-DES with Outer-CBC, Triple-DES with CBCM (CBC with OFB Masking), etc.[213]-[214] However, these alternative modes are proven to be not efficient enough and do not impact significant improvements on the security of triple modes, or their resistivity limits to cryptanalytic attacks haven' t been tested yet.[215]-[216]

---

[205] William Stallings, *Network and Internetwork Security - Principles and Practice*, p. 67.
[206] Ivan B. Damgard and Lars R. Knudsen, "Two-Key Triple Encryption", Journal of Cryptology, vol.11 Number 3, p. 212, 1998.
[207] *ibid*, p. 212.
[208] William Stallings, *Network and Internetwork Security - Principles and Practice*, p. 67.
[209] Ivan B. Damgard and Lars R. Knudsen, "Two-Key Triple Encryption", Journal of Cryptology, vol.11 Number 3, p. 212, 1998.
[210] William Stallings, *Network and Internetwork Security - Principles and Practice*, pp. 67-69.
[211] Ivan B. Damgard and Lars R. Knudsen, "Two-Key Triple Encryption", Journal of Cryptology, vol.11 Number 3, p. 212, 1998.
[212] *ibid*, p. 212.
[213] Eli Biham, Lars R. Knudsen, "DES, Triple-DES and AES", RSA Laboratories' CryptoBytes, vol.4 Number 1, pp. 19-20, 1998.
[214] Bruce Schneier, *Applied Cryptography - Second Edition*, pp. 360-363.
[215] *ibid*, pp. 361-363.
[216] Eli Biham, Lars R. Knudsen, "DES, Triple-DES and AES", RSA Laboratories' CryptoBytes, vol.4 Number 1, pp. 19-21, 1998.

- **Triple-DES with Minimum Keys**

It should be noted that, for all the symmetric block ciphers, this variant is a subtype of generalized method known as Multiple Encryption with Minimum Keys[217] that is used for multiple key alternatives. Also, one of the well known types of multiple encryption with minimum keys where encryption is repeated three times is named as TEMK (Triple Encryption with Minimum Key)[218] which can also be applied to DES. And, a different variant of TEMK is known as 3-PEK[219], which is implemented for DES and these two types are also referred as Triple-DES with Minimum Keys.

The general scheme for multiple encryption with minimum keys is described as follows;

Given a block cipher X, with key-length k, the encryption under X is

$$C = E_K(P), \quad \text{with a key } K$$

and decryption process under cryptosystem X is

$$P = D_K(C).$$

A new cipher Y can be devised using a function G where;

$$G(K_1, K_2) = (X_1, X_2, X_3)$$

and this function maps two *X-keys* to three previous *X-keys*, where the keys in Y consist of pairs (K_1, K_2) of *X-keys*. Then the encryption process under Y is defined as;

$$E_{K_1,K_2}(P) = E_{X_3}\left(E_{X_2}\left(E_{X_1}(P)\right)\right).$$

where decryption can be achieved by using X_i' s in reverse order.[220] It' s proven that for all the types of attacks, Y is at least as secure and as hard to break as X.[221]

TEMK is defined shortly as a method for triple-encryptions with the minimum number of keys; which is achieved by deriving three keys K_1, K_2, K_3, from two initial keys, X_1 and X_2, namely.[222] This can be shown as below where T_1, T_2 and T_3 are initial vectors, or constants with random values which needn' t have to be kept secret[223];

---

[217] Ivan B. Damgard and Lars R. Knudsen, "Two-Key Triple Encryption", Journal of Cryptology, vol.11 Number 3, p. 213, 1998.
[218] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 360.
[219] Ivan B. Damgard and Lars R. Knudsen, "Two-Key Triple Encryption", Journal of Cryptology, vol.11 Number 3, p. 215, 1998.
[220] *ibid*, p. 213.
[221] *ibid*, pp. 213-215.
[222] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 360.
[223] *ibid*, p. 360.

$$K_1 = E_{X_1}\left(D_{X_2}\left(E_{X_1}(T_1)\right)\right)$$
$$K_2 = E_{X_1}\left(D_{X_2}\left(E_{X_1}(T_2)\right)\right)$$
$$K_3 = E_{X_1}\left(D_{X_2}\left(E_{X_1}(T_3)\right)\right)$$

So far, the best attack against this scheme is proven to be known-plaintext attack.[224]

3-PEK is a new method for triple encryption with pseudorandomly expanded keys similar to TEMK, and 3-PEK is implemented and used for DES.[225] The algorithm can be simply defined as follows, where the three keys that will be used in encryption are X_1, X_2, X_3, and the key length for each of the keys is k , which is 56-bits for DES and G is the mapping function;

$$G(K_1, K_2) = (X_1, X_2, X_3)$$

and the construction of G, hence the X_i' s are achieved by;

$$X_1 = E_{K_1}\left(E_{K_2}(W_1)\right)$$
$$X_2 = E_{K_1}\left(E_{K_2}(W_2)\right)$$
$$X_3 = E_{K_1}\left(E_{K_2}(W_3)\right)$$

It should be noted that W_i ' s are three different initial values that can be randomly generated with any method.[226] Several attacks have been applied to 3-PEK DES, and it has been proven that this method provides fairly well security with acceptable efficiency. The resistivity boundaries are shown in Table 2.9 in comparison with the previous methods. The best attack against 3-PEK DES is proven to be known-plaintext attack with 2^{55} computational complexity and 2^{43} known plaintexts.[227] It' s also shown that for DES, 3-PEK don' t produce any weak keys and related-key attack is not applicable, like the other triple encryptions with two key variants.[228]-[229]

As a general comment, it can be concluded that all of the triple-DES models produced so far have some weak or bad points somehow. But on the other hand, the financial services industry has developed ANSI X9.52, a standard for triple-DES as an interim solution.[230]-[231] But this standard is not related with NIST and do not imply a general standard approval for the replacement of 56-bit DES.

---

[224] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 360.
[225] Ivan B. Damgard and Lars R. Knudsen, "Two-Key Triple Encryption", Journal of Cryptology, vol.11 Number 3, pp. 215-216, 1998.
[226] *ibid*, p. 216.
[227] *ibid*, p. 216.
[228] *ibid*, pp. 216-217.
[229] John Kelsey, Bruce Schneier, David Wagner, "Key-Schedule Cryptanalysis of IDEA, G-DES, GOST, Safer and Triple-DES", Advances in Cryptology--CRYPTO '96 Proceedings, p. 249, 1996.
[230] Burt Kaliski, "Life After DES", RSA Data Security Inc., Internet Document, http://www.rsa.com, 1998.

Table 2.8 Comparison of several variants of DES Multiple Encryption.[232]

Number of Encryptions	Number of Keys	Computational Complexity	Required Storage (words of Memory)	Type of Attack
single	1	2^{56}	-	known plaintext
single	1	2^{38}	2^{38}	chosen plaintext
single	1	-	2^{56}	chosen plaintext
double	2	2^{112}	-	known plaintext
double	2	2^{56}	2^{56}	known plaintext *
double	2	-	2^{112}	chosen plaintext
triple	2	2^{112}	-	known plaintext
triple	2	2^{56}	2^{56}	2^{56} chosen plaintext **
triple	2	2^{120-t}	2^{t}	2^{t} known plaintext ***
triple	2	-	2^{56}	chosen plaintext
triple	3	2^{112}	2^{56}	known plaintext ****
triple	3	2^{56}	2^{112}	chosen plaintext

* *it was a meet-in-the-middle type of known plaintext attack.*
** *this attack was performed by Merkle and Hellman.*
*** *this attack was performed by Van Oorschot and Wiener where; for n plaintext / ciphertext pairs, 2^{120} / n, hence $2^{120-\log_2 n}$ computations needed; in other words, if n is denoted by 2^t, then computational complexity is 2^{120-t}.*
**** *it was a meet-in-the-middle type of known plaintext attack.*

Table 2.9 Bounds on the time complexities of Multiple Encryption modes for symmetric ciphers in general.[233]

Encryption mode	Key size *	Lower bound (all attacks)	Upper bound (best known attack)
Block cipher single	k	2^k	2^k
Two-key triple	$2k$	Unknown	2^k
Three-key triple	$3k$	2^k	2^{2k}
3-PEK **	$2k$	2^{k-1}	2^{2k}

* *For DES, $k = 56$.*
** *Triple encryption with minimum key.*

[231] Cryptography FAQ, Cryptosystems Journal, Internet Document, http://ourworld.compuserve.com/homepages/crypto/cryfaq05.htm, p. 4, 1998.
[232] RSA Data Security Inc, "Answers to FAQ About Today's Cryptography", RSA Laboratories paper, p. 72, 1996.
[233] Ivan B. Damgard and Lars R. Knudsen, "Two-Key Triple Encryption", Journal of Cryptology, vol.11 Number 3, p. 217, 1998.

2.4.5.2 Different DES Variants

Apart from the multiple encryption models, another alternative approach to the improvement of DES has been making some changes in the algorithm design or implementation as well as the key size. Various models have been proposed which some have been outdated and rejected while some still being questioned and tested and some others are being used for special purposes. The variants such as G-DES, DES with Alternate S-Boxes or Isolated-DES are proven to be insecure or infeasible. For instance, 2 x Isolated Double DES was proposed as an alternative to 56-bit DES, but it was proven to be highly susceptible to differential cryptanalytic attack, although proven to be sufficiently secure to other well-known cryptanalytic attacks.[234] Another model, designed by *Schaumuller-Bichl* was the Generalized DES (G-DES) which was aiming to speed up DES as well as improving its security. But later on, *Biham* and *Shamir* proved that G-DES was breakable easily and even less secure than 56-bit DES in some occasions.[235]-[236]

Some of the alternative DES models were proven to be fairly resistive to known cryptanalytic attacks and highly secure as well as promising efficient performances; but still they are not approved as a standard and not accepted as formal alternatives that would replace DES worldwide. Anyway, some of these DES variants will be mentioned shortly in the following paragraphs.

- **DES with Independent Subkeys**

An alternative DES variant proposed was DES with Independent Subkeys, based on the idea of using different independently generated subkeys for each of the 16 rounds of DES instead of generating them from the initial 56-bit key. In other words, since 48-bit subkeys are used in each round, this would imply the usage of a 48*16=768-bit key in total instead of 56 bits. Thus, the resistivity of the cipher is extremely strengthened up to 2^{768} computations against brute-force attack and to 2^{384} against meet-in-the-middle-attack. However, it' s proven that this model is not secure against linear and differential cryptanalysis as expected; where 2^{61} chosen plaintexts are required for a successful differential cryptanalysis and 2^{60} known plaintexts are enough for the linear cryptanalysis. Thus, this variant is not accepted efficient for a good alternative model and proven to be slightly more secure than 56-bit DES.[237]-[238]-[239]

---

[234] Terry Ritter, "2xIsolated DES: Another Weak Two-Level DES Structure", Ritter Software Engineering White Paper, pp. 2-4, February 16, 1994.

[235] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 296.

[236] RSA Data Security Inc, "Answers to FAQ About Today's Cryptography", RSA Laboratories paper, p. 74, 1996.

[237] *ibid*, p. 71.

[238] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 295.

[239] Cryptography FAQ, Cryptosystems Journal, Internet Document, http://ourworld.compuserve.com/homepages/crypto/cryfaq05.htm, p. 4, 1998.

- **DESX**

DESX is another DES variant developed by *RSA Data Security Inc.*[240] DESX algorithm is mostly based on a technique known as *whitening* which is added to the 56-bit DES structure. Whitening can be simply defined as a technique which some key values are XORed with the input to a block algorithm and some other key material is XORed with the output generated from the block algorithm.[241] The goal in the usage of whitening is to prevent a cryptanalyst from obtaining plaintext / ciphertext pairs from the target algorithm where the cryptanalyst is obliged to guess or break at least one of the whitening keys as well as the algorithm key.[242] Thus, applying this technique to DES, DESX is derived in consequence where two whitening keys each of 64 bits in length are used. The 64-bit key that is XORed with the 64-bit plaintext block before the first round of DES is named as pre-whitening key; and the additional 64-bit key which is XORed with the 64-bit ciphertext block after the last round of DES is known as post-whitening key; derived from the computation of a one-way function of the 120-bit key value (64-bit pre-whitening key plus original 56-bit DES key).[243]-[244]

The algorithm of DESX can also be simply denoted as below, where k_1 and k_2 are the pre and post-whitening keys respectively, K is the 56-bit standard DES key, E_K is DES encryption with key K and D_K is DES decryption with key K;

for each plaintext / ciphertext block i,

$$C_i = k_2 \oplus E_K\left(P_i \oplus k_1\right), \qquad \text{for encryption}$$

$$P_i = k_1 \oplus D_K\left(C_i \oplus k_2\right), \qquad \text{for decryption}$$

It's proven that DESX is much stronger than DES against brute force attacks since this attack needs $2^{120} / n$ operations with n known plaintexts.[245] It should be noted that in fact, the total key length used in DESX is 64+56+64=184 bits, but since the pre-whitening key is derived from the other two, cryptographically, the effective key space is taken as 120 bits in computations.[246]-[247] In addition to its high resistivity to various key search attacks, DESX is proven to be improving the security of 56-bit DES against differential and linear cryptanalysis; where the first attack requires 2^{61} chosen plaintexts and the second one needs 2^{60} known plaintexts.[248]-[249]-[250] It should be

---

[240] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 295.
[241] *ibid*, p. 366.
[242] *ibid*, p. 367.
[243] *ibid*, p. 295.
[244] Phillip Rogaway, "The Security of DESX", RSA Laboratories' CryptoBytes, vol.2 Number 2, p. 8, 1996.
[245] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 295.
[246] Phillip Rogaway, "The Security of DESX", RSA Laboratories' CryptoBytes, vol.2 Number 2, p. 8, 1996.
[247] Burt Kaliski, "Life After DES", RSA Data Security Inc., Internet Document, http://www.rsa.com, 1998.
[248] Phillip Rogaway, "The Security of DESX", RSA Laboratories' CryptoBytes, vol.2 Number 2, p. 11, 1996.
[249] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 295.

stressed that these values are exactly equivalent to that of DES with independent subkeys, but the security of DESX against brute force is also much greater than DES with independent subkeys as well as the standard DES.[251]

However, DESX is not accepted to be a very efficient alternative, since totally 184 bits and three keys are used, but the security strength is not as high as an expected outcome from that augmented key space. Also, the key size of DESX is claimed to be inconvenient among various applications.[252] DESX is still not approved as a standard algorithm and a formal alternative to DES in general, but due to its acceptable resistivity boundaries, it' s been in use commercially and in some special applications. For instance, *RSA Data Security Inc.* has included DESX in some of its security toolkits, such as MailSafe electronic mail security program and BSAFE toolkit.[253]-[254]

- **s^n DES**

Another DES variant was proposed by a group of Korean researchers led by *Kwangjo Kim* and the basic idea was to derive an alternative to DES with optimal security against both differential and linear cryptanalysis.[255] The former models such as s^2 and s^3 were proven to be even worse resistive than 56-bit DES against differential and linear cryptanalysis respectively. Later on, the upgrades s^4 and s^5 were proposed and proven to be highly secure against both differential and linear cryptanalysis. Rather than using s^n DES alone, mixing it with triple-DES is strongly suggested in order to produce a very strong algorithm against all types of cryptanalytic attacks.[256]

- **DES with Key-Dependent S-Boxes**

Another DES variant which focuses on the improvements in design and usage of S-boxes is DES with key-dependent S-boxes. The basic idea in this model is establishing key-dependent S-boxes and choosing them by a cryptographically strong method. This approach is proven to convert 56-bit DES into a much stronger cipher against both differential and linear cryptanalysis by adding secrecy to S-boxes themselves as well as keeping their randomly generated structure.[257] (It' s proven that knowing the composition and inner structure of S-boxes enable linear and differential cryptanalysis.)

It' s proven that the exhaustive search could succeed at least in 2^{102} computations to break this variant which makes it much more resistive than 56-bit DES. The computational complexity for a successful differential cryptanalytic attack to

---

[250] RSA Data Security Inc, "Answers to FAQ About Today's Cryptography", RSA Laboratories paper, p. 74, 1996.
[251] *ibid*, p. 74.
[252] Phillip Rogaway, "The Security of DESX", RSA Laboratories' CryptoBytes, vol.2 Number 2, p. 10, 1996.
[253] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 295.
[254] RSA Data Security Inc, "Answers to FAQ About Today's Cryptography", RSA Laboratories paper, p. 74, 1996.
[255] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 298.
[256] *ibid*, p. 298.
[257] *ibid*, p. 298.

this algorithm is calculated as 2^{51} and for a successful linear cryptanalysis, it' s found out to be 2^{53}.[258]

Considering the efficiency and feasibility, DES with Key-Dependent S-boxes is claimed to be favourable among the other variants. Because it can be implemented in existing hardware with chips having loadable S-boxes and also in software; in addition, its encryption performance and speed is shown to be no worse than 56-bit DES.[259] However, this model is still not used in common and not approved to be the standard alternative for 56-bit DES.

2.4.6 Future of DES

There are different approaches and comments about whether DES will be in existence in the future with the enhanced variants or will be completely replaced with the new algorithms. While some security experts suggest the new alternatives for DES such as DESX or triple-DES would be a sufficient choice for today and hopefully for the near future;[260] others recommend the usage of key-dependent S-boxes with *Biham*' s construction as a secure DES implementation instead of 56-bit DES.[261]

Meanwhile, in the recent years, a new alternative solution is being formulated. This solution is based on the replacement of DES entirely and building a new symmetric block cipher with larger key sizes and block sizes.[262] The suggestion of larger block sizes comes from the fact that, in today' s cryptology it' s strongly suggested not only the key length but also the block length of the plaintext / ciphertext pairs for each round in the encryption algorithm must be increased considerably to assure high level of security. This proposed idea has been put into a formal shape by the announcement of NIST in 1997 that an Advanced Encryption Standard (AES) will be developed. With this announcement, NIST intends to standardize a new encryption algorithm, AES, as a replacement with 56-bit DES.[263]-[264] So far, numerous candidate algorithms have been proposed to NIST and these are still being tested by an international group of experts. Some of these AES ciphers are; Cast-256, Crypton, DEAL, DFC, LOKI97, Mars, Magenta, RC6, SAFER+, Serpent, TWOFISH.[265] The common criteria for all these AES candidates are; the algorithm should support at least a block size of 128 bits and three key sizes of 128, 192 and 256 bits.[266] At the time of this thesis' writing, no AES candidate has been declared formally by NIST as the new standard, but researches are continually going on. However, it' s strongly expected that the AES process will result with a block cipher strongly resistive to all of the

---

[258] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 300.
[259] *ibid*, pp. 300-301.
[260] Burt Kaliski, "Life After DES", RSA Data Security Inc., Internet Document, http://www.rsa.com, 1998.
[261] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 301.
[262] Eli Biham, Lars R. Knudsen, "DES, Triple-DES and AES", RSA Laboratories' CryptoBytes, vol.4 Number 1, p. 21, 1998.
[263] *ibid*, p. 21.
[264] Burt Kaliski, "Life After DES", RSA Data Security Inc., Internet Document, http://www.rsa.com, 1998.
[265] AES, Internet Document, http://csrc.nist.gov/encryption/aes/aes_home.htm, 1999.
[266] Eli Biham, Lars R. Knudsen, "DES, Triple-DES and AES", RSA Laboratories' CryptoBytes, vol.4 Number 1, p. 21, 1998.

known cryptanalytic attacks, will be much more secure than 56-bit DES as well as providing feasibility and efficiency with only a slightly lower speed than DES.[267]-[268]

In conclusion, it can be stated that 56-bit DES won' t be used, and probably its variants won' t be accepted as the new standard in the future. It' s presumed that an algorithm entirely different from DES, such as an AES model, will be approved as the new symmetric block cipher standard and shall be served for worldwide use. However, it should not be forgotten that until that day, DES and its variants will be reluctantly and forcefully used and still be in existence carrying the dusty crown.

[267] Eli Biham, Lars R. Knudsen, "DES, Triple-DES and AES", RSA Laboratories' CryptoBytes, vol.4 Number 1, p. 21, 1998.

[268] Burt Kaliski, "Life After DES", RSA Data Security Inc., Internet Document, http://www.rsa.com, 1998.

Chapter 3

DIFFERENTIAL / LINEAR CRYPTANALYSIS

3.1 Differential Cryptanalysis

3.1.1 Introduction

A new method in cryptanalysis of symmetric block ciphers, the differential cryptanalysis was introduced in 1990 by *Eli Biham* and *Adi Shamir*. The differential cryptanalysis is in fact very complex and based on several mathematical properties which will be focused on in the following sections. This section will give a short overview of this attack as an introduction.

Differential cryptanalysis is simply defined as a type of chosen plaintext attack on iterative block cipher systems. (In fact, this method can be extended to known plaintexts, but not preferred due to performance considerations.) It' s a method which analyzes the effect of particular differences in plaintext pairs on the differences of the corresponding resultant ciphertext pairs. In other words, differential cryptanalysis analyzes the evolution of the differences of the plaintext input pairs as these pairs propagate through the rounds of DES (or any iterated block cipher) encrypted under the same initial key. Thereafter, these differences are used to assign probabilities to the possible key values and to deduce the most probable key which is aimed to be broken. For DES and many other similar symmetric block ciphers, the difference operation is chosen as XOR which might be another operation for different cipher models. The XOR outcomes of plaintext or ciphertext pairs are the differences of those pairs. By choosing random plaintext pairs with fixed difference values and using the differences coming from the resultant ciphertext pairs, different probabilities can be assigned to various key values and thus the attack can be carried out. It should be stressed that the real values of plaintext and ciphertext pairs are of no importance to the cryptanalyst but their difference values must satisfy particular conditions. The intruder only needs to have the chosen plaintext pairs in hand which give the required difference values and produce necessary ciphertext differences that enable the correct guess for the key bits.[1,2,3]

Recalling from Chapter 2, DES and most of the other symmetric block ciphers are based on applying several encryption functions and operations to the plaintext blocks in an iterated manner for specific number of rounds. Thus, these systems are also known as *iterated cryptosystems* which are accepted as a family of cryptographically strong functions based on iterating a weaker function n times. Each iteration is called as *round* and the cryptosystem is called an *n-round cryptosystem.* Thus, the differential cryptanalytic attacks are carried out amongst these iterated ciphers exploiting the characteristics of the round functions and operations.[4] Since DES is a 16-round iterated cipher, it' s also a type of iterated cryptosystem chosen as

---

[1] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, p. 11.
[2] Bruce Schneier, *Applied Cryptography - Second Edition*, pp. 285-286.
[3] Eli Biham, Adi Shamir, "Differential Cryptanalysis of DES-like Cryptosystems", The Weizmann Institute of Science - Department of Applied Mathematics, Research Paper, p. 8, 1990.
[4] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, p. 1.

the fundamental case study for the differential cryptanalysis. In iterated cryptosystems, *the round function* is the function of the output coming from the previous round and hence, the round function in DES is the *f function* which is explained in detail in Chapter 2, Section 2.4.2.

An important remark about the differential cryptanalysis should be made in general: The applicability of a differential cryptanalytic attack is determined by comparing the number of encryptions needed by the attack to the size of the key space and the size of the plaintext space. If the number of encryptions is larger than the size of the key space, the expected encryption time of the chosen plaintexts is proven to be larger than an exhaustive key search. Even worse, it' s also stated that if the number of encryptions is larger than the possible size of the plaintext space, the attack cannot be carried out at all.[5] In other words, it can be proven that an algorithm is resistant to differential cryptanalysis by showing that the amount of plaintext required to mount such an attack is greater than the amount of plaintext possible.[6]

When the applicability and implementations of differential cryptanalysis is in question, some additional points must be also noticed. First of all, this attack is proven to be largely theoretical so far. It' s shown that this attack' s extremely huge amount of time and data requirements made it beyond the reach of almost everyone. For instance, the necessary chosen plaintext data for 16-round DES can be gathered at least in three year' s time, even within an encryption speed of 1.5 megabits per second. Thus, if a cryptanalyst doesn' t have the complete chosen plaintext set already in hand, then he won' t be able to deal with any study involving differential cryptanalysis unless he has several spare years. A second issue is that when the attack is implemented with known plaintext data, things seem to get worse. The cryptanalyst has to choose among millions of plaintext / ciphertext pairs analyzing each and every pair, which brings a complexity of $2^{55.1}$ operations. Collecting that amount of data is an overhead alone, besides the analysis and computation overhead. Consequently, the differential cryptanalysis of 16-round DES with known plaintexts is not better than the brute-force attack in real life by no means, which makes it insensible to implement such an attack.[7]

Before going into details of the differential cryptanalysis, the theoretical basis of this attack is simply formulated and explained in this section so as to give a short outlook of the model.

For each round i of DES, if the chosen plaintext pairs are X and X^*, then the difference of these pairs is denoted as;

$$\Delta X = XOR\ (X, X^*)$$

Similarly, if the corresponding output ciphertext pairs are Y and Y^*, then the difference of these are;

$$\Delta Y = XOR\ (Y, Y^*)$$

---

[5] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, pp. 30-31.
[6] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 289.
[7] *ibid*, pp. 289-290.

Thus, for any round i, analyzing ΔX and ΔY for several characteristic values, the subkey K_i for that round is guessed with a calculated probability of p; in other words, ΔX may cause ΔY with a probability of p. For several differences and characteristics, these probability values are high and hence, the subkey guessed will be correct with a high probability. The relationship between ΔX , ΔY and K_i can be established due to the design of DES and f function. Since the expansion permutation and the P-box are known, the output difference ΔE from the expansion functions E(X) can be easily deduced. On the other hand, since the permutation operation after S-boxes and the output ciphertext difference ΔY are known, the input to the final permutation, thus the output difference value ΔS_O from S-boxes can also be found out easily. Finally, exploiting the properties of XOR operation the input difference to the S-boxes (ΔS_I) can be trivially obtained which is exactly equal to the ΔE, regardless of the value of K_i. Using the difference distribution tables for each S-box and checking out the derived ΔS_I and ΔS_O values, the most possible K_i values can be extracted.[8] Knowing the structure of DES round function f, and the properties posed by the very structure of f provides these deductions which is also simply shown in the Figure 3.1.

Due to the structure of the f function in DES, $\Delta E = \Delta S_I$ is a valid deduction and can be proven mathematically as follows;

Let E_1 and E_2 be the two expansion permutation outputs, and S_{I1} and S_{I2} be the two inputs for S-boxes derived from the plaintext pair and K_i be the subkey for any round i such that;

$$\Delta E = E_1 \oplus E_2 \qquad \text{and} \qquad \Delta S_I = S_{I1} \oplus S_{I2}$$

Prove that:

$$E_1 \oplus E_2 = S_{I1} \oplus S_{I2} \qquad \text{thus,} \quad \Delta E = \Delta S_I$$

Proof:

$$S_{I1} = E_1 \oplus K_i \qquad \text{(a)}$$
$$S_{I2} = E_2 \oplus K_i \qquad \text{(b)}$$

with the property of XOR, (a) and (b) can be rewritten as,

$$K_i = S_{I1} \oplus E_1$$
$$K_i = S_{I2} \oplus E_2$$

K_i can be discarded by equalizing the duo above and,

$$S_{I1} \oplus E_1 = S_{I2} \oplus E_2, \qquad \text{thus}$$
$$E_1 \oplus E_2 = S_{I1} \oplus S_{I2}$$

[8] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 286.

So it' s proven that $\Delta E = \Delta S_I$, for any round of DES regardless of any subkey value and this is valid for each and every round of DES.

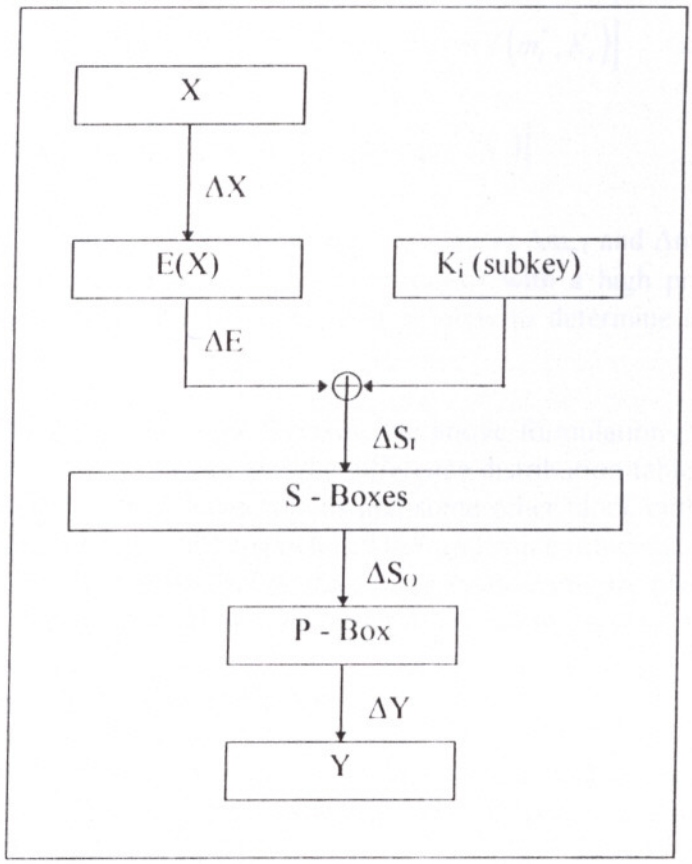


Figure 3.1 Differences throughout the DES round *(f)* function.[9]

The generalization of the differential cryptanalysis for all the 16 rounds of DES can be summarized as below;

For each initial plaintext block m, m_0 is the left half and m_1 is the right half for the first round. For each round, the right 32-bit block is processed through f function and XORed with the left half to form the new right half for the next round. Thus, if each new block is labelled as m_i, the interim data blocks through each round can be denoted as;

$$m_{i+1} = m_{i-1} \oplus f(m_i, K_i) \qquad \text{for } i = 1,..,16$$

K_i is the 48-bit subkey value for that round. In differential cryptanalysis, the difference of two chosen plaintext blocks is used which is $\Delta m = m \oplus m^*$. And throughout any round of DES, the difference value of interim data blocks is $\Delta m_i = m_i \oplus m_i^*$. Thus it can also be shown that;

[9] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 286.

$$\Delta m_{i+1} = m_{i+1} \oplus m^*_{i+1}$$
$$\Delta m_{i+1} = \left[m_{i-1} \oplus f\left(m_i, K_i\right)\right] \oplus \left[m^*_{i-1} \oplus f\left(m^*_i, K_i\right)\right]$$
$$\Delta m_{i+1} = \left[m_{i-1} \oplus m^*_{i-1}\right] \oplus \left[f\left(m_i, K_i\right) \oplus f\left(m^*_i, K_i\right)\right] \quad \text{so,}$$

$$\Delta m_{i+1} = \Delta m_{i-1} \oplus \left[f\left(m_i, K_i\right) \oplus f\left(m^*_i, K_i\right)\right]$$

As a conclusion, it can be stated that, as long as Δm_{i-1} and Δm_i are known with a high probability, then Δm_{i+1} can also be retrieved with a high probability. Hence, collecting a number of such differences, it is possible to determine the subkey K_i for each round.[10]

Exploiting the structure of S-boxes, the above formulations, the iterative and some other special characteristics and the difference distribution tables, the differential cryptanalysis of DES with reduced rounds and some other block ciphers are achieved successfully whereas for 16-round standard DES and some other strong ciphers, some necessary additional algorithms and mathematical mechanisms are generated and used. These will be explained in detail with examples in the following sections.

3.1.2 Definitions and The Basic Model

In this section, the basic model of the differential cryptanalysis technique will be analyzed in more detail within all the necessary terms, theorems, definitions, etc.

Before going into further details of the differential cryptanalysis, it' d be better to revise the fundamentals and the background which this attack is based on. As mentioned in the previous section, all the differential attacks focus mainly on the properties of the iterative round function used in the cryptographic algorithm. The *f* function of DES takes 32-bit input coming from the right half of the data block and after expanding it by the expansion function E(X) to 48 bits, it' s XORed with 48-bit subkey K_i. This produces the input to the S-boxes and after being processed in the S-boxes, the resultant 32-bit output is permuted by a P(X) operation in the P-box and the output is the 32-bit output of the *f* function which is then XORed with the left 32-bit half to form the new right half for the next round of DES. The basic idea in the differential cryptanalytic attack is to analyze the differential behaviour of this function. Given two plaintexts X and X*, the differential of this pair after the expansion function can be formulated as;

$$E(X) \oplus E(X^*) = E(X \oplus X^*)$$

For any data block pairs, the key doesn' t change the XOR value in the pair, due to the property of XOR;

$$\left(X \oplus K_i\right) \oplus \left(X^* \oplus K_i\right) = X \oplus X^*$$

[10] William Stallings, *Network and Internetwork Security - Principles and Practice*, p. 56.

Also for the outputs coming from the P-box permutation for each of the data pair X and X^*, their difference can be also rewritten as;

$$P(X) \oplus P(X^*) = P(X \oplus X^*).$$

In addition, for any round of DES, the outputs of the *f* function for the initial pair that will be XORed with their left halves for the next cycle, a linearity exists for the difference of the pairs such that;

$$(X \oplus Y) \oplus (X^* \oplus Y^*) = (X \oplus X^*) \oplus (Y \oplus Y^*)$$

Therefore it can be concluded that the difference of pairs is invariant in the key and is linear in the E(X) expansion, P(X) permutation and the XOR operation.[11]-[12] This provides an ease of use for the implementation of the attack; because rather than processing each of the chosen plaintext pair through separately and then measuring the differences of those afterwards, their difference value could be directly fed into the rounds of DES and the results could be delivered in less effort. This also explains the logic behind the method of differential cryptanalysis and provides its validity.

However, the S-boxes are proven to be non-linear. (In fact, they should be, considering the security of DES, explained in Chapter 2, Section 2.4.4.1.) Even knowing the difference value of the pair input to the S-boxes and the output difference coming from the S-boxes, ΔS_I and ΔS_O respectively, doesn't give any knowledge of the output values of each S_O. There can be numerous possible input pairs and output pairs which can produce the same difference values for each of the S-boxes. But, the frequency of the possible pairs, hence their probabilities are not uniformly distributed; thus making good comparisons and analyzing through the values, one can make a good guess or exactly find out the correct pair.[13]

- **Difference Distribution Tables**

In 56-bit standard DES algorithm, there are 8 S-boxes where each one has 64*64 possible input pairs and each of this pair has an input XOR and corresponding output XOR. There are 64*16 possible tuples of input and output XORs for each S-box since each S-box gets 6-bit input and produces 4-bit output which makes $2^6 * 2^4$ tuples. Thus, it can be concluded that each tuple results from 4 pairs in average. On the other hand, all the tuples do not exist as a result of a pair, and the existing ones do not suggest a uniform distribution either. This is issued as a very important fact for the differential cryptanalysis, and the usage of these distribution values and knowing the special properties of each of the S-box is crucial.[14] In a nutshell, due to the design of DES, given some input XOR, how many possible pairs could produce a given output XOR can be determined.

---

[11] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, p. 15.

[12] Alcourt, "Differential Cryptanalysis", Internet Document, http://www.execpc.com/~alcourt/desdoc.html, 1998.

[13] Eli Biham, Adi Shamir, "Differential Cryptanalysis of DES-like Cryptosystems", The Weizmann Institute of Science - Department of Applied Mathematics, Research Paper, p. 12, 1990.

[14] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, p. 16.

As mentioned in the several previous parts of this chapter, the difference distribution tables play an important role in the differential cryptanalysis and their usage is essential. The definition of the difference distribution tables is given below:

A table that shows the distribution of the input XORs and output XORs of all the possible pairs of an S box is called the difference distribution table of the S box. In this table each row corresponds to a particular input XOR, each column corresponds to a particular output XOR and the entries themselves count the number of possible pairs with such an input XOR and an output XOR.[15]

Since there are eight S-boxes in DES, there are eight difference distribution tables as well. The difference distribution table of S1 is given in Table 3.1 as an example. It could be analyzed from this table, for example; given that the input XOR is 0 (the two inputs are identical), a difference distribution table of S1 would show that there are 64 possible pairs for that input XOR and an output XOR of 0. With the usage of these distribution tables several theorems, methods and definitions have been constructed for differential cryptanalysis.

Considering the input and output XORs for any S-box, ΔS_I and ΔS_O named previously, a relationship is defined as follows;

Let X and Y be two values representing potential ΔS_I and ΔS_O respectively. It can be stated that *X may cause Y by the S-box* if there is a pair in which ΔS_I equals *X* and ΔS_O equals *Y*. If there' s such a pair, it could be written $X \rightarrow Y$, and if there' s no such pair it could be stated that *X may not cause Y by the S-box* and written as $X \not\rightarrow Y$.[16-17]

In other words, if X may cause Y by the S-box, then in the X/Y location of the table, some number greater than 0 should be observed. If however, X may not cause Y by the S-box, then 0 should be found in the X/Y location of the difference distribution table of the S-box. For instance, it could be seen from Table 3.1 that for S1, input XOR C_x (12 in hexadecimal notation) may not cause output XOR 2_x but may cause output XOR 3_x with eight different possible pairs.

It should also be noted that in order to have a useful knowledge, not only the difference distribution tables but also the numerical values of all possible input pairs for each and every S-box' s input/output XOR entries must be in hand. These are obtained after collecting many chosen plaintext samples and analyzing them. For example, possible input values for the input XOR S1 = 34_x by the output XOR (for all possible output XOR values that may be caused by that input XOR) are given in Table 3.2.

---

[15] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, p. 16.
[16] *ibid*, p. 18.
[17] Alcourt, "Differential Cryptanalysis", Internet Document, http://www.execpc.com/~alcourt/desdoc.html, 1998.

Table 3.1 The difference distribution table for the S-box S1.[18]

Input XOR	Output XOR															
	0_x	1_x	2_x	3_x	4_x	5_x	6_x	7_x	8_x	9_x	A_x	B_x	C_x	D_x	E_x	F_x
0_x	64	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1_x	0	0	0	6	0	2	4	4	0	10	12	4	10	6	2	4
2_x	0	0	0	8	0	4	4	4	0	6	8	6	12	6	4	2
3_x	14	4	2	2	10	6	4	2	6	4	4	0	2	2	2	0
4_x	0	0	0	6	0	10	10	6	0	4	6	4	2	8	6	2
5_x	4	8	6	2	2	4	4	2	0	4	4	0	12	2	4	6
6_x	0	4	2	4	8	2	6	2	8	4	4	2	4	2	0	12
7_x	2	4	10	4	0	4	8	4	2	4	8	2	2	2	4	4
8_x	0	0	0	12	0	8	8	4	0	6	2	8	8	2	2	4
9_x	10	2	4	0	2	4	6	0	2	2	8	0	10	0	2	12
A_x	0	8	6	2	2	8	6	0	6	4	6	0	4	0	2	10
B_x	2	4	0	10	2	2	4	0	2	6	2	6	6	4	2	12
C_x	0	0	0	8	0	6	6	0	0	6	6	4	6	6	14	2
D_x	6	6	4	8	4	8	2	6	0	6	4	6	0	2	0	2
E_x	0	4	8	8	6	6	4	0	6	6	4	0	0	4	0	8
F_x	2	0	2	4	4	6	4	2	4	8	2	2	2	6	8	8
10_x	0	0	0	0	0	0	2	14	0	6	6	12	4	6	8	6
11_x	6	8	2	4	6	4	8	6	4	0	6	6	0	4	0	0
12_x	0	8	4	2	6	6	4	6	6	4	2	6	6	0	4	0
13_x	2	4	4	6	2	0	4	6	2	0	6	8	4	6	4	6
14_x	0	8	8	0	10	0	4	2	8	2	2	4	4	8	4	0
15_x	0	4	6	4	2	2	4	10	6	2	0	10	0	4	6	4
16_x	0	8	10	8	0	2	2	6	10	2	0	2	0	6	2	6
17_x	4	4	6	0	10	6	0	2	4	4	4	6	6	6	2	0
18_x	0	6	6	0	8	4	2	2	2	4	6	8	6	6	2	2
19_x	2	6	2	4	0	8	4	6	10	4	0	4	2	8	4	0
$1A_x$	0	6	4	0	4	6	6	6	6	2	2	0	4	4	6	8
$1B_x$	4	4	2	4	10	6	6	4	6	2	2	4	2	2	4	2
$1C_x$	0	10	10	6	6	0	0	12	6	4	0	0	2	4	4	0
$1D_x$	4	2	4	0	8	0	0	2	10	0	2	6	6	6	14	0
$1E_x$	0	2	6	0	14	2	0	0	6	4	10	8	2	2	6	2
$1F_x$	2	4	10	6	2	2	2	8	6	8	0	0	0	4	6	4
20_x	0	0	0	10	0	12	8	2	0	6	4	4	4	2	0	12
21_x	0	4	2	4	4	8	10	0	4	4	10	0	4	0	2	8
22_x	10	4	6	2	2	8	2	2	2	2	6	0	4	0	4	10
23_x	0	4	4	8	0	2	6	0	6	6	2	10	2	4	0	10
24_x	12	0	0	2	2	2	2	0	14	14	2	0	2	6	2	4
25_x	6	4	4	12	4	4	4	10	2	2	2	0	4	2	2	2
26_x	0	0	4	10	10	10	2	4	0	4	6	4	4	4	2	0
27_x	10	4	2	0	2	4	2	0	4	8	0	4	8	8	4	4
28_x	12	2	2	8	2	6	12	0	0	2	6	0	4	0	6	2
29_x	4	2	2	10	0	2	4	0	0	14	10	2	4	6	0	4
$2A_x$	4	2	4	6	0	2	8	2	2	14	2	6	2	6	2	2
$2B_x$	12	2	2	2	4	6	6	2	0	2	6	2	6	0	8	4
$2C_x$	4	2	2	4	0	2	10	4	2	2	4	8	8	4	2	6
$2D_x$	6	2	6	2	8	4	4	4	2	4	0	6	0	8	2	6
$2E_x$	6	6	2	2	0	2	4	6	4	0	6	2	12	2	6	4
$2F_x$	2	2	2	2	2	6	8	8	2	4	4	6	8	2	4	2
30_x	0	4	6	0	12	6	2	2	8	2	4	4	6	2	2	4
31_x	4	8	2	10	2	2	2	2	6	0	0	2	2	4	10	8
32_x	4	2	6	4	4	2	2	4	6	6	4	8	2	2	8	0
33_x	4	4	6	2	10	8	4	2	4	0	2	2	4	6	2	4
34_x	0	8	16	6	2	0	0	0	14	4	6	8	0	2	14	0
35_x	2	2	4	0	8	0	0	0	4	4	6	8	0	2	4	0
36_x	2	6	2	2	8	0	2	2	4	2	6	8	6	4	10	0
37_x	2	2	12	4	2	4	4	10	4	4	2	6	0	2	2	4
38_x	0	6	2	2	2	0	2	2	4	6	4	4	4	6	10	10
39_x	6	2	2	4	12	6	4	8	4	0	2	4	2	4	4	0
$3A_x$	6	4	6	4	6	8	0	6	2	2	6	2	2	6	4	0
$3B_x$	2	6	4	0	0	2	4	6	4	6	8	6	4	4	6	2
$3C_x$	0	10	4	0	12	0	4	2	6	0	4	12	4	4	2	0
$3D_x$	0	8	6	2	2	6	0	8	4	4	0	4	0	12	4	4
$3E_x$	4	8	2	2	2	4	4	14	4	2	0	2	0	8	4	4
$3F_x$	4	8	4	2	4	0	2	4	4	2	4	8	8	6	2	2

Checking both the tables 3.1 and 3.2, some useful information is made available for the cryptanalyst. For instance, it could be seen from the Table 3.1 that when input XOR is 34_x there are only 8 possible output XOR' s, 1_x , 2_x, 3_x , 4_x , 7_x , 8_x , D_x , F_x . (Since for the other output XORs their entries are 0.) The total number of possible pairs are given in the each entry, and by checking the Table 3.2, all the possible values for those pairs can be observed. For instance, $34_x \rightarrow 3_x$ with possible input pairs 01_x,02_x; 15_x,21_x; 35_x,36_x . It should be noted these constitute three pairs at first glance,

[18] Eli Biham, Adi Shamir, "Differential Cryptanalysis of DES-like Cryptosystems", The Weizmann Institute of Science - Department of Applied Mathematics, Research Paper, p. 91, 1990.

but their duals are also counted as another pair, but not shown for the sake of simplicity. Thus, it makes up totally 6 pairs which is equal to the relevant entry in Table 3.1. This duality aspect is necessary since for the same $(01_x,02_x)$ input values, both input pairs $S1_I = 01_x \oplus S1_I^* = 02_x$ and $S1_I = 02_x \oplus S1_I^* = 01_x$ produce $\Delta S1_I = 34_x$; but the values of each input S1 differ in themselves which implies totally two distinct pairs dual of each other. This is important for the cryptanalyst, since a single (mathematically) pair yields two possible chosen plaintext pairs for the same input difference.

Table 3.2 Possible input values for the input XOR $(\Delta S1_I) = 34_x$ with all the output XORs it may cause (values in hexadecimal) [19]

Output XOR ($\Delta S1_O$)	Possible Inputs ($S1_I$)
1	03, 0F, 1E, 1F, 2A, 2B, 37, 3B
2	04, 05, 0E, 11, 12, 14, 1A, 1B, 20, 25, 26, 2E, 2F, 30, 31, 3A
3	01, 02, 15, 21, 35, 36
4	13, 27
7	00, 08, 0D, 17, 18, 1D, 23, 29, 2C, 34, 39, 3C
8	09, 0C, 19, 2D, 38, 3D
D	06, 10, 16, 1C, 22, 24, 28, 32
F	07, 0A, 0B, 33, 3E, 3F

It could easily be seen from the previous examples and tables 3.1 and 3.2 that for a fixed input XOR, the possible output XORs do not have a uniform distribution. Thus the former definition for the relationship of differences can be transformed into a new definition with a probabilistic property as follows;

It can be said that *X may cause Y with probability p by the S-box* if for a fraction p of the pairs in which ΔS_I equals X and ΔS_O equals Y.[20]

For example, $34_x \rightarrow 4_x$ provides only two pairs out of the 64 possible pairs of S1 with a probability $p = 1/32$ and similarly $34_x \rightarrow D_x$ posing a probability 1/8 with eight pairs out of 64.

The distributions, thus the probabilities are proven to be different among the S-boxes. In total, around 70-80% of the entries are possible ($p \neq 0$) ones and the rest 20-30% are impossible entries ($p = 0$, thus the entry value itself is 0 in the difference distribution tables). The exact percentage of the possible entries is given Table 3.3.

The usage of the distribution tables in differential cryptanalysis can be shown with the following example:

[19] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, p. 19.
[20] *ibid*, p. 18.

Table 3.3 Percentage of possible (non-zero) entries for the distribution tables of all the S-boxes.[21]

S-box	Percentage
S1	79.4%
S2	78.6%
S3	79.6%
S4	68.5%
S5	76.5%
S6	80.4%
S7	77.2%
S8	77.1%

Suppose that the expansion outputs for S1 of a chosen pair are E1 = 1_x and E1* = 35_x and the output XOR is $\Delta S1_O = D_x$. The subkey K' s 6-bit portion $S1_K$, which is XORed with the E1 values, is being searched.

$\Delta E1 = 1_x \oplus 35_x = 34_x$, and since $\Delta E1 = \Delta S1_I$, $\Delta S1_I = 34_x$. Since the input XOR is now in hand, by looking up the row for $34_x \rightarrow D_x$ in the Table 3.2, the eight possible pairs for the S1 inputs can be extracted. These eight possibilities make eight possibilities for the $S1_K$, because $S_K = E \oplus S_I$, due to the structure of f function. A new table can be constructed for all the possible $S1_K$ candidates by XORing either E1or E1* with each of the $S1_I$ value. In this new table, each line denotes two pairs with the same two inputs with the opposite order. Thus, each line leads to two possible key values making up 8 possible $S1_K$ values in total. The correct 6-bit subkey portion is exactly one of these eight values. Thus, so far, the cryptanalyst has a 1/8 hit chance of the correct $S1_K$ value only by a simple analysis and very few initial knowledge. The resultant table is given in Table 3.4.

Table 3.4 Eight possible key values for $34_x \rightarrow D_x$ by S1 (values in hexadecimal).[22]

Possible Inputs ($S1_I$)	Possible Keys ($S1_K$)
06 32	07 33
10 24	11 25
16 22	17 23
1C 28	1D 29

This example can be extended further with another chosen input pair so as to achieve a narrowed possible key space. Suppose that a new input pair is provided with E1 = 21_x , E1* = 15_x and $\Delta S1_O = 3_x$. Again, the input XOR $\Delta S1_I$ is 34_x and similarly by looking up the row for $34_x \rightarrow 3_x$ in the Table 3.2, six possible input pairs for S1 and the corresponding six possible key values can be extracted. These values are given in the Table 3.5. Thus, comparing the two possible key spaces, the correct key candidates can be chosen, which must be the ones that occur in both tables. The intersection key values are 17_x and 23_x , which the right key is one of these two and the correct key can be guessed by a 50% chance.

[21] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, p. 19.
[22] *ibid*, p. 20.

Table 3.5 Six possible key values for $34_x \rightarrow 3_x$ by S1 (values in hexadecimal) [23]

Possible Inputs ($S1_I$)	Possible Keys ($S1_K$)
01 35	20 14
02 36	**23 17**
15 21	34 00

It should be stressed that if a new input pair is used with a different XOR input value rather than 34_x, the exact correct key value can be detected among the possible values 17_x and 23_x.

The previous definitions of relationship between input and output XORs and probabilities can also be extended to a new definition for use with the f function as follows;

Let X and Y be two values that stand for potential input and output XOR values of the f function. It can be stated that *X may cause Y with probability p by the f function* if for a fraction p of all the possible input pairs encrypted by all the possible subkey values in which the input XOR of the f function equals X, the output XOR equals Y. If $p > 0$, this possibility is denoted by $X \rightarrow Y$.[24]

With the definition above, a theoretical assertion for DES, a lemma namely, is proposed as;

In DES, if $X \rightarrow Y$ with probability p by the f function then every fixed pair P, P^* with $P' = P \oplus P^* = X$ causes the f function output XOR to be Y by the same fraction p of the possible subkey values.[25]-[26]

It should be noted that this lemma is not exactly valid for the other iterated cryptosystems. On the other hand, among most of the other cryptosystems the fraction is found to be very close to p.[27]

The methods discussed and the examples given previously in this section that are involved with finding the key bits entering S-boxes can be extended to find the entire subkey entering the f function. This extension can also be defined as a generalized basic method for the differential cryptanalysis of DES which is given in the steps below;

1. Choose a suitable plaintext XOR, in other words, choose a specific difference value of any two plaintexts.
2. Create an appropriate number of plaintext pairs with the chosen plaintext XOR, encrypt them and only keep the resultant ciphertext pairs.

---

[23] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, p. 20.
[24] *ibid*, p. 21.
[25] *ibid*, p. 21.
[26] Alcourt, "Differential Cryptanalysis", Internet Document, http://www.execpc.com/~alcourt/desdoc.html, 1998.
[27] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, p. 21.

3. For each pair compute the expected output XOR of S-boxes in the last round from the plaintext XOR and the ciphertext pair as many as possible. (It should be remembered that the input pair of the last round is known because it is part of the ciphertext pair.)
4. For each possible key value, count the number of pairs that produce the expected output XOR using this key value in the last round.
5. The right, and presumably the unique, key value is the key value provided by all the pairs.[28]-[29]

- **Characteristics**

Until now, it' s mainly focused on the techniques and the basic methodology of the differential cryptanalysis for any single round. However, DES or any other iterated cryptosystem is constructed within several rounds which brings the necessity of having the knowledge of the XORs of the plaintext pairs as many rounds as possible without making them zero. This is essential because if the XORs of the pairs are 0, then this means that both chosen input texts are equal, thus the outputs are equal, which makes all the candidate keys with equal probability. As a result, the cryptanalyst is unable to choose or extract some keys or a single key among the whole key space. Because of this, an additional mechanism is required so as to provide a statistical characteristic of the cryptosystem which is an extension of the single round analysis.[30] This mechanism is named as *characteristic* and can be simply described as follows;

The related components of an encryption within any input pair are the XOR value of its two plaintexts, the XOR of its ciphertexts after the last round, the XORs of the inputs through each round in the two executions and the XORs of the outputs through each round in the two executions. These XOR values produce an *n-round characteristic*. A characteristic has a probability which is the probability that a random pair with the chosen plaintext XOR has the round and ciphertext XORs specified by that characteristic.[31]

It should be noted that the above informal definition is proposed for DES where for a more generalized definition including all the other iterated cryptosystems, difference can be used instead of the term XOR.

In general, the plaintext XOR of a characteristic is denoted by Ω_P and its ciphertext XOR by Ω_I. Also, Ω_P and Ω_I are expressed in two-tuple variables, where the left element of the two-tuple of Ω_P stands for the left 32-bit of the plaintext XOR and the right element represents the right 32-bit half of the plaintext XOR; similarly, the left element of the two-tuple of Ω_I is for the ciphertext XOR' s left 32-bit and the right element is the ciphertext XOR' s right 32-bit half. This notation is shown in the Figure 3.2 as a simple example for the characteristic. This figure denotes a one-round characteristic for DES with probability $p = 1$. In fact, this characteristic given in Figure

[28] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, pp. 21-22.
[29] Alcourt, "Differential Cryptanalysis", Internet Document, http://www.execpc.com/~alcourt/desdoc.html, 1998.
[30] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, p. 22.
[31] *ibid*, p. 22.

3.2 is proven to be the only and best one-round characteristic with $p > 1/4$. This characteristic is proven to be very useful and is also applicable to any other DES-like cryptosystem as well as DES.[32]

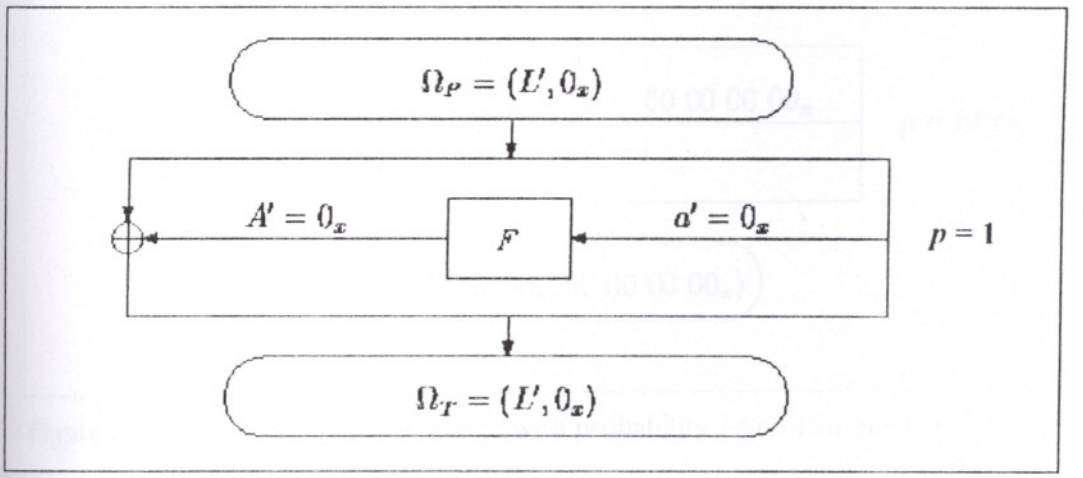


Figure 3.2 A one-round characteristic with probability 1 (for any L').[33]

It can be analyzed from the Figure 3.2 that the right 32-bit plaintext XOR is 0_x and no matter what the left plaintext XOR (L') is, the right 32-bit ciphertext XOR is produced as 0_x while preserving the left 32-bit ciphertext XOR value as L', every time, regardless of the value of L'. It should be recalled that, in DES, if a right half entering *f* function consists of all 0's, then the output of the function is purely 0 aftermath.

Another simple one-round character in DES with a probability 14/64 is given in Figure 3.3. In this one-round characteristic, the input XORs of seven S-boxes are zero, however the input XOR to S1 is non-zero and its input value is specifically chosen so as to maximize the probability that the input XOR may cause the output XOR. Thus, the input XOR, denoted by a', is $(60\ 00\ 00\ 00)_x$ which provides two bits to be 1 as input to S1. By this way, the best probability for S1, 14/64, could be achieved by assigning an entry with 14 pairs that cause the input of the other seven S-boxes to be zero. This can be given in a mathematical notation as;

$$S1: \quad 0C'_x \rightarrow E_x \quad \text{with } p = 14/64$$
$$S2, \ldots, S8: \quad 00_x \rightarrow 0_x \quad \text{with } p = 1$$

Thus, the probability of this one-round characteristic is 14/64 while producing the output XOR from the *f* function as $(00\ 80\ 82\ 00)_x$ which imposes only 3 non-zero bits for the output. It should be noted that the actual output from the S-boxes is $(E0\ 00\ 00\ 00)_x$, but after the P-box permutation, it becomes $(00\ 80\ 82\ 00)_x$. And, after XORed with the right half of the input XOR, Ω_T is derived as $L' \oplus (00\ 80\ 82\ 00)_x$ for the left half and $(60\ 00\ 00\ 00)_x$ for the right half. Like the previous example, this characteristic is true for any value of L', imposing the L'-invariant effect again.

---

[32] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, p. 22.

[33] Eli Biham, Adi Shamir, "Differential Cryptanalysis of DES-like Cryptosystems", The Weizmann Institute of Science - Department of Applied Mathematics, Research Paper, p. 20, 1990.

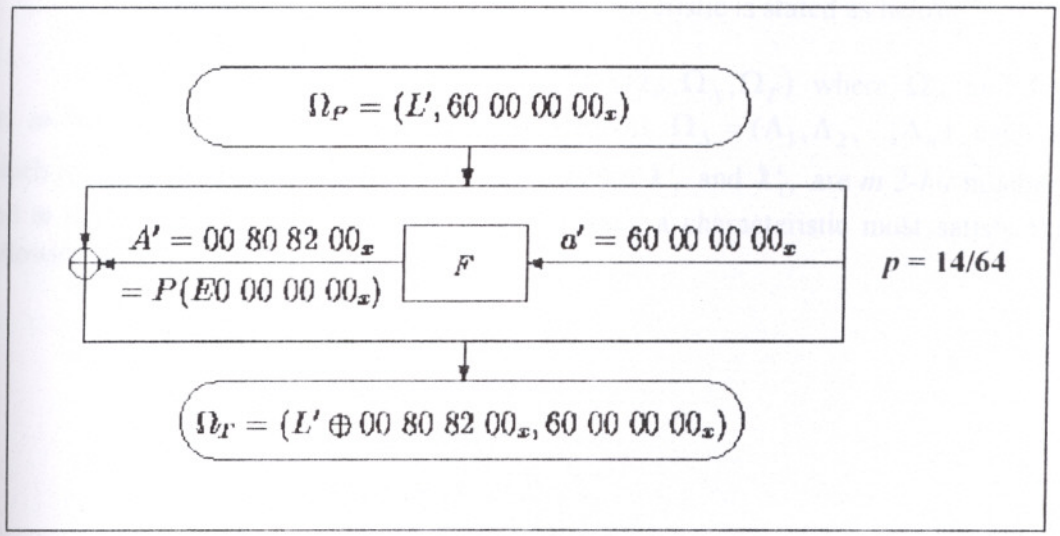


Figure 3.3 A one-round characteristic with probability 14/64 (for any L').[34]

Another simple characteristic case is given in the Figure 3.4, but this time a two-round characteristic with probability 14/64. This two-round characteristic' s probability can be obtained by concatenating the two one-round characteristics, hence multiplying their probabilities. As can be seen from the Figure 3.4, the first one-round characteristic' s probability p = 14/64 and the second one-round characteristic' s probability p = 14/64, thus the probability of their compound, two-round characteristic becomes p = 14/64 * 1 = 14/64. The theoretical aspects lying behind this solution will be given in the following paragraphs.

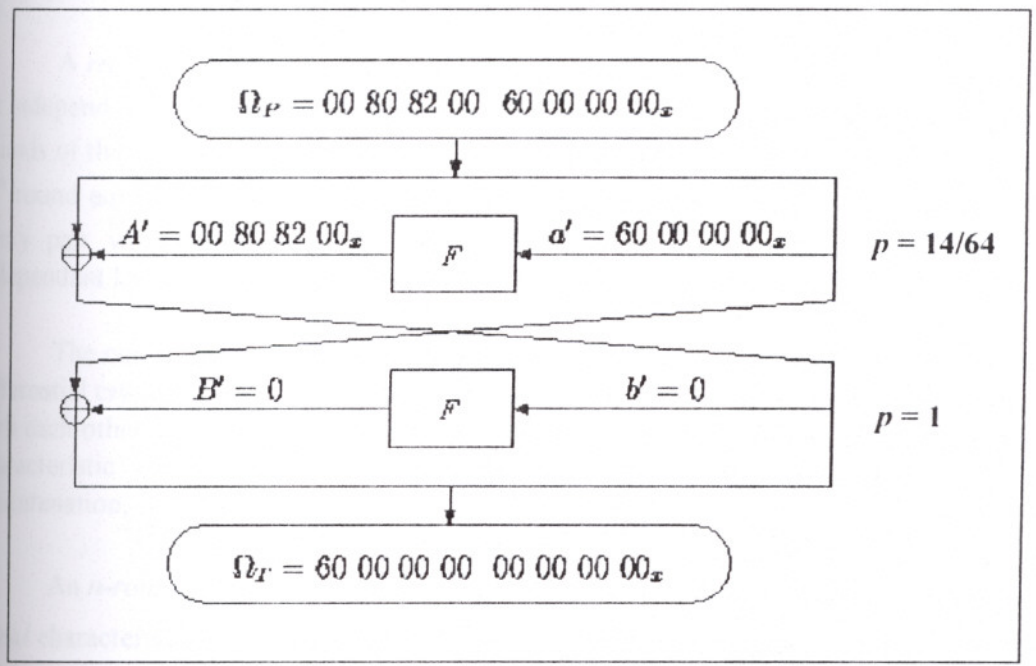


Figure 3.4 A two-round characteristic with probability 14/64.[35]

[34] Eli Biham, Adi Shamir, "Differential Cryptanalysis of DES-like Cryptosystems", The Weizmann Institute of Science - Department of Applied Mathematics, Research Paper, p. 21, 1990.
[35] *ibid*, p. 21.

The exact or the formal definition of a characteristic is stated as below;

An *n-round characteristic* is a tuple $\Omega = (\Omega_P, \Omega_\Lambda, \Omega_T)$ where Ω_P and Ω_T are *m-bit* numbers and Ω_Λ is a list of n elements $\Omega_\Lambda = (\Lambda_1, \Lambda_2, \ldots, \Lambda_n)$ each of which is a pair in the form of $\Lambda_i = (\lambda'_i, \lambda'_{i+1})$ where λ'_i and λ'_{i+1} are *m 2-bit* numbers and m is the block size of the cryptosystem. Then, a characteristic must satisfy the following criteria:[36]

λ^1_i = the right half of Ω_P

λ^2_i = the left half of $\Omega_P \oplus \lambda^1_{i+1}$

λ^n_i = the right half of Ω_T

λ^{n-1}_i = the left half of $\Omega_T \oplus \lambda^n_{i+1}$

and also requiring the condition that:

$$\lambda'_{i+1} = \lambda^{i-1}_i \oplus \lambda^{i+1}_i \qquad \text{for every } i \text{, where } 2 \le i \le n-1$$

In differential cryptanalysis, when dealing with n-round characteristics, there' s another term involved, right pair / wrong pair, namely. This can be simply described as; a plaintext pair that satisfies the characteristic is a referred as a *right pair* and a plaintext which does not satisfy is named as a *wrong pair*. Thus, a right pair implies the correct round key for the last round of that characteristic and a wrong pair suggests a random round key.[37] A more detailed and formal definition is stated as follows;

A *right pair* with respect to an n-round characteristic $\Omega = (\Omega_P, \Omega_\Lambda, \Omega_T)$ and an independent key K is a pair for which P' = Ω_P and for each round i of the first n rounds of the encryption of the pair using the independent key K, the input XOR of the i^{th} round equals λ'_i and the output XOR of the f function equals λ'_{i+1}. Conversely, every pair which is not a right pair with respect to the characteristic and the independent key is called a *wrong pair*.[38]

The concatenation of several different characteristics is another key point in the differential cryptanalysis, since usually different characteristics need to be concatenated with each other in order to reach the aimed number of rounds with a proper and useful characteristic value. The following definition is the formal description of the concatenation;

An *n-round* characteristic $\Omega^1 = (\Omega^1_P, \Omega^1_\Lambda, \Omega^1_T)$ can be concatenated with an *m-round* characteristic $\Omega^2 = (\Omega^2_P, \Omega^2_\Lambda, \Omega^2_T)$ if Ω^1_T equals the swapped value of the two halves of Ω^2_P. The *concatenation* of the characteristics Ω^1 and Ω^2 is the

[36] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, p. 24.
[37] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 287.
[38] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, p. 24.

characteristic $\Omega = (\Omega_P^1, \Omega_\Lambda, \Omega_T^2)$ where Ω_Λ is the concatenation of the lists Ω_Λ^1 and Ω_Λ^2.[39]

The probability of any characteristic is defined formally as follows;

Round i of characteristic Ω has probability p_i^Ω if $\lambda_I' \rightarrow \lambda_O'$ with probability p_i^Ω by the f function. Similarly, an *n-round* characteristic Ω has probability p^Ω, if p^Ω is the product of the probabilities of its n rounds. This can be denoted as:

$$p^\Omega = \prod_{i=1}^{n} p_i^\Omega \quad .[40]$$

This definition is also valid for the concatenation of several different characteristics' resultant probabilities. Therefore, it can be stated that the probability of a characteristic Ω, which is the concatenation of the characteristic Ω^1 with the characteristic Ω^2, is the product of their associated probabilities: $p^\Omega = p^{\Omega^1} * p^{\Omega^2}$.[41]

A theorem has also been provided which also deals with the probability of a characteristic and it' s as follows;

The formally defined probability of a characteristic $\Omega = (\Omega_P, \Omega_\Lambda, \Omega_T)$ *is the actual probability that any fixed plaintext pair satisfying* $P' = \Omega p$ *is a right pair when random independent keys are used.*[42]

It should be stressed that for the sake of simplicity and practicality, the particular probability associated with a characteristic is the probability that a pair (whose plaintext XOR is equivalent to the characteristic' s plaintext XOR) is accepted as a *right pair* using a fixed key that' s being searched. However, it' s proven that this probability is not constant for all the keys whereas it could be assumed that for randomly chosen key, it' s well approximated by the probability of the characteristic.[43]

Related with these concepts, some simple examples are given in the figures 3.5 and 3.6 which denote the concatenation of several characteristics and their probabilities. For instance, in Figure 3.5, a three-round characteristic is shown with the probability $\left(\frac{14}{64}\right)^2 \approx 0.05$. In fact, this characteristic is the concatenation of the characteristic given in the Figure 3.4 with the one in Figure 3.3. Since, the probability of the first characteristic (two-round) is 14/64, and the probability of the second one

[39] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, p. 24.
[40] *ibid*, p. 25.
[41] *ibid*, p. 25.
[42] *ibid*, p. 25.
[43] *ibid*, p. 25.

(one-round) is 14/64, their concatenated probability for the resultant three-round characteristic is the product of these two probabilities, which is $\frac{14}{64} * \frac{14}{64} = \left(\frac{14}{64}\right)^2$.

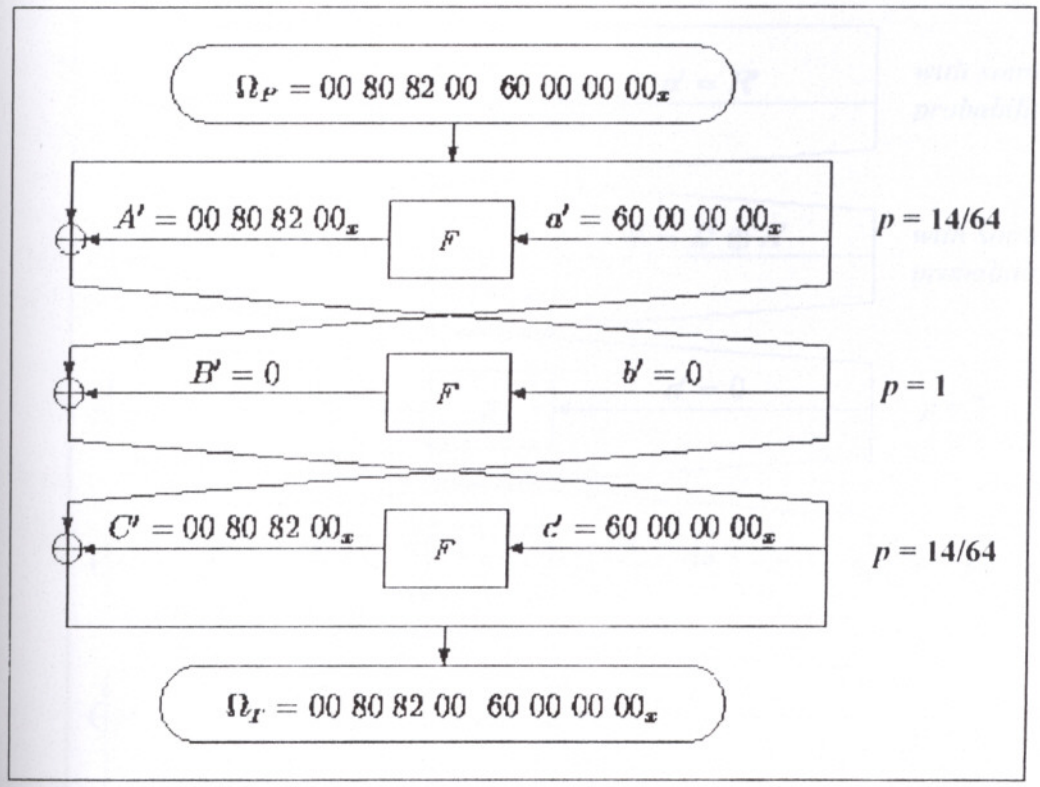


Figure 3.5 A three-round characteristic with probability $\left(\frac{14}{64}\right)^2$.[44]

It can be seen from the Figure 3.5 that when the input plaintext XOR differ in five bit locations, the output XOR from the *f* function in the third round differ in three bits and also, after the third round the ciphertext XOR output is exactly equal to the plaintext XOR. It should be noted that this three-round characteristic with a zero input XOR in the middle is proven to be having the best probability among all the known three-round characteristics.[45]

A similar structure can be constructed for a five-round characteristic where the middle round has 0 input/output XORs with a probability $p = 1$ and possessing a symmetry for the rounds around it. This five-round characteristic is shown in Figure 3.6 where the existence of the structure $b' \rightarrow a' \rightarrow A'$ within its rounds ensures the existence of such a five-round characteristic. In fact, this characteristic's probability is proven to be quite low due to the fact that three S-box inputs must differ in both rounds $b' \rightarrow a'$ and $a' \rightarrow A'$, and six bits in the entire five-round characteristic.

[44] Eli Biham, Adi Shamir, "Differential Cryptanalysis of DES-like Cryptosystems", The Weizmann Institute of Science - Department of Applied Mathematics, Research Paper, p. 24, 1990.
[45] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, p. 26.

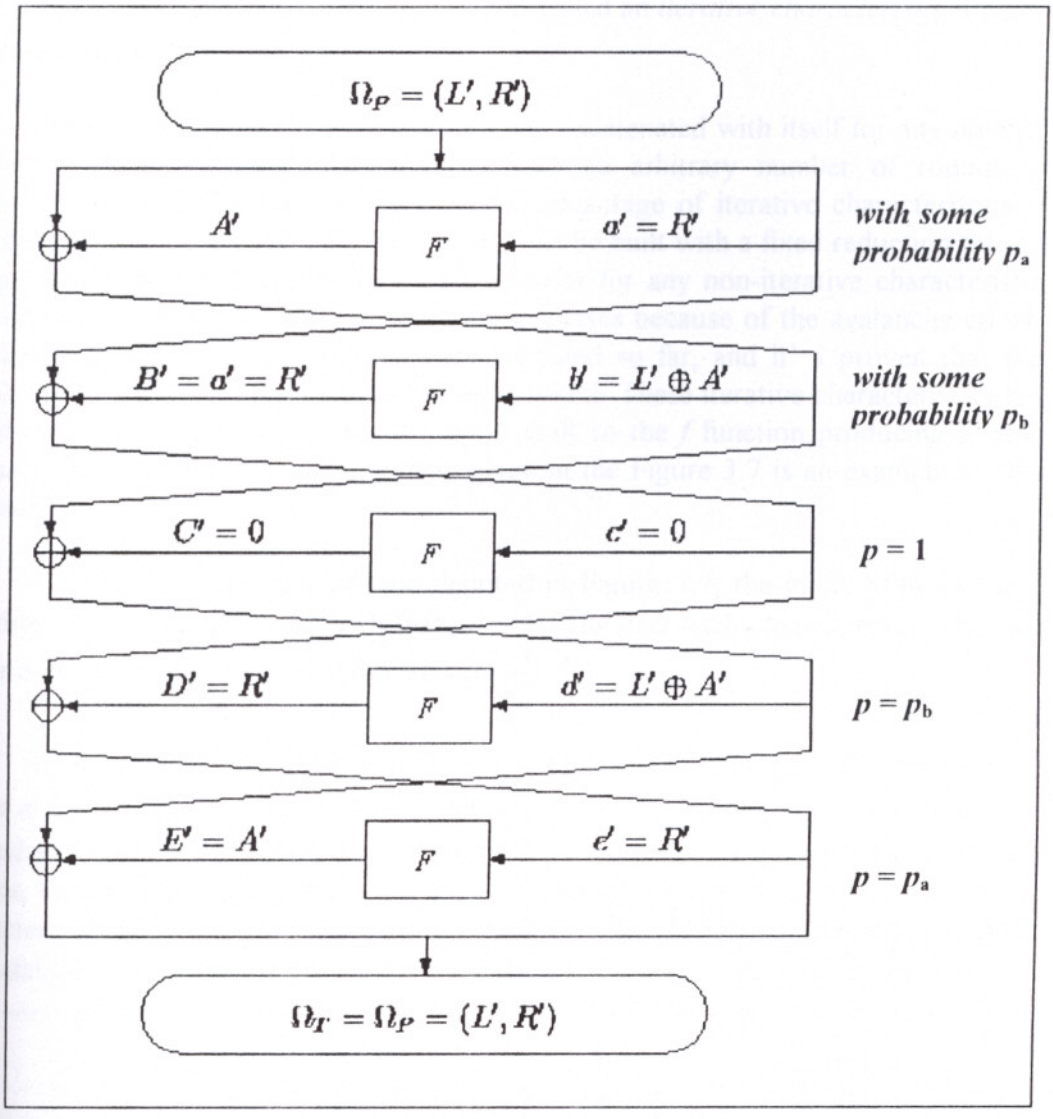


Figure 3.6 A five-round characteristic.[46]

Therefore the probability of the five-round is $(p_a)^2 * (p_b)^2$ in general, and since the best probability for an S-box is ¼, this limits the five-round characteristic's probability to be equal to or less than $\left(\frac{1}{4}\right)^6 = \frac{1}{4096}$. However, the best known five-round characteristic achieved so far has probability around $\frac{1}{10486}$.[47]

Besides these known best characteristics, there' s another type of characteristic which is proven to be mostly useful for the cryptanalyst, named as *iterative characteristic*. It can be simply defined as follows;

[46] Eli Biham, Adi Shamir, "Differential Cryptanalysis of DES-like Cryptosystems", The Weizmann Institute of Science - Department of Applied Mathematics, Research Paper, p. 25, 1990.
[47] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, p. 27.

A characteristic $\Omega = (\Omega_P, \Omega_\Lambda, \Omega_T)$ is called an *iterative characteristic* if it can be concatenated with itself.[48]

Thus, an iterative characteristic can be concatenated with itself for any number of times which produces characteristics with an arbitrary number of rounds in consequence. It' s claimed that the essential advantage of iterative characteristics is that an *n-round* characteristic for any large n can be built with a fixed reduction rate of the probability for each additional round whereas for any non-iterative characteristic the reduction rate of the probability usually increases because of the avalanche effect. There are several iterative characteristics provided so far, and it' s proven that the simplest ones (with less rounds) are the most useful. These iterative characteristics are based on a structure with a non-zero input XOR to the f function producing a zero output XOR. The iterative characteristic given in the Figure 3.7 is an example to this structure.

For the iterative characteristic denoted in Figure 3.7, the input XOR of the f function is abbreviated by ψ, such that $\psi \rightarrow 0$. The best such characteristic achieved so far is proven to have a probability around $\frac{1}{234}$.[49]

It can be seen from the Figure 3.7 that the derived output Ω_T after the second round and the final permutation is just the reverse of the Ω_P regarding the right and left halves. Thus, if these output right and left halves are to be input to any further rounds, they will be exactly the same input XORs of the first round of the iterative characteristic. (In fact, after the second round, $L' = \psi$ and $R' = 0$, but after the final permutation, they are swapped, hence $\Omega_T = (0, \psi)$). In other words, using this

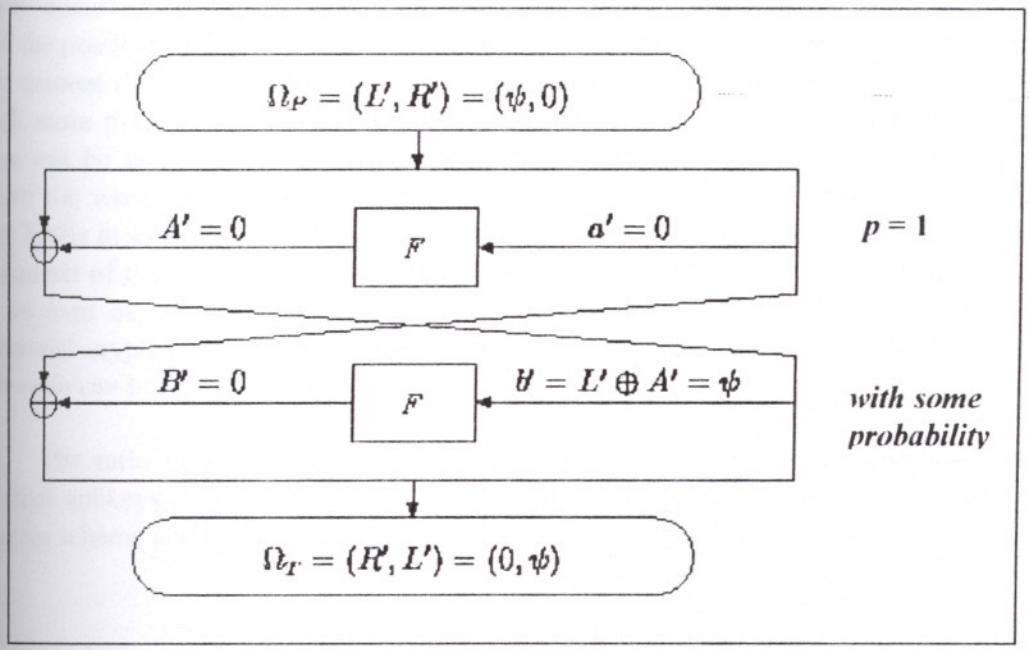


Figure 3.7 An iterative characteristic.[50]

[48] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, p. 27.
[49] *ibid*, p. 28.

two-round characteristic consecutively or embedding into any other characteristic, the extension can be achieved for any *n* rounds. This makes the specific property of the so-called iterative characteristic.

It should be stressed that several times throughout this section, while commenting on various characteristics and their related probabilities, "...the best value known so far..." remark is made. This is due to the fact these information is based on a few previous studies, manual calculations and some heuristic search programs derived by *Biham* and *Shamir*, et al. *Biham* and *Shamir* has made a very important and noticeable remark which says;

Although we believe that we have found the best DES characteristics, we have no proof that better characteristics do not exist.[51]

- **The Signal to Noise Ratio**

There's another term involved in characteristics and differential cryptanalytic attacks which is known as *the signal to noise ratio*. As mentioned previously, the statistical behaviour of most of the characteristics does not allow the cryptanalyst to look for the intersection of all the keys suggested by the various pairs. This is due to the fact that; when the characteristics are shorter than the cryptosystem, it' s impossible to identify the right pairs, hence, the intersection of the suggested key-space is null. However, it' s proven that the right key value can be extracted with the characteristic' s probability from right pairs plus the other random occurrences driven from the wrong pairs. Hence, counting the number of total occurrences of each of the suggested keys, the right key could be deduced which would be the key with the highest occurrence.[52] It should also be noted that counting the number of occurrences of all the possible values of a large number of bits of the key usually requires extremely large amount of memory. In order to decrease this memory requirement and make the attack more practical, a smaller number of subkey bits entering a less number of S-boxes can be counted, and all the other S-boxes can be used only to identify and discard the wrong pairs. These wrong pairs are proven to be the ones in which the input XORs in such S-boxes cannot cause the expected output XORs. In consequence, the number of the possible wrong pairs can be calculated as well as the right pairs and can be used as an extra knowledge in addition to the characteristics used for the differential cryptanalysis of any *n-round* scheme. The formal definition of signal to noise ratio can be stated as follows;

The ratio between the number of right pairs and the average count of the incorrect subkeys in a counting scheme is called *the signal to noise ratio* of the counting scheme and is abbreviated by *S/N*.[53]

---

[50] Eli Biham, Adi Shamir, "Differential Cryptanalysis of DES-like Cryptosystems", The Weizmann Institute of Science - Department of Applied Mathematics, Research Paper, p. 26, 1990.
[51] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, p. 28.
[52] *ibid*, p. 29.
[53] *ibid*, p. 30.

The S/N can be derived by the following formula:

$$\frac{S}{N} = \frac{2^k \cdot p}{\alpha \cdot \beta}$$

In the above formula, k is the number of key bits counted making 2^k possible key values, p is the probability of the characteristic, α is the average count per analyzed pair and β is the fraction of the analyzed pairs among all the pairs.[54]

It' s been observed that when S/N is high enough, only a few occurrences of right pairs are required to identify the right value of the subkey bits uniquely. For instance, when the signal to noise ratio is around 1-2, nearly 20-40 occurrences of right pairs are proven to be sufficient. When the signal to noise ratio is much higher, only 3 or 4 pairs are shown to be sufficient enough, conversely, when this ratio is much smaller than 1, then an extremely large number of pairs are required to identify the right value of the subkey bits.[55]

3.1.3 Differential Cryptanalysis with Known Plaintexts

The differential cryptanalysis is mostly carried out with chosen plaintexts, but it can be extended to known plaintexts for any iterated block cryptosystem. However, it' s usually observed that the performance of differential cryptanalysis with the known plaintexts decreases significantly when compared to chosen plaintexts. But, whenever necessary, any differential cryptanalytic attack to any cipher model with chosen plaintext can be easily converted to known plaintext, a property which is proven not to be valid for some other cryptanalytic attacks.

The conversion to known plaintext attack for DES is simply explained here which can be similarly applied to any other symmetric block cipher. It can be assumed that for a differential cryptanalytic chosen plaintext attack requiring m pairs, it' s given that $2^{32} \cdot \sqrt{2m}$ random known plaintexts and their corresponding ciphertexts. Thus, these can form at most $\frac{\left(2^{32} \cdot \sqrt{2m}\right)^2}{2} = 2^{64} \cdot m$ possible pairs of plaintexts. Each pair's plaintext difference (XOR) value can also be calculated trivially. Since, the block size is 64 bits in DES, there will be 2^{64} possible plaintext XOR values, implying that there are about $\frac{2^{64} \cdot m}{2^{64}} = m$ possible pairs associated with each plaintext XOR value. Therefore, it can be concluded that with high probability, there are about m pairs with each one of the several plaintext XOR values needed for the differential cryptanalysis.[56] It should be also stressed that as well as ECB mode, differential cryptanalysis with known plaintexts is applicable to CBC, CFB and OFB mode which is also valid for the differential cryptanalysis with chosen plaintexts having similar computational complexities.[57-58-59]

---

[54] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, p. 30.
[55] *ibid*, p. 30.
[56] *ibid*, p. 31.
[57] *ibid*, p. 31.

The S/N can be derived by the following formula:

$$\frac{S}{N} = \frac{2^k \cdot p}{\alpha \cdot \beta}$$

In the above formula, k is the number of key bits counted making 2^k possible key values, p is the probability of the characteristic, α is the average count per analyzed pair and β is the fraction of the analyzed pairs among all the pairs.[54]

It' s been observed that when S/N is high enough, only a few occurrences of right pairs are required to identify the right value of the subkey bits uniquely. For instance, when the signal to noise ratio is around 1-2, nearly 20-40 occurrences of right pairs are proven to be sufficient. When the signal to noise ratio is much higher, only 3 or 4 pairs are shown to be sufficient enough, conversely, when this ratio is much smaller than 1, then an extremely large number of pairs are required to identify the right value of the subkey bits.[55]

3.1.3 Differential Cryptanalysis with Known Plaintexts

The differential cryptanalysis is mostly carried out with chosen plaintexts, but it can be extended to known plaintexts for any iterated block cryptosystem. However, it' s usually observed that the performance of differential cryptanalysis with the known plaintexts decreases significantly when compared to chosen plaintexts. But, whenever necessary, any differential cryptanalytic attack to any cipher model with chosen plaintext can be easily converted to known plaintext, a property which is proven not to be valid for some other cryptanalytic attacks.

The conversion to known plaintext attack for DES is simply explained here which can be similarly applied to any other symmetric block cipher. It can be assumed that for a differential cryptanalytic chosen plaintext attack requiring m pairs, it' s given that $2^{32} \cdot \sqrt{2m}$ random known plaintexts and their corresponding ciphertexts. Thus, these can form at most $\frac{\left(2^{32} \cdot \sqrt{2m}\right)^2}{2} = 2^{64} \cdot m$ possible pairs of plaintexts. Each pair's plaintext difference (XOR) value can also be calculated trivially. Since, the block size is 64 bits in DES, there will be 2^{64} possible plaintext XOR values, implying that there are about $\frac{2^{64} \cdot m}{2^{64}} = m$ possible pairs associated with each plaintext XOR value. Therefore, it can be concluded that with high probability, there are about m pairs with each one of the several plaintext XOR values needed for the differential cryptanalysis.[56] It should be also stressed that as well as ECB mode, differential cryptanalysis with known plaintexts is applicable to CBC, CFB and OFB mode which is also valid for the differential cryptanalysis with chosen plaintexts having similar computational complexities.[57-58-59]

---

[54] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, p. 30.
[55] *ibid*, p. 30.
[56] *ibid*, p. 31.
[57] *ibid*, p. 31.

The performance degradation is the only problem of differential cryptanalysis with known plaintexts. For instance, differential cryptanalysis of 16-round standard DES requires 2^{47} chosen plaintexts, but 2^{55} known plaintexts. However, for the differential cryptanalysis of 16-round DES with independent subkeys, the performance is not so significantly different where 2^{60} chosen plaintexts and 2^{61} known plaintexts are required.[60] The performance of differential cryptanalysis with chosen and known plaintexts can be comparatively analyzed from the tables 3.6 and 3.7.

Table 3.6 The differential cryptanalysis of FEAL for different rounds with the required chosen plaintexts and known plaintexts.[61]

No. of Rounds	Chosen Plaintexts	Known Plaintexts
4	2^3	2^{34}
8	2^7	2^{36}
12	2^{21}	2^{42}
16	2^{29}	2^{46}
20	2^{37}	2^{50}
24	2^{45}	2^{54}
28	2^{56}	2^{60}
30	2^{60}	2^{62}
31	2^{63}	2^{63}

Table 3.7 The differential cryptanalysis of DES for different rounds with the required chosen plaintexts and known plaintexts.[62]

	Dependent Key		**Independent Key**	
No. of Rounds	Chosen Plaintexts	Known Plaintexts	Chosen Plaintexts	Known Plaintexts
4	2^3	2^{33}	2^4	2^{33}
6	2^8	2^{36}	2^8	2^{36}
8	2^{14}	2^{38}	2^{16}	2^{40}
9	2^{24}	2^{44}	2^{26}	2^{45}
10	2^{24}	2^{43}	2^{35}	2^{49}
11	2^{31}	2^{47}	2^{36}	2^{50}
12	2^{31}	2^{47}	2^{43}	2^{53}
13	2^{39}	2^{52}	2^{44}	2^{54}
14	2^{39}	2^{51}	2^{51}	2^{57}
15	2^{47}	2^{56}	2^{52}	2^{58}
16	2^{47}	2^{55}	2^{60}	2^{61}

[58] Eli Biham, "Cryptanalysis of Multiple Modes of Operation", Journal of Cryptology, vol.11 Number 1, p. 47, 1998.

[59] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 289.

[60] Eli Biham, "Cryptanalysis of Multiple Modes of Operation", Journal of Cryptology, vol.11 Number 1, p. 47, 1998.

[61] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, p. 9.

[62] *ibid*, p. 8.

$c' = a' \oplus B'$, thus

$c' = B'$, since $a' = 0$.

Thus, for the output difference C', about six bits will be different, or 6 bits with the value 1. These bits will be XORed with the known difference of the input of S1 from the second round, b', namely, and a difference of seven bits will be produced aftermath which is also the input for the fourth round and final round. All these can also be analyzed from the Figure 3.9. After the f function of the fourth round, a total of 11 bits will differ, and hence, there will be eleven 1's in the ciphertext differences after the fourth round. This poses the fact that such an avalanche ensures the input of all the S-boxes differ at the fourth round, providing the knowledge that the exact value of d' is usually different for every initial plaintext pair.[65]

The former information also suggests that the total 28 output XOR bits of the S-boxes S2, .., S8 for the second round must be zero, since their corresponding inputs are 0. Hence, the value of D' could be achieved from a', B' and T_L' as below;

$$D' = a' \oplus B' \oplus T_L' \qquad \text{(a)}$$

It should be noted that T_L' is the left half of the output XORs of the ciphertext T' which is $T' = T \oplus T^*$. When the ciphertext pairs T, T^* are known, then d and d^* are also known, since the input pairs to the fourth round are also the right halves of the ciphertext pairs ($d = T_R$ and $d^* = T_R^*$). By the equation (a), the 28 bits of D' are also known, which imply the 28 output bits of the seven S-boxes S2, .., S8. As a result, the 28-bit portion of the expansion permutation outputs (S_{Ed}, S^*_{Ed}) that are XORed with the subkey (S_{Kd}) , and the output difference (S_{Od}') of the seven S-boxes S2, .., S8 of the fourth round will be in hand. Having granted all this knowledge, all the 64 possible values for the subkey S_{Kd} can be tried and checked for the right one using the following equation;[66]

$$S(S_{Ed} \oplus S_{Kd}) \oplus S(S^*_{Ed} \oplus S_{Kd}) = S'_{Od}$$

The above equation is valid, because;

$$S_{Id} = S_{Ed} \oplus S_{Kd}$$
$$S^*_{Id} = S^*_{Ed} \oplus S_{Kd}$$

and since,

$$S'_{Od} = S_{Od} \oplus S^*_{Od}$$

and also,

---

[65] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, p. 34.
[66] *ibid*, p. 35.

$$S(S_{Id}^{*}) = S_{Od}^{*} \qquad S(S_{Id}) = S_{Od}$$

therefore,

$$S(S_{Ed} \oplus S_{Kd}) \oplus S(S_{Ed}^{*} \oplus S_{Kd}) = S_{Od}' .$$

The right subkey value will be the one suggested by all the pairs, since the probability of the characteristic Ω^1 is 1. In fact, this right subkey value of the fourth round is the 42-bit portion of the entire 56-bit key value, since this method is applied to seven of the eight S-boxes making a total of 7*6 = 42 bits for the key value. However, the remaining 14 bits can be deduced by trying all the 2^{14} possibilities. This is simply achieved by holding constant the 42-bit portion and deriving all the key combinations for the remaining 14-bit part, and decrypting the given ciphertexts by each of this subkey value and comparing the results with the original plaintexts. The 56-bit right key will be the one that decrypts all the pairs correctly. Thus, the 56-bit subkey of the fourth round, and also the initial original key for 4-round DES will be broken with a few effort.

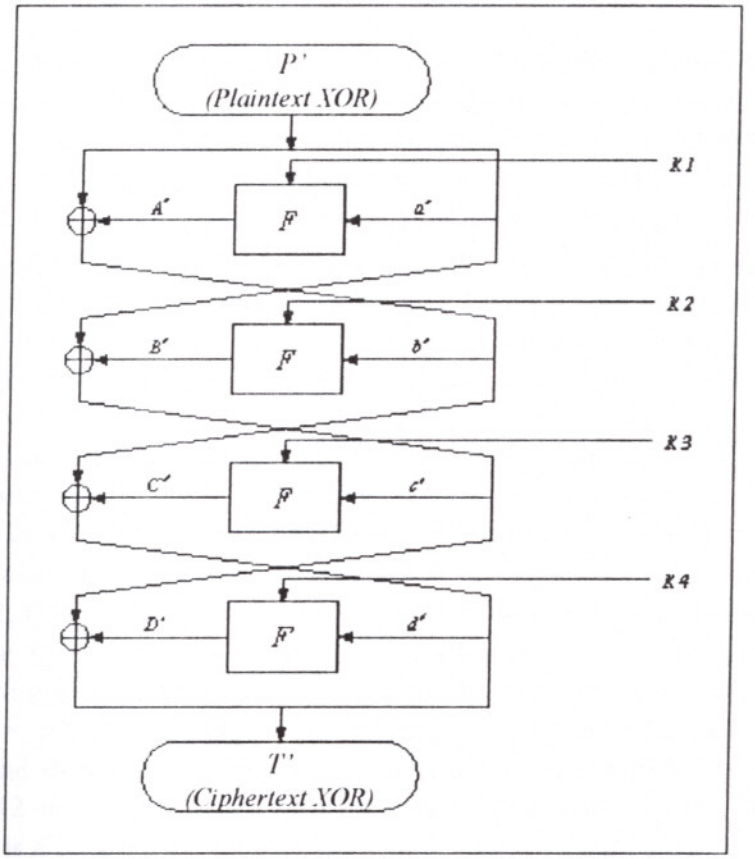


Figure 3.9 DES reduced to four rounds.[67]

[67] Eli Biham, Adi Shamir, "Differential Cryptanalysis of DES-like Cryptosystems", The Weizmann Institute of Science - Department of Applied Mathematics, Research Paper, p. 31, 1990.

As a result, it was found out that the differential cryptanalysis of DES reduced to four rounds can be achieved successfully with 2^3 chosen plaintext pairs and 2^{33} known plaintext pairs.[68]

3.1.4.2 DES Reduced to Six Rounds

In the previous DES model, a single-round characteristic with a probability of 1 was used; but for the differential cryptanalysis of six-round DES, two different three-round characteristics each with a probability of 1/16 are used. Each one of these characteristics is concatenated with additional three rounds of DES (this extension with the additional three rounds is known as *3R-attack* which will be explained later on) to form a six-round characteristic, and the outcomes of these six-round characteristics are analyzed and compared. This attack is proven to be somehow more complex than the previous one, and an additional counting scheme is required in order to deduce the key bits. Each of the two characteristics enable the cryptanalyst to find out the 30 bits of the subkey in the sixth round, but since three S-boxes are common in both, the actual number of key bits extracted turn out to be 42 in total. The remaining 14 bits can be calculated by the exhaustive search method, as implemented in the four-round DES, or by a new scheme which carefully counts the key bits entering S8 in the sixth round.[69]

The first three-round characteristic Ω^1 has a probability 1/16, which is also given in the Figure 3.10. When an additional three-round is placed after this characteristic, five S-boxes (S2, S5, S6, S7, S8) in the fourth round is analyzed as having zero input XORs ($S'_{Ed} = 0$) and consequently zero output XORs ($S'_{Od} = 0$). The output XORs in the sixth round can be derived by $F' = c' \oplus D' \oplus T'_L$.[70] It should be noted that the output XORs for the first three rounds, are A', B', C', as shown in the Figure 3.10. And similarly, for the other additional three rounds, the fourth, fifth and sixth output XORs are abbreviated as D', E', F' and their corresponding input XORs as d', e', f', where the input XOR in the fourth round is $d' = b' \oplus C' = (40\ 08\ 00\ 00)_x$.

Using the similar assumptions used in four-round analysis, the 30 bits of the subkey of the sixth round relevant to the five S-boxes mentioned previously could be deducted with a feasible probability. The same mechanism is implemented to another three-round characteristic, Ω^2 , to derive the 30-bit of the subkey associated with the S-boxes (S1, S2, S4, S5, S6). As similar to the first case with Ω^1, five boxes in the fourth round are proven to have zero input XORs after concatenating Ω^2 with additional three rounds. (As similar to Ω^1, the input XOR in the fourth round becomes $d' = b' \oplus C' = (00\ 20\ 00\ 08)_x$). Combining the 30-bits derived from both characteristics, and extracting the bits associated with the common S-boxes (S2, S5, S6) a total of 42 bits is extracted from subkey of the sixth round. The second characteristic Ω^2 is also given in Figure 3.11.

---

[68] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, p. 8, 53.
[69] *ibid*, p. 37.
[70] *ibid*, p. 37.

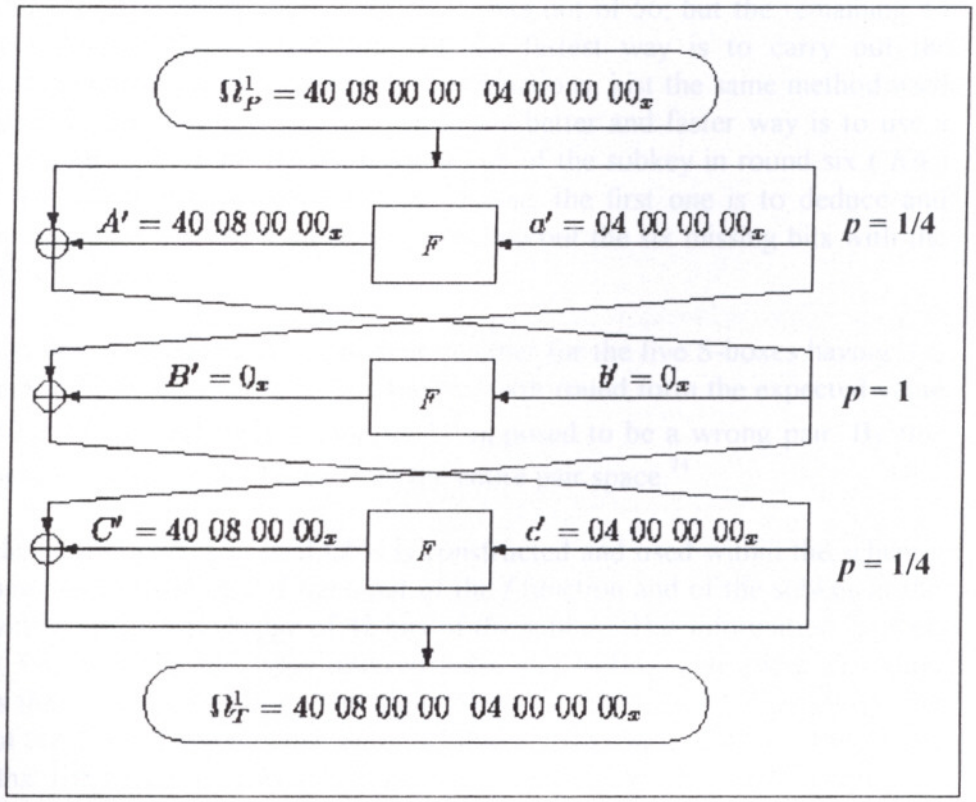


Figure 3.10 The first characteristic Ω^1 with $p = 1/16$ of the six-round DES.[71]

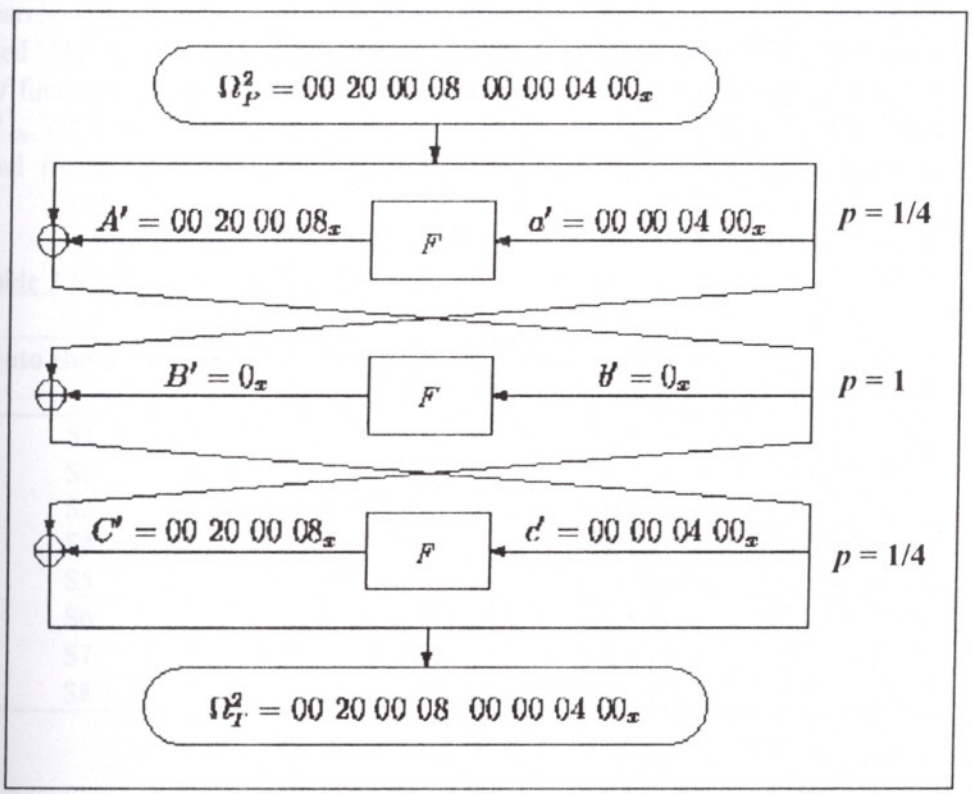


Figure 3.11 The second characteristic Ω^2 with $p = 1/16$ of the six-round DES.[72]

[71] Eli Biham, Adi Shamir, "Differential Cryptanalysis of DES-like Cryptosystems", The Weizmann Institute of Science - Department of Applied Mathematics, Research Paper, p. 35, 1990.

By this way, the cryptanalyst can find 42 bits out of 56; but the remaining 14 bits are still unknown. The easiest, but not the fastest way is to carry out the exhaustive search among all the possible 2^{14} combinations, just the same method used in DES reduced to four rounds. It' s proven that a better and faster way is to use a scheme that enables looking for the six missing bits of the subkey in round six (*K6*) which enter S3. There are two parts in this scheme, the first one is to deduce and discard the wrong pairs and the second part is to find out the six missing bits with the help of extracted right pairs.[73]

For the first part, each pair is checked whether for the five S-boxes having $S'_{Ed} = 0$, the output difference from the S-boxes at the sixth round form the expected value from $F' = c' \oplus D' \oplus T'_L$ or not. If not, this pair is supposed to be a wrong pair. By this way, all the wrong pairs can be filtered from the entire pair space.[74]

For the second part, a special table is constructed and used within the scheme. This table shows the known bits of the input of the *f* function and of the subkey at the fifth round with the prior knowledge of 42 bits of the subkey. This information is given in the Table 3.8, which is the table referred here, and in this table there are some abbreviations that need explanation. The digit '3' implies that the bit depends on the exact value of the missing key bits that enter S3 in the sixth round. On the other hand, '+' denotes that the sought bit depends only on known key bits. The eight key bits of the subkey *K6* which are excluded in this part are abbreviated as '•' In other words, '•' marks are the remaining eight key bits which totally make up 14, with the six missing bits being searched. Thus, using this table, the correctness of the guessed six bits of *K6* can be verified. This is achieved by calculating input bits (denoted as *e* bits in the Table 3.8) to the *f* function in the fifth round for each right pair and by verifying that the values of $S2'_{Oe}$, $S3'_{Oe}$ and $S8'_{Oe}$ (the output XORs from the S-boxes S2, S3 and S8 at the fifth round for which all the input and key bits are known) are as expected by $E' = d' \oplus f'$.

Table 3.8 Known bits at the fifth round.[75]

Into the S-box	*e* bits (S_{Ee})	Key bits (S_{Ke})
S1	+ + + + + +	3 + • • + +
S2	+ + 3 + + +	+ 3 + 3 3 3
S3	+ + + + + +	+ + + + + +
S4	+ + + + 3 +	+ + • • + +
S5	3 + + + + +	+ + + • + +
S6	+ + + + 3 +	+ • + • + +
S7	3 + + + + +	+ + + • + +
S8	+ + 3 + + +	+ + + + + +

[72] Eli Biham, Adi Shamir, "Differential Cryptanalysis of DES-like Cryptosystems", The Weizmann Institute of Science - Department of Applied Mathematics, Research Paper, p. 36, 1990.
[73] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, pp. 38-39.
[74] *ibid*, p. 39.
[75] *ibid*, p. 39.

For the other five S-boxes, it could be further verified that there are values of the missing key bits which are not used in the subkey *K6*. As a result, the verification of most of the 64 possibilities of the six missing bits will probably fail providing a unique single 6-bit value as the correct one with a high probability.[76]

It should be noted that similar tables can be constructed for different characteristics' rounds whenever necessary.

After the deduction of these six subkey bits, yet the cryptanalyst is to search the remaining unknown eight bits. These eight bits can be found by an exhaustive search of 256 combinations, or by applying a similar key analysis scheme to these eight bits that enter S-boxes in the fifth round.[77]

Calculating the signal to noise ratios, *Biham* and *Shamir* found out that a total of 240 pairs of ciphertexts, hence 240 pairs of chosen plaintexts, are necessary for the differential cryptanalysis of DES reduced to six rounds. Among these 240 pairs, 120 pairs come from the first characteristic Ω^1 and the other 120 pairs are derived from the second characteristic Ω^2. They also computed that an average of 2^{36} known plaintexts are required for this attack.[78]

Biham and *Shamir* also remark that in order to decrease the memory requirements in this attack and for the attacks to the ciphers with higher rounds, they devised a special algorithm named as *clique*. They also state that other efficient algorithms might be produced for the counting scheme that could get even better results with less memory consumption and faster look-ups.[79]

3.1.4.3 DES Reduced to Eight Rounds

The differential cryptanalysis of DES reduced to eight rounds is more complex than the former implementations, but the basic model is similar. Again, a suitable characteristic is chosen to extract some portion of the subkey in the eighth round, and the remaining key bits are found by construction of look-up tables and by the necessary manipulations on the key bits, similar to the one implemented in DES with six rounds. A five-round characteristic with a probability $\frac{1}{10485.76}$ was chosen for the cryptanalytic attack when *Biham* and *Shamir* implemented this model. This five-round characteristic can be extended to eight rounds by concatenating it with specifically chosen three additional rounds that are not covered by this characteristic. (The same mechanism was used in DES with six rounds.) The five-round characteristic is shown in Figure 3.12. In the same manner, for the succeeding rounds six, seven and eight; the input XORs are f', g', h', and the output XORs are F', G', H'.

The input XOR to the f function in the sixth round is the right XOR output from the fifth round of the characteristic, which is $(40\ 5C\ 00\ 00)_x$. Consequently, it' s also observed that for the five S-boxes (S2, S5, S6, S7, S8) in the sixth round, the

[76] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, p. 39.
[77] *ibid*, p. 39.
[78] *ibid*, pp. 40-41.
[79] *ibid*, p. 40.

output XOR from expansion permutation, $S'_{Ef} = 0$, the input XOR to the S-boxes, $S'_{If} = 0$, and the output XOR from these S-boxes, $S'_{Of} = 0$.[80]

By deriving the formula, $H' = T'_L \oplus g' = T'_L \oplus e' \oplus F'$ the output XORs of the corresponding S-boxes in the eighth round can be found. Since the input data of the eighth round can be granted from the ciphertexts, the 30 subkey bits entering the five S-boxes at the eighth round can be found by decrypting for each pair and counting the bits. This is the same mechanism used in the previous two versions of DES that extracted the 42 bits in both. The S/N is found to be 100 for this part of the analysis.[81]

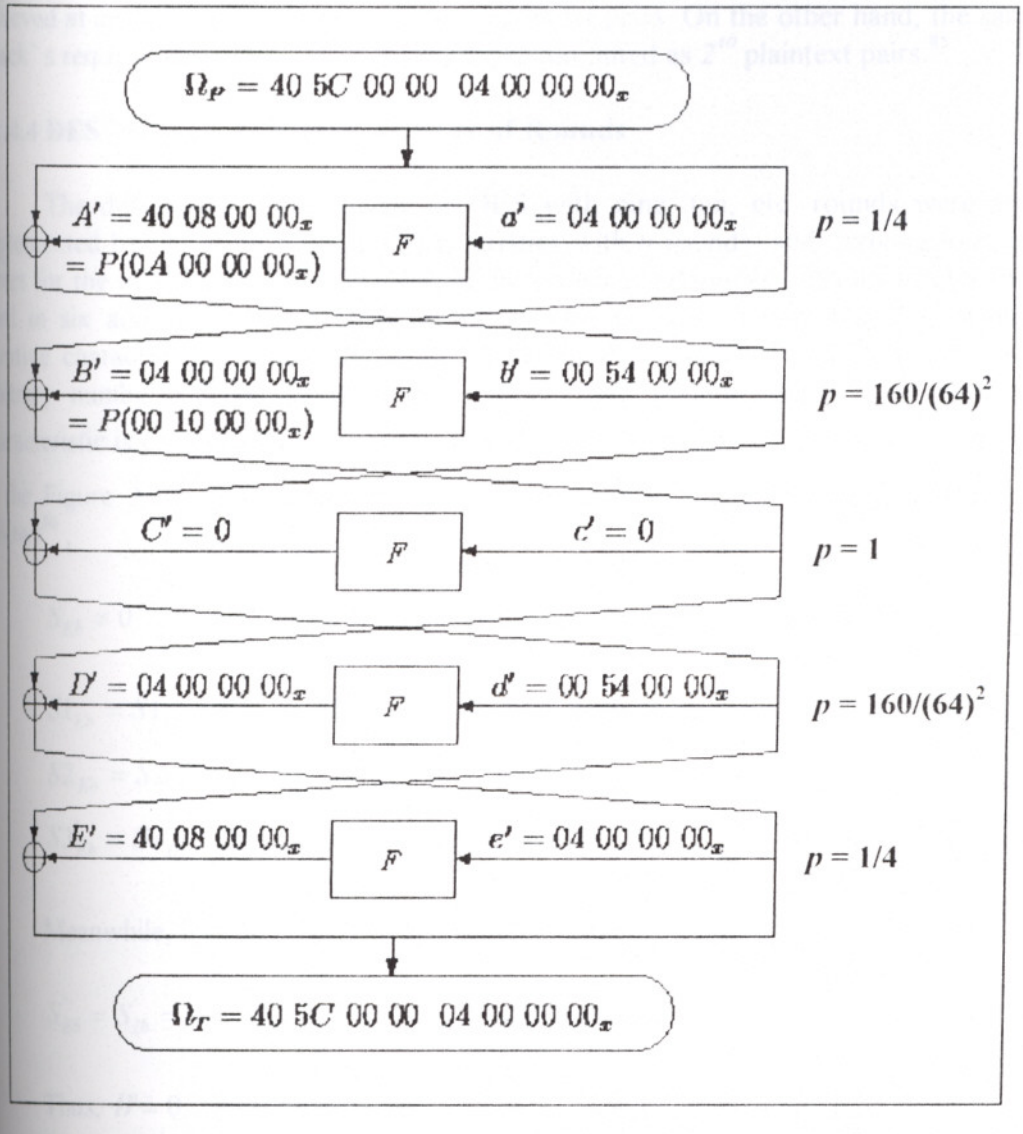


Figure 3.12 The five-round characteristic used for eight-round DES.[82]

Since exploring the 30 bits require a huge memory of 2^{30} counters, *Biham* and *Shamir* devised extra mechanisms and algorithms in order to reduce the memory consumption and speed up the counting process.[83]

[80] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, p. 42.
[81] *ibid*, p. 42.
[82] Eli Biham, Adi Shamir, "Differential Cryptanalysis of DES-like Cryptosystems", The Weizmann Institute of Science - Department of Applied Mathematics, Research Paper, p. 40, 1990.

In order to find out the remaining 26 bits, a filtering mechanism that extracts the wrong pairs, and a look-up table, that produces 18 bits out of 26, are used. The basic logic behind the scheme is similar to the one used in DES reduced to six rounds. This time, a table for the known bits at the seventh round is constructed, and the correct 18 bits of the subkey *K8* is extracted by the aid of known right pairs, bit dependencies, known bits, and so on. After the successful implementation of the scheme, 48 bits of the subkey *K8* is found and the cryptanalyst has only got to deduce the remaining eight bits, which can be found out by the ordinary exhaustive search.[84]

The differential cryptanalysis of DES reduced to eight rounds is successfully achieved at an average of *250,000* chosen plaintext pairs. On the other hand, the same attack' s requirement with known plaintexts is computed as 2^{40} plaintext pairs.[85]

3.1.4.4 DES With an Arbitrary Number of Rounds

The differential cryptanalysis of DES with nine, ten, etc. rounds were also implemented by choosing specific characteristics with n rounds, and deriving look-up tables for the missing bits, in other words, by establishing similar schemes to the ones used in six and eight-round DES. However, *Biham* and *Shamir* found a special iterative characteristic which enable the differential cryptanalysis of DES with any arbitrary number of rounds in a conventional and generic sense. This iterative characteristic is composed of two rounds and has a probability $\frac{1}{234}$ which is also given in the Figure 3.13. The probability of this iterative characteristic is calculated as follows[86];

$S'_{Eb} \neq 0$ only valid for three S-boxes: S1, S2 and S3 and hence,

$S1'_{Eb} = S1'_{Ib} = 03_x \rightarrow S1'_{Ob} = 0$ with probability $\frac{14}{64}$

$S2'_{Eb} = S2'_{Ib} = 32_x \rightarrow S2'_{Ob} = 0$ with probability $\frac{8}{64}$

$S3'_{Eb} = S3'_{Ib} = 2C_x \rightarrow S3'_{Ob} = 0$ with probability $\frac{10}{64}$

Meanwhile, for the other five S-boxes S4, ..., S8:

$S'_{Eb} = S'_{Ib} = 0 \rightarrow S'_{Ob} = 0$ with probability 1.

Thus, $B' = 0$ with probability $\frac{14}{64} * \frac{8}{64} * \frac{10}{64} \approx \frac{1}{234}$.

The iterative concatenation of this iterative characteristic with itself and with the one-round characteristic having $p = 1$ (given in Figure 3.3), enable the differential cryptanalysis of DES with any odd number of rounds. The attacks to these odd number

---

[84] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, pp. 42-44.
[85] *ibid*, pp. 44-45.
[86] *ibid*, p. 46.
[86] Eli Biham, Adi Shamir, "Differential Cryptanalysis of DES-like Cryptosystems", The Weizmann Institute of Science - Department of Applied Mathematics, Research Paper, pp. 47-48, 1990.

of rounds and their corresponding characteristics' probabilities are denoted in the Table 3.9. All these characteristics given in the table have $\Omega_P = \Omega_T = (19\ 60\ 00\ 00\ 00\ 00\ 00\ 00)_x = (\psi, 0)$. If any new round is added to any of these characteristics, the input XOR to the *f* function will be ψ in the next round and five of the S-boxes satisfy $S_E = 0$ in that next round.[87] This can be proven as follows[88] ;

The XOR data through the intermediate rounds for the plaintext XOR $\Omega_P = (\Psi, 0)$:

$a' = 0 \rightarrow A' = 0$ with probability 1

$b' = \Psi \rightarrow B' = 0$ with probability $\approx \frac{1}{234}$

$c' = a' \oplus B' = 0 \rightarrow C' = 0$ with probability 1

$d' = \Psi \rightarrow D' = 0$ with probability $\approx \frac{1}{234}$

$e' = c' \oplus D' = 0 \rightarrow E' = 0$ with probability 1

.

.

.

and so on, for any number of rounds.

Table 3.9 The probability of the iterative characteristic with respect to the number of rounds.[89]

Number of Rounds	Probability
3	$2^{-7.9} \approx 1/234$
5	$2^{-15.7} \approx 1/55000$
7	$2^{-23.6}$
9	$2^{-31.5}$
11	$2^{-39.4}$
13	$2^{-47.2}$
15	$2^{-55.1}$

There are also some other possible types of attacks carried out amongst arbitrary number of rounds in which some additional rounds are concatenated with the chosen characteristic whereas these rounds are independent of the characteristic, in other words, are not covered by the characteristic. For instance, as mentioned previously, in the differential cryptanalysis of DES reduced to six rounds, three additional rounds were concatenated with three-round characteristics and similarly, in the attack to DES with eight rounds, a five-round characteristic and additional three rounds were used. This kind of attack is named as *3R-attack*. There are other similar types of this attack; *2R-attack*, which adds two extra rounds and *1R-attack* with one additional round. A *0R-attack* is also possible but since it' s been proven that a *0R-attack* can be reduced to *1R-attack* with better probability (with better statistics and

[87] Eli Biham, Adi Shamir, "Differential Cryptanalysis of DES-like Cryptosystems", The Weizmann Institute of Science - Department of Applied Mathematics, Research Paper, p. 48, 1990.

[88] *ibid.* pp. 48-49.

[89] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, p. 48.

the same S/N), *0R-attack* is not used. It is also proven that *3R-attack* is advisable over *2R-attack*, and both are more preferable than *1R-attack* due to their relative probabilities and also due to the fact that for a fixed cryptosystem it' s advisable to use the shortest possible characteristic having better statistics.[90-91] In the following paragraphs, these attacks will be mentioned shortly.

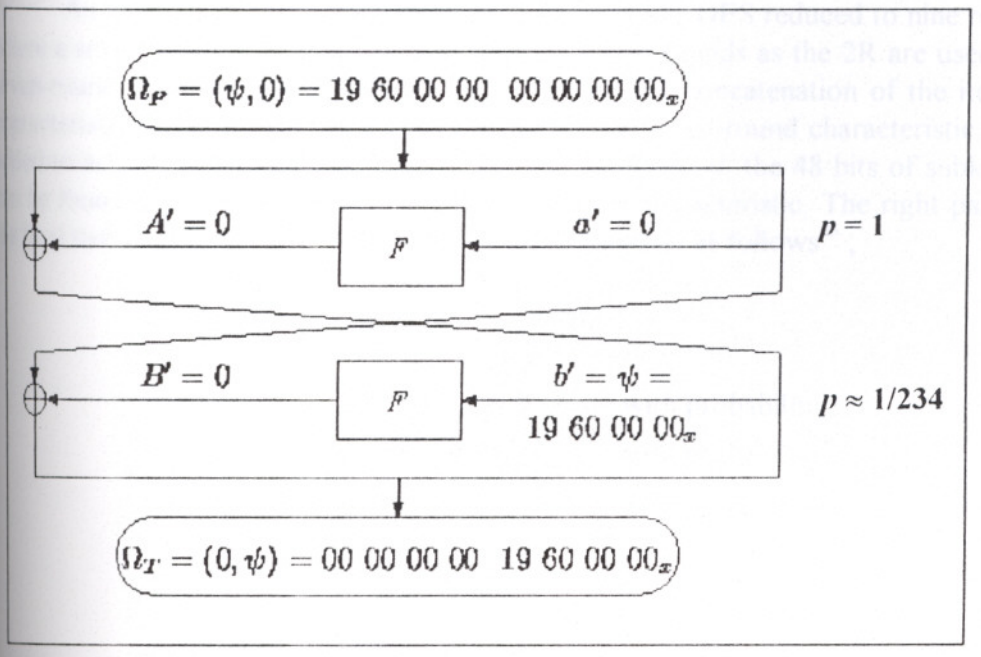


Figure 3.13 The iterative characteristic with probability about $\frac{1}{234}$.[92]

• 3R-Attacks

In 3R-attacks, three additional rounds are concatenated with the characteristic chosen for the attack. With the aid of these three rounds, counting can be applied to the bits of the subkey of the last round that enter S-boxes whose corresponding S-boxes in the first round of the additional three rounds have zero input XORs. The four, six and eight-round attacks described previously and also nine-round attack are all implemented in this manner.[93]

However, for DES reduced to 10 or more rounds, 3R-attacks are not recommended because it' s shown that the signal to noise ratio of the 3R-attack becomes too small.[94]

• 2R-Attacks

In 2R-attacks, counting is done on all the bits of the subkey of the last round. Whenever the input XORs of the S-boxes in the former round may not cause the

[90] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, p. 49.
[91] Eli Biham, Adi Shamir, "Differential Cryptanalysis of DES-like Cryptosystems", The Weizmann Institute of Science - Department of Applied Mathematics, Research Paper, p. 49, 1990.
[92] *Ibid.* p. 47.
[93] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, p. 49.
[94] *Ibid.* p. 50.

expected output XORs, wrong pairs can be discarded. For instance, the success rate of the check that output XOR is zero for the corresponding input XOR for an S-box is calculated as 1/16. For the other S-boxes, the success rate is proven to be approximately 0.8.[95]

An example of the implementation of 2R-attack is DES reduced to nine rounds, where a seven-round characteristic and additional two rounds as the 2R are used. The seven-round characteristic is derived from the iterative concatenation of the iterative characteristic given in Figure 3.13. At the end of the seven-round characteristic, 2R is added to achieve a total of nine rounds. For the ninth round, the 48 bits of subkey *K9* can be found using 2^{26} pairs within the seven-round characteristic. The right pairs for the final two rounds are also known which can be derived as follows[96] ;

for the plaintext XOR $\Omega_P = (\Psi,0)$:

round 1:	$a' = 0 \rightarrow A' = 0$	with probability 1
round 2:	$b' = \Psi \rightarrow B' = 0$	with probability $\approx \frac{1}{234}$
round 3:	$c' = 0 \rightarrow C'' = 0$	with probability 1
.	.	
.	.	
.	.	
round 7:	$g' = 0 \rightarrow G' = 0$	with probability 1

round 8: $h' = \Psi \rightarrow H' = i' \oplus g' = T_R'$

round 9: $i' = T_R' \rightarrow T_L' = h' \oplus T_L' = T_L' \oplus \Psi$

ciphertext XOR $\Omega_T = (T_L', T_R')$.

Thus, it could be checked that $h' \rightarrow H'$ and $i' \rightarrow I'$ for the two rounds of 2R (rounds eight and nine) and then, the possible occurrences of the key bits in the remaining pairs could be counted. It' s analyzed that for $h' \rightarrow H'$ five S-boxes satisfy $S_{Eh}' = S_{Ih}' = 0$ and hence S_{Oh}' will be 0 with a probability 1/16 among the wrong pairs, while the other three S-boxes satisfy $S_{Ih}' \rightarrow S_{Oh}'$ with a probability 0.8 among the wrong pairs. Therefore, counting on the key bits and deriving the associated S/N values, the subkey *K9* can be found within 2^{26} pairs.[97]

The 2R-attack can also be applied to other reduced round variants of DES in a similar way. DES reduced to eleven rounds can be broken using a nine-round characteristic and additional two rounds within 2^{35} pairs. The 13-round DES can be attacked with the eleven-round characteristic plus 2R using 2^{43} pairs and 15-round variant can be broken with a 13-round characteristic plus 2R using 2^{51} pairs.[98]

---

[95] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, p. 50.

[96] Eli Biham, Adi Shamir, "Differential Cryptanalysis of DES-like Cryptosystems", The Weizmann Institute of Science - Department of Applied Mathematics, Research Paper, p. 50, 1990.

[97] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, pp. 50-51.

[98] *ibid*, p. 51.

• **1R-Attacks**

In 1R-attacks, all the bits of the subkey, which enter S-boxes with non-zero input XORs can be counted for the last round. The values of T_R' can be verified and the possibility checks on all the other S-boxes in the last round can be achieved in order to find whether the input XOR may cause the output XOR or not. As analyzed in 2R-attacks, the check' s success rate is 1/16 for those S-boxes having zero output XOR within zero input XOR. But this time, it cannot be distinguished between several subkey values due to the fact that the input XOR of the last round is constant. However, this can be overcome since there aren' t many such subkey values and each of these can be checked later via exhaustive search or by a simple differential cryptanalytic method.[99]

An example for 1R-attack is the differential cryptanalysis of DES with ten rounds which can be achieved by a nine-round characteristic and additional round as 1R. It can be shown as below;

for the plaintext XOR $\Omega_P = (\Psi, 0)$:

round 1:	$a' = 0 \rightarrow A' = 0$	with probability 1
round 2:	$b' = \Psi \rightarrow B' = 0$	with probability $\approx \frac{1}{234}$
round 3:	$c' = 0 \rightarrow C' = 0$	with probability 1
.	.	
.	.	
.	.	
round 8:	$h' = \Psi \rightarrow H' = 0$	with probability $\approx \frac{1}{234}$
round 9:	$i' = 0 \rightarrow I' = 0$	with probability 1

round 10: $j' = \Psi = T_R' \rightarrow J' = i' \oplus T_L' = T_L'$

ciphertext XOR $\Omega_T = (T_L', T_R')$.

From this knowledge, the right pairs can be identified easily. It' s proven that those pairs that satisfy $T_R' = \psi$ and the 20 bits of T_L' coming from the S-boxes S4,..., S8 are zero. This assertion is also proven to be valid with a probability of 2^{-52} for the wrong pairs. Deriving S/N for the other S-boxes' bits and counting these bits, the attack can be successfully achieved using an average of 2^{34} pairs.[100]

In a similar manner, the twelve-round variant of DES can be broken using an 11-round characteristic and 1R within 2^{42} pairs. The 14-round variant can be broken using 13-round characteristic and 1R within 2^{50} pairs.[101]

---

[99] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, p. 51.
[100] *ibid*, p. 52.
[101] *ibid*, p. 52.

The 16-round DES, the complete standard DES in other words, is proven to be broken using a 15-round characteristic and 1R within 2^{57} pairs.[102] However, the 2^{57} complexity is even higher than a brute-force attack, which turns out the differential cryptanalytic attack useless. Thus, *Biham* and *Shamir* made some enhancements and derived a different differential cryptanalysis method for a successful attack to the full 16-round DES, which shall be explained in the following section.

3.1.5 Differential Cryptanalysis of the Full 16-Round DES

As mentioned in the previous section, the standard model for the differential cryptanalysis of DES reduced to several rounds was proven to be inefficient for the full 16-round DES, yet even having a worse performance than the brute-force attack. For this reason, a new method is established for the differential cryptanalysis of the standard 56-bit DES with 16 rounds, which will be described in this section. However, it should be stressed that this method is not applicable to independent-key variants of DES, which the former model was proven to be suitable among such variants.

In this new differential cryptanalysis model, a 13-round characteristic, an additional round and a 2R-attack with two rounds are combined so as to establish the total 16 rounds. The 13-round characteristic is derived from the iteration of a two-round iterative characteristic six and a half times, which this iterative characteristic is the one mentioned in the former sections and given in Figure 3.13. Since, this iterative characteristic' s probability is about 1/234, iterating it six and a half times result with a 13-round characteristic having probability $p \approx 2^{-47.2}$.[103] This 13-round characteristic is used after combining it with an initial one-round, thus the 13-round characteristics form the 2nd to 14th rounds of the 16-round DES. This additional one-round which is embedded as the first round of DES is stated as the crucial part of the attack. This one-round is added in order to eliminate the possible mixture of wrong and right pairs, to decrease the huge memory requirement, and most importantly, to establish an alternative method which can entirely eliminate the use of counters.[104] After the implementations, it' s been proven that this one-round satisfies all these requirements. With the usage of this additional round as the first round of DES, rather than extracting possible values for a subset of key bits, a list of complete 56-bit candidate keys can be suggested and examined independently within each pair. In fact, this aspect is stated to be one of the core differences of this method when compared to former basic differential cryptanalysis model. By this way, each suggested 56-bit key value can be immediately tested via trial encryption, without any counting scheme or any counters.[105] The 2R-attack is added finally to the 14 rounds which makes up the 16-rounds in total. This full 16-round DES implementation of the attack is also denoted in the Figure 3.14.

The attack can be divided into two parts, where the first part is the first 14 rounds, the combination of the first round and 13-round characteristic, and the second part is the usage of 2R-attack, which is also defined as the *data analysis* phase.

---

[102] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, p. 52.
[103] *ibid*, p. 79.
[104] *ibid*, p. 80.
[105] *ibid*, p. 80.

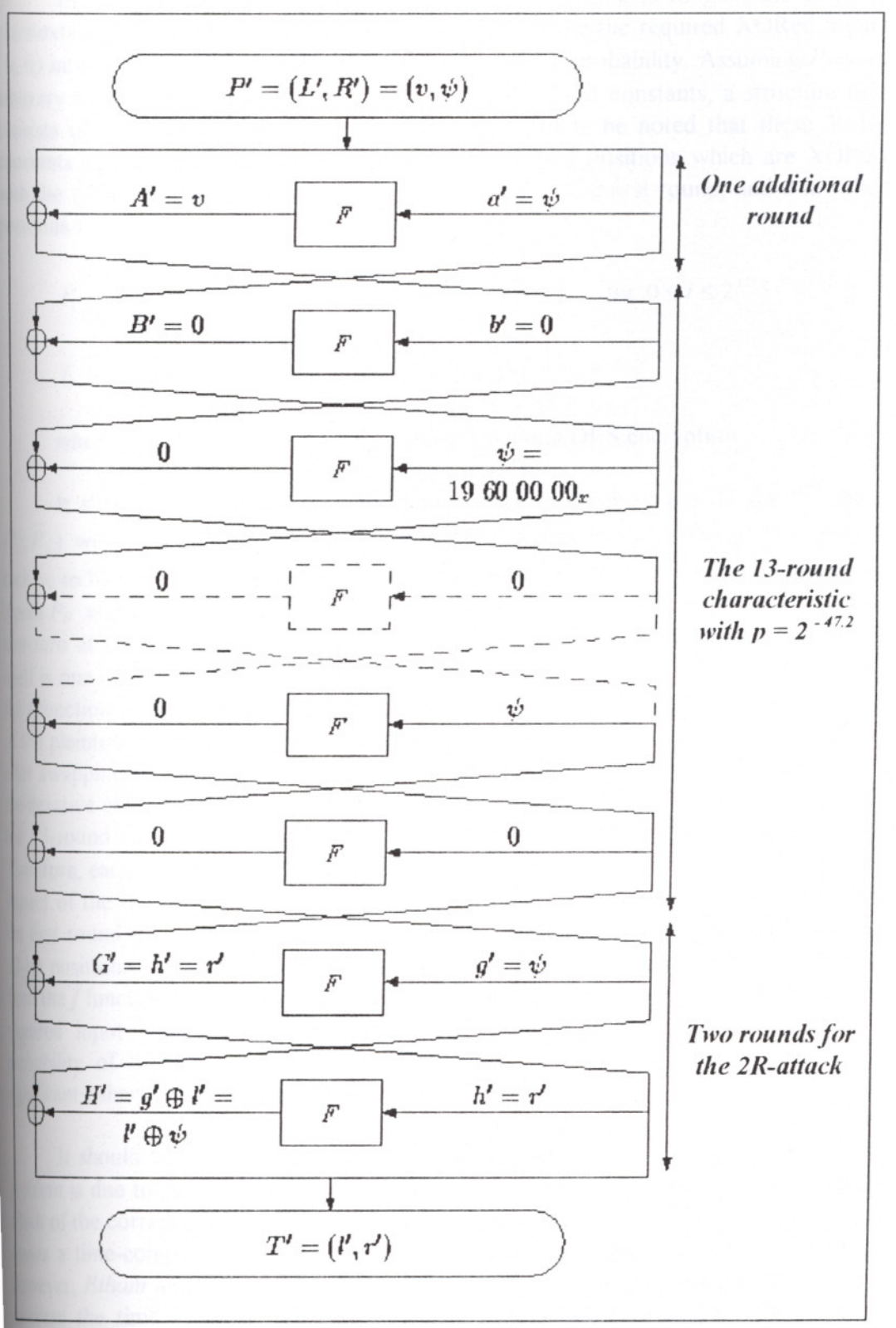


Figure 3.14 The differential cryptanalytic attack to the full 16-round DES.[106]

[106] Eli Biham, Adi Shamir, "Differential Cryptanalysis of the full 16-round DES", Technion- Israel Institute of Technology, Technical Report, p. 5, 1991.

In the first part, or phase namely, the essential goal is to generate pairs of plaintexts whose XORed outputs after the first round are the required XORed inputs (ψ, 0) into the 13-round characteristic without a loss of probability. Assuming P as an arbitrary 64-bit plaintext and v_0, ..., v_{4095} as the 2^{12} 32-bit constants, a structure that consists of 2^{13} plaintexts is defined as follows (it should be noted that these 32-bit constants consist of all the possible values at the 12-bit positions which are XORed with the 12 output bits of the S-boxes S1, S2, S3 after the first round, and 0 in other positions.)[107] ;

$$P_i = P \oplus (v_i, 0) \qquad \overline{P}_i = \left(P \oplus (v_i, 0)\right) \oplus (0, \psi) \qquad \text{for } 0 \le i \le 2^{12}$$

$$T_i = DES(P_i, K) \qquad \overline{T}_i = DES(\overline{P}_i, K)$$

where K is the 56-bit initial key used in 16-round DES encryption.

It should be remarked that the plaintext pairs involved are all the 2^{24} pairs $(P_i, \overline{P}_j)$ with $0 \le i, j \le 2^{12}$. Their plaintext XOR is proven to be always (v_k, ψ) and each v_k to be occurring exactly in 2^{12} pairs. Since for the first round of the attack the inputs P_R and $P_R \oplus \psi$ entering the f function produces an output XOR which is only non-zero at the output bits coming from S-boxes S1, S2, and S3, this output XOR itself is one of the correct v_k value. Thus, among all the 2^{12} pairs, the output XOR of the f function in the first round is completely cancelled by XORing it with the left half of the plaintext XOR. As a result, the output XOR of the first round becomes (ψ, 0) after swapping the right and left halves, and this is the desired input XOR value into the iterative characteristic. This result enables the concatenation of the first round with the 13-round iterative characteristic which can also be analyzed from Figure 3.14. Therefore, each structure has a probability $p \approx 2^{12} * 2^{-47.2} \approx 2^{-35.2}$. In other words, the impact of the first additional round on the attack can be summarized as follows: For the first round of the 16-round attack, 2^{12} 32-bit samples for all the combinations of 12-bit positions are used in order to derive the required v_k that cancels output XOR v from the f function so that the output XOR of the first round can be forwarded as the required input XOR for the second round which is (ψ, 0). This increases the probability of the 13-round characteristic from $2^{-47.2}$ to $2^{-35.2}$ which provides a significant enhancement to the overall 16-round attack.[108]

It should be noted that a potential problem can be faced in this method. This problem is due to the fact that the actual value of v_k is not known initially and hence, which of the correct pair among the 2^{12} plaintext pairs to be chosen is not known. This causes a time-complexity problem because all the 2^{24} possible pairs should be tried. However, *Biham* and *Shamir* also derived a scheme to overcome this problem and to decrease the time required for computations. This scheme uses the cross-product structure of the pairs so that the right pairs could be extracted among them in 2^{12} time. And using some sorting and selection mechanisms as well as exploiting the specific properties of the S-boxes, most of the wrong pairs can be eliminated in a few computations resulting with an average of 1.19 pairs per structure as the expected

[107] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, pp. 80-81.
[108] *ibid*, p. 82.

output of the data collection phase. All these necessary computations and mechanisms are shown to be implemented by a few table look-up operations as well; decreasing the time complexity further and hence, eliminating the necessary problem.[109]

The first phase of the attack within the first 14 rounds extracts most of the required right pairs but all the wrong pairs are not discarded; thus the input of the second phase, data analysis phase namely, is still a mixture of right and wrong pairs. In the data analysis phase, the goal is to discard the wrong pairs completely and to derive the correct 56-bit key finally with the aid of 2R-attack. As explained in the former sections, the data analysis phase use huge arrays up to 2^{42} counters imposing extremely high memory consumption, usually whenever the rounds increase. Hence, in order to provide an efficient attack to the full 16-round DES, a new method is developed and used in this phase as well. Rather than using counters and counting on the key bits, each suggested 56-bit key value is tried immediately. This assertion is proven to be valid, because an entire key value can be suggested whenever it can create the output XOR values of the last round as well as the expected output XOR of the first round and the 15^{th} round for the particular plaintexts and ciphertexts. Checking and comparing some of the bits of the key that enter and exit some specific S-boxes, most of the wrong pairs can be discarded as well as reducing the possible candidate key values. At the end of this analysis each analyzed pair suggests around 0.84 values for the 52 bits of the entire key, where each value corresponds to 16 possible values of the 56-bit key. Therefore, each structure suggests approximately 1.19*0.84*16 = 16 choices for the key value. With the combination of all the structures and some further filtering mechanisms, only 2^{12} possible key values can be left in the hand. These values can be tested via trial encryption among any single plaintext / ciphertext couple, and with a very high probability, the extracted key will be the correct 56-bit key value. It should also be noted that S/N is proven to be $2^{16.8}$ for the data analysis phase, much better than the 2^9 value calculated for the 16-round DES using the basic differential cryptanalytic attack model described previously.[110]

In addition, it' s also proven that this data analysis can be carried out more efficiently by carefully choosing the order of the various key bits being tested. With the help of some specific S-boxes' input bits for the first and 16^{th} rounds, and analyzing these in some order, and by making some necessary calculations, the deduction of some of the key bits could be achieved in less computations.[111]

Now that all the 16 rounds of the attack has been covered, the performance of the overall attack can be calculated as follows: Each structure contains a right pair with a probability $2^{-35.2}$ coming from the first phase of the attack. Also, since a total of 2^{35} structures are encrypted, and since each structure consists of 2^{13} plaintexts, there are $2^{35} * 2^{13} = 2^{48}$ chosen plaintexts in hand. Moreover, since it' s been calculated that about 1.19 pairs exist per each structure, then about $2^{35} * 1.19 = 2^{35.25}$ pairs and hence, $2^{36.25}$ ciphertexts remain as candidate inputs to the second (data analysis) phase. The probability that at least one of these pairs is a right pair is found to be 0.58, and the analysis of any right pair assures the deduction of the correct 56-bit key. Thus, the

---

[109] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, pp. 82-83.
[110] *ibid*, p. 83.
[111] *ibid*, pp. 83-85.

overall time complexity of the attack becomes $2^{35} * 4 = 2^{37}$ equivalent DES operations.[112] Finally, all these values show that this attack can be regarded as feasible and successful when compared to brute-force attack with 2^{55} complexity.

In order to reduce the number of chosen plaintexts involved in this attack, *Biham* and *Shamir* extended the method by using an additional structure established within a two-round iterative characteristic. Since, the basic collection of plaintexts in this attack is proven to be a structure rather than a pair, some metastructures that contain 2^{14} chosen plaintexts could be established. They developed a metastructure that' s composed of four structures where two of them come from the iterative characteristic used in the 13-round of the attack, and the other two is developed from an iterative characteristic shown in the Figure 3.15. With the use of metastructures, four times as many pairs from twice as many plaintexts can be obtained. Thus, the number of chosen plaintexts required in this attack is reduced from 2^{48} to 2^{47}.[113]

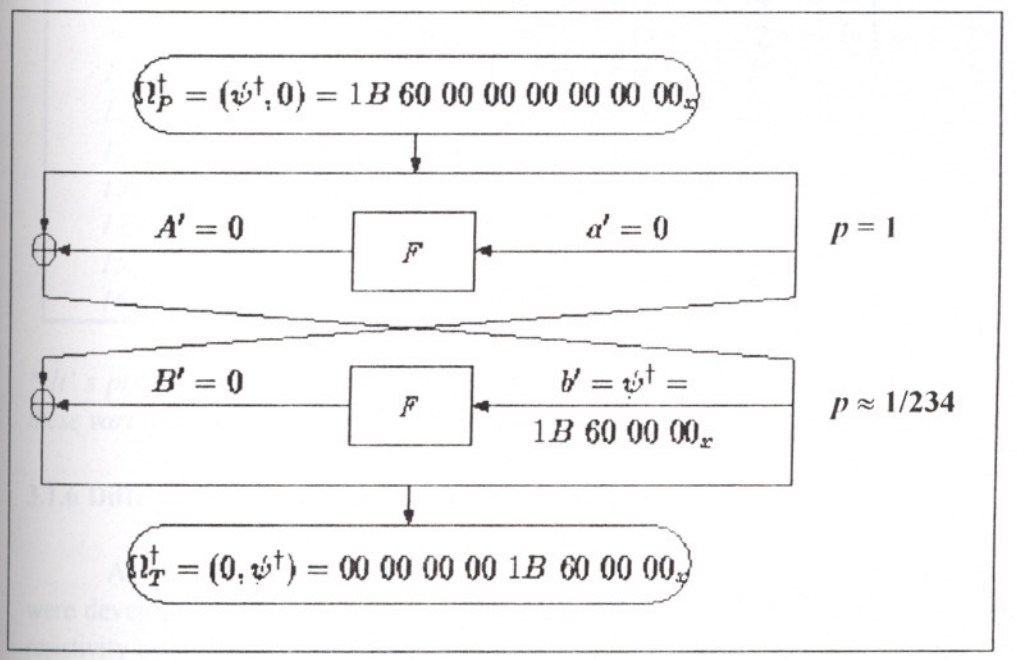


Figure 3.15 The iterative characteristic used for improving the attack to the full 16-round DES.[114]

The differential cryptanalysis of the full 16-round DES might even be improved with further achievements. For instance, *Biham* and *Shamir* noted that since the mechanisms in this attack use different structures and instances independent and unrelated of each other, the model could be successfully implemented in parallel with up to 2^{33} nodes. However, the differential cryptanalysis of the full 16-round DES has also some shortcomings. Besides its inapplicability to DES with independent keys as stated previously, this model is not directly applicable to plaintexts having ASCII characters only. This is due to the fact that such plaintexts cannot supply the desired XOR differences. Only, by making some changes in the model and by using different

[112] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, p. 85.
[113] *ibid*, p. 86.
[114] Eli Biham, Adi Shamir, "Differential Cryptanalysis of the full 16-round DES", Technion- Israel Institute of Technology, Technical Report, p. 8, 1991.

iterative characteristics, the attack can be successfully implemented within 2^{49} chosen ASCII plaintexts.[115]

There are other variants of this attack developed for some enhanced versions of 16-round DES and also for the reduced rounds of DES using the similar schemes and the model explained in this section. Thus, the best results could be deducted for DES with any rounds via all the different differential cryptanalysis models explained so far. These results are given in the Table 3.10.

Table 3.10 Best results achieved for the differential cryptanalysis of DES with different rounds.[116]

No. of Rounds	Chosen Plaintexts	Known Plaintexts	Analyzed Pairs	Complexity of Analysis
8	2^{14}	2^{38}	4	2^{9}
9	2^{24}	2^{44}	2	2^{32} *
10	2^{24}	2^{43}	2^{14}	2^{15}
11	2^{31}	2^{47}	2	2^{32} *
12	2^{31}	2^{47}	2^{21}	2^{21}
13	2^{39}	2^{52}	2	2^{32} *
14	2^{39}	2^{51}	2^{29}	2^{29}
15	2^{47}	2^{56}	2^{7}	2^{37}
16	2^{47}	2^{55}	2^{36}	2^{37}

* *It' s proven that the complexity of the analysis could be reduced significantly for these variants by using the clique method.*[117]

3.1.6 Differential Cryptanalysis of DES Variants

As mentioned in Chapter 2, section 2.4.5.2, some different variants of DES were developed so as to improve the security and the efficiency of the algorithm. The resistivity performances of these variants among several cryptanalytic attacks including the differential cryptanalysis are also given in the section 2.4.5.2. The differential cryptanalytic attack is proven to be successful among most of these implementations and some of these will be mentioned shortly throughout this section. By analyzing all these attacks and making comparisons, an important and final statement for the differential cryptanalysis of DES might be as follows: The amount of information the cryptanalyst obtains is heavily dependent on how many rounds are used for DES. For a DES version with many rounds, less information can be obtained directly, making analysis rather difficult. For reduced round variations of DES, *Biham* and *Shamir* proved that their method can break DES even with independent subkeys in less than two minutes on an average personal computer platform for eight-round variations, and far faster for fewer rounds.[118] From these results, it becomes worthwhile to study their method, and search for new schemes and improvements in this method.

[115] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, p. 86.
[116] *ibid*, p. 87.
[117] *ibid*, p. 87.
[118] *ibid*, p. 68.

• **DES with Modified *P* Permutation**

Since it was proven that the choice of the *P* permutation had a major influence on the existence of high probability characteristics, some changes were applied to the design of the *P* permutation so as to improve the security of DES. However, these implementations were proven to be weakening the security of DES, all but one. The only improvement was shown to be achieved via replacement of the *P* permutation by the identity permutation or entirely elimination of the *P* permutation.[119] The differential cryptanalysis against such a DES variant is established successfully with the usage of a special three-round iterative characteristic. When this characteristic is iterated to a ten-round characteristic, the probability is shown to be around $2^{-14.5}$ where this characteristic can be used for the entire 16 rounds of DES by making some extensions with the aid of some of the S-boxes' property in the 14th round. This extended characteristic is proven to have a probability $2^{-16.5}$ and within 2^{20} pairs and the attack is shown to be successful.[120]

This analysis is extended further to attack to DES with all sort of random permutations, and is proven to be successful among all such variants within the complexity between 2^{20} - 2^{42}. All these successful differential cryptanalytic attacks have also proven that the replacement of the *P* permutation by any permutation cannot make DES stronger.[121]

• **DES with Modified and Random S-Boxes**

Some of the DES variants were changed in the design of S-boxes in order to improve the security of the cipher. One of these variants is DES with S-boxes modified in the order. Since, in the standard DES, the eight boxes are arranged in a certain order, altering the order of these S-boxes and rearranging them probably changes the structure of the encryption system. However, with the help of differential cryptanalysis all such alternative DES models are proven to be weaker than the original DES. For any different order arrangement, a suitable iterative characteristic can be generated and used for the differential cryptanalytic attack. *Biham* and *Shamir* proved this in several successful experiments. For one of the modified order model, they achieved the differential cryptanalysis of the 16-round system successfully using 2^{46} chosen plaintexts with S/N = 2^{21} and using 2^{55} known plaintexts.[122]

Another approach was making some modifications in the inner structure of the S-boxes themselves, ie, changing bit entries, etc. so as to improve the security of DES. Sooner or later, such these variants are proven to be weakening the security of DES with the aid of differential cryptanalytic attacks. For instance, a 16-round DES variant with a modified S1 was broken using a specific two-round iterative characteristic 7.5 times plus 1R-attack with 2^{37} pairs and S/N = 2^{29}, an attack much better than made to the original 16-round DES.[123]

---

[119] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, p. 56.
[120] *ibid*, p. 56.
[121] *ibid*, p. 57.
[122] *ibid*, pp. 57-58.
[123] *ibid*, pp. 61-62.

Yet another idea aiming to improve the security of DES was using random S-boxes in the algorithm. Since, all the eight S-boxes used in the standard DES algorithm are known with all their entries where these entries are constant; using randomly selected different S-boxes from a wide range set of S-boxes for each encryption was thought to shield the cryptanalyst from knowing the complete structure of the algorithm, presumably. Alas, this is also proven to be an attempt in vain because of the successful differential cryptanalytic attacks. The essential aspect lying behind this success is due to the discovery of a specific two-round iterative characteristic. It' s proven that no matter what random S-boxes are used, 97% of such sets have this iterative characteristic with a probability of 1/8 or higher. Producing a 13-round characteristic with $p = 2^{-18}$ by the concatenation of this iterative characteristic with itself six and a half times, and adding a 3R-attack, 42 subkey bits are proven to be found within the requirement of 2^{20} pairs. Moreover, the entire 56-bit key is proven to be broken for nearly any random S-box set within 2^{43} - 2^{47} pairs.[124]

- **DES with Independent Subkeys**

In the standard DES model, each subkey for each round is derived from a simple shifting mechanism of the initial 56-bit key which makes all of the subkeys dependent on the initial key. Some security experts developed a new variant of 16-round DES, which each 48-bit subkey generated for each of the 16 round is independent of each other as well as the initial key. This new variant seems much more secure than the standard 56-bit DES at first sight, especially considering the brute-force attack with a gigantic computational complexity of 2^{768}. But the differential cryptanalytic attacks implemented for this DES variant afterwards proved that this wasn' t true. However, these attacks also proved that this variant is more secure than the standard 56-bit DES besides proving the differential cryptanalysis' applicability and efficiency when compared to exhaustive search or other cryptanalytic attacks.

Biham and *Shamir* first developed a model for the differential cryptanalysis of DES with independent subkeys reduced to eight rounds. In fact, this model is basically similar to the one explained in this Chapter, section 3.1.4.3; but this time some necessary changes and additional mechanisms are established due to the independent subkey phenomena. These mechanisms and the algorithm of the attack is not given here for the sake of simplicity, but in short, the basic idea is to adapt the scheme used for finding the subkey *K8* in round eight to the previous rounds and for other keys by analyzing each independently and by counting the subkey bits and analyzing the S-boxes' entries and output differences independently through each round. This attack is successfully carried out within 250,000 pairs for the chosen plaintext version. Also, it is proven that within 2^{40} known plaintexts, the attack can be successfully implemented with all the subkeys broken.[125]

The attack for the eight rounds can be extended to 16 rounds with independent keys in a similar manner. Using three different characteristics and an additional iterative characteristic, and analyzing some of the rounds independently, all the independent keys through the 16 rounds can be found out. The entire attack requires 2^{59} pairs

[124] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, pp. 60-61.
[125] *ibid.* pp. 65-68.

derived from 2^{60} chosen plaintexts within 2^{60} computational complexity where the known plaintext version requires $2^{61.5}$ known plaintexts. These are the best results achieved ever since, far from being practical and effective but even much better than the brute-force attack.[126]

• **GDES**

GDES is another variant of DES, which was initially designed aiming both to improve the security of DES and to speed up the encryption / decryption processes. However, GDES was proven to be much less secure than the original DES by some different cryptanalytic attacks. *Biham* and *Shamir* also managed to prove this fact via differential cryptanalytic attack.

GDES is a structurally modified variant of DES, where some changes are made in the design of the algorithm, especially the structure and processing of the *f* function is modified so as to increase the ratio between the block size and the number of evaluations in the *f* function. By this way, the processing speed of the cryptosystem can be increased significantly.[127]

Since, the some of the basic operations and the *f* function itself is modified in GDES, the differential cryptanalytic attack to GDES had to be modified considerably as well. This ended up with a new differential cryptanalysis method in which characteristics are not used, but the attack is mostly focused on the *f* function. Again, exploiting the properties of XOR, some special difference operations and equations are derived and used. Also, sum of the key bits for specific rounds and for specific S-boxes are analyzed within the right pairs and some counting schemes are used in the attack. This attack is proven to be applicable both with chosen plaintext pairs and known plaintext pairs. The attack is also extended to different various models of GDES with different rounds and successfully implemented amongst all of them.[128]

Some of the results achieved for differential cryptanalysis of GDES are given here whereas the details of the attack and the mechanisms in GDES are not explained for the sake of simplicity. GDES with 16 rounds is proven to be broken easily with *6* ciphertexts only, and with *16* ciphertexts if independent keys are used. GDES with 22 rounds can be broken with 24 pairs, hence 48 ciphertexts. Even more complicated GDES models can be broken; for instance, GDES of 31 rounds is proven to be successfully attacked with 250,000 pairs and S/N $\approx 2^7$ and the most secure GDES model, 64 rounds with independent keys can even be broken with 2^{49} chosen plaintexts, proving that any feasible GDES model is less secure than a standard 56-bit DES. However, the differential cryptanalytic attacks with known plaintexts to GDES variants of high rounds are not recommended, since this is shown to be requiring extremely huge number of plaintexts and making the attack inefficient.[129]

---

[126] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, p. 68.
[127] *ibid*, p. 69.
[128] *ibid*, pp. 71-76.
[129] *ibid*, pp. 76-77, 88.

3.1.7 Differential Cryptanalysis of Some Other Cryptosystems

The differential cryptanalysis of some iterated symmetric block cryptosystems are proven to be successfully achieved as well as DES. Since in this study DES, as being the core and basic model of all symmetric block ciphers, is mostly focused and analyzed, its differential cryptanalysis is also explained in detail and mainly concerned. The differential cryptanalytic attacks to some other well-known ciphers are summarized in this section to give an idea and any further detailed information is left to the reader. It should be stressed that in some of these attacks such as FEAL, REDOC-II, etc, XOR is used again for the difference operation and some special characteristics are derived as similar to DES; but in some others, entirely different methodologies and some difference operations rather than XOR are used for the differential attacks.

- **Differential Cryptanalysis of FEAL**

The differential cryptanalysis of FEAL is first achieved successfully by *Biham* and *Shamir* where they managed to attack several variants of FEAL such as FEAL-8, FEAL-N and FEAL-NX as well. For instance, they have proven that four-round standard FEAL cryptosystem is breakable with eight pairs; an eight-round FEAL (FEAL-8) can be broken with *128* pairs of chosen plaintexts and 2^{36} known plaintexts. Both FEAL-N and FEAL-NX is proven to be broken faster than brute-force attack for rounds $N \leq 31$, hence being effective among those rounds, where for a 31-round attack 2^{63} chosen plaintexts or 2^{63} known plaintexts are needed.[130]

- **Differential Cryptanalysis of Khafre**

As similar to DES and FEAL, Khafre uses some kind of S-boxes and *f*-functions which makes the differential cryptanalysis possible and suitable by using XORs and deriving some special characteristics. *Biham* and *Shamir* proved that the differential cryptanalysis of Khafre can be even more efficient than exhaustive key search for the known highest rounds. The 16-round Khafre can be broken with *1536* chosen plaintexts or $2^{37.5}$ known plaintexts. Moreover, 24-round Khafre can be broken with 2^{53} chosen plaintexts or $2^{58.5}$ known plaintexts.[131]

- **Differential Cryptanalysis of RC6**

The differential cryptanalysis of RC6 and also some of its variants such as RC6-I, RC6-NFR and RC6-I-NFR have been successfully implemented by some cryptanalysts. The difference operation used in the attack is not only XOR, but also an additional operation named as *integer-subtraction mod* 2^{32} is used; some unique methods and algorithms such as *Hamming weight* and *rotation amounts*; and special characteristics entirely different from the ones in DES are derived and adapted for the attack. Some of the results achieved within the attacks to RC6 variants are considered to be successful. For instance, a 20-round RC6-I variant is proven to be broken with 2^{59} plaintext pairs and a 24-round RC6-I is broken with the usage of 2^{80} plaintext pairs. Consequently, a 20-round RC6-NFR variant is broken with 2^{84} plaintext pairs and its

[130] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, pp. 89-104.
[131] *ibid*, pp. 109-114.

24-round version could be attacked successfully with 2^{103} plaintext pairs. For the RC6-I-NFR variant, the 20-round is broken with 2^{66} plaintext pairs and the 24-round can be broken with 2^{76} plaintext pairs.[132]

- **Differential Cryptanalysis of REDOC-II**

REDOC-II is a ten-round cryptosystem with 70-bit blocks and for each round, some special substitution and permutation operations are processed on data bits with a round key. However, instead of S-boxes, some specific *enclave*, permutation and key tables are used. For this reason, different techniques and special mechanisms are established and used for the analysis of differences in the differential cryptanalysis of REDOC-II. The difference operation is XOR again, and the attack is carried out again with the aid of some specific characteristics derived for REDOC-II within the plaintext pairs. The attacks known so far are achieved by *Biham* and *Shamir*, which is proven to be successful and efficient up to four rounds. The differential cryptanalysis of REDOC-II for further rounds might be achieved successfully if new and better implementations can be made. It should be noted that among the attacks for the rounds 1 up to 4, the complexities within chosen plaintexts and known plaintexts are proven to be very close to each other and this was not observed in any of the previous cryptosystems. For instance, the differential cryptanalytic attack to the three-round REDOC-II is achieved within 2^{46} chosen plaintexts and $2^{56.5}$ known plaintexts; for four-round REDOC-II the attack can be carried out with equivalent number of pairs, 2^{67} chosen plaintexts and known plaintexts.[133]

Ronald L. Rivest, Scott Contini, M. J. B. Robshaw, Yiqun Lisa Yin, "The Security of the RC6 ock Cipher", RSA Laboratories, Technical Report, pp. 15-40, 1998.

Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, pp. 115-121.

3.2 Linear Cryptanalysis

3.2.1 Introduction

Linear cryptanalysis is another type of cryptanalytic attack generated for symmetric block ciphers. This attack is invented by *Mitsuru Matsui*, several years after the invention and implementation of differential cryptanalysis. Linear cryptanalysis can be simply defined as a cryptanalytic attack that uses linear approximations to analyze and discover the action of a block cipher, where these approximations are derived from the statistical linear relations between the bits of the plaintexts, ciphertexts, and the relevant keys. With the aid of these linear relations, good predictions can be made for some portion of the key bits, or the entire key can be broken. It' s shown that linear cryptanalysis is quite different from the differential cryptanalysis in implementation and detail; however, considering the basic structure and viewing from the point of the design' s core, both attacks are similar to each other.[134,135]

The basic notions and the logic lying behind the linear cryptanalysis is explained in this section, and in the following sections, the details of the algorithm, structures, methodologies, the implementations among DES and several other ciphers, and any further information will be given. However, it should be stressed that since linear cryptanalysis is much newer than differential cryptanalysis, less knowledge and results have been granted so far where new improvements and extensions are being derived every day.

Linear cryptanalysis is generally carried out with known plaintexts since it' s proven that with chosen plaintexts, the performance is significantly lowered. This is just the opposite of the differential case. However, any linear cryptanalytic known plaintext attack can be converted to its chosen plaintext version, if desired.

The essential and basic idea behind the linear cryptanalysis is to exploit the linear flaws in the design or the processing of the cryptosystem. Most of the cryptosystems such as DES, are designed to have non-linear features and accepted to be non-linear. However, some linear properties were uncovered later on which led to the linear cryptanalysis of these systems. In DES, these linear flaws or some linear properties are proven to exist within the S-boxes where all the other operations and the functions of the algorithm are non-linear. Thus, the linear cryptanalysis of DES is established by attacking to the S-boxes.[136]

In other words, it' s proven that the linear cryptanalytic attack is heavily dependent on the structure of the S-boxes among DES and DES-like cryptosystems, and the S-boxes in the standard DES are not optimized against such an attack. Yet, one of the designers of DES, *Don Coppersmith* once stated that;

---

[134] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 290.
[135] Eli Biham, "On Matsui' s Linear Cryptanalysis", Technion- Israel Institute of Technology, Technical Report, p. 1, 1994.
[136] *ibid*, p. 2.

Resistance to linear cryptanalysis was not part of the design criteria of DES.[137]

This might yield the cryptanalysts to look more hopefully over linear cryptanalysis, recalling the differential issue. Even that the strong resistivity to differential cryptanalysis was a part of the design criteria for DES, the differential cryptanalytic attacks generated so far have been successful to some extent. Therefore, with the new improvements, linear cryptanalysis might be much more efficient and impressive than differential cryptanalysis in the future.[138] A more important fact is that, due to the results achieved so far, linear cryptanalysis is proven to be more efficient and successful than the differential cryptanalysis; furthermore, it is accepted as the most powerful attack known on DES and DES-like block cipher algorithms at the time of this thesis' writing.[139]

The first step in any linear cryptanalysis is deriving the linear relations between some of the specific bits of the plaintext, corresponding ciphertext and the key. For DES and most of the other ciphers, this relation can be simply defined by the equation;

$$\left(P_i \oplus P_j \oplus ... \oplus P_t\right) \oplus \left(C_i \oplus C_j \oplus ... \oplus C_t\right) = \left(K_i \oplus K_j \oplus ... \oplus K_t\right)$$

where, P_i, ..., P_t can be any subset of bits of the plaintext P, C_i, ..., C_t can be any subset of bits of the corresponding ciphertext C and similarly K_i, ..., K_t can be any subset of the key or subkey bits for any round. It should be noted that in both sides of the equation, a single bit value, 0 or 1 is achieved. This equation denotes a linear relation which produces a linear approximation. In other words, if the system is completely unbiased or non-linear, this equation must always hold with a probability of ½ for any combination. But, whenever this probability is lower or higher than 0.5, than there' s a bias in the system posing a linear relation in some or all of these bits in the equation. Exploiting this bias, some of the key bits or any single bit of the key can be guessed with some probability. The more data used and the more relations derived, the higher the probability of a correct guess. Moreover, the greater the bias, the higher the probability of a correct guess with the same amount of data.[140]-[141] In other words, the success of any linear cryptanalytic attack is strictly dependent on the size of the set of plaintext / ciphertext blocks so as to provide good guesses for the key bits and this can be established by using these linear approximations derived from such sets.[142]

The linear approximations are basically achieved for each S-box independently. Also, these linear approximations are converted into linear approximation tables for each and every S-box. In contrast, these tables are somewhat like the difference

---

[137] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 293.
[138] *ibid*, p. 293.
[139] S. Bakhtiari, R. Safavi-Naini, "Application of PVM to Linear Cryptanalysis", University of Wollongong, Technical Report, p. 2, July 25, 1994.
[140] Bruce Schneier, *Applied Cryptography - Second Edition*, pp. 290-291.
[141] Ronald L. Rivest, Scott Contini, M. J. B. Robshaw, Yiqun Lisa Yin, "The Security of the RC6 Block Cipher", RSA Laboratories, Technical Report, p. 41, 1998.
[142] S. Bakhtiari, R. Safavi-Naini, "Application of PVM to Linear Cryptanalysis", University of Wollongong, Technical Report, p. 2, July 25, 1994.

distribution tables of S-boxes used in differential cryptanalysis. The detailed study of these tables will be given in the subsequent section.

A simple example of a one-round linear approximation for DES is denoted in the Figure 3.16. In this approximation 17th bit of the plaintext X, X_{17} namely, 26th bit of the key for the round i, $K_{i,26}$ namely and several bits of the output ciphertext, Y_3, Y_8, Y_{14}, and Y_{25} are used to derive the equation as below[143] ;

$$X_{17} \oplus Y_3 \oplus Y_8 \oplus Y_{14} \oplus Y_{25} = K_{i,26}$$

As denoted in Figure 3.16, the four output bits from the S-box S5 are c_{17}, c_{18}, c_{19}, c_{20}. The input to the S5 is the bit b_{26} and this input bit is derived from the XOR of the a_{26} , the output bit from the expansion permutation, with the subkey bit $K_{i,26}$. Tracing back further, the corresponding input bit to the expansion permutation for a_{26} is extracted as X_{17}, which can also be seen from the same figure. Also, the four output bits from S5 are processed through the P-box so as to get the resultant four ciphertext bits which are used in the linear approximation.

This linear approximation is proven to be achieved with a probability 3/16, the best probability or the bias achieved so far among all the S-boxes in DES.[144]

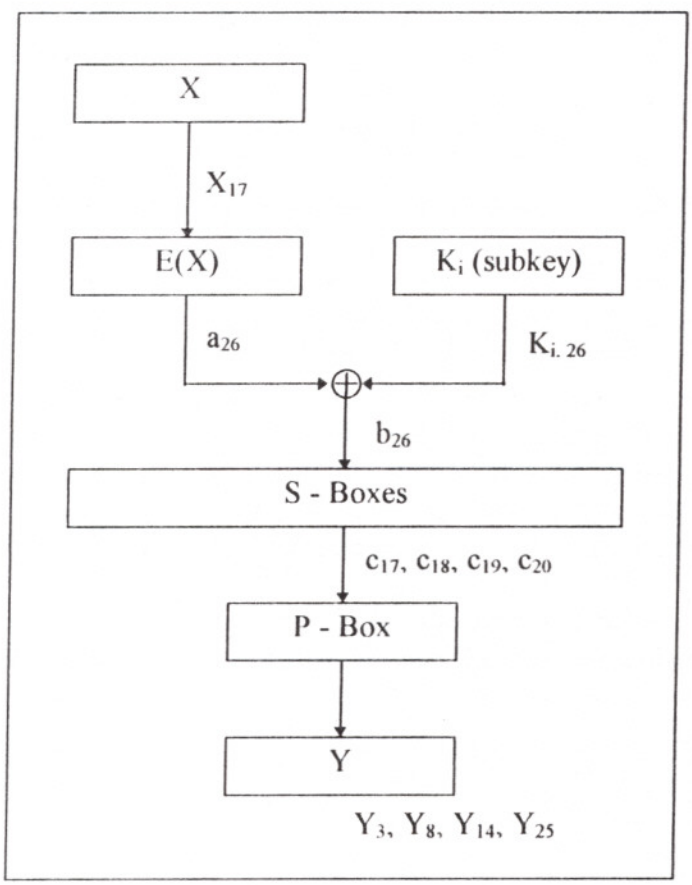


Figure 3.16 A sample one-round linear approximation for DES.[145]

[143] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 292.
[144] *ibid*, p. 291.

In order to establish a suitable attack the linear approximations of S-boxes are combined which form a compound expression with a product probability including several S-boxes. This operation is achieved by a special method known as *piling-up lemma* which is invented by *Matsui*. This compound linear approximations only give information for any single round. But, using the same logic derived from the piling-up lemma, several rounds of DES can be concatenated together so as to form linear characteristics with known probabilities. These characteristics are also similar to the ones in differential cryptanalysis in the basis and the purpose they are used for. On the other hand, linear characteristics are different in implementation and details which will be given in the following section.

Having derived the characteristics, the rest of the attack is trivial. By using these characteristics, some bits of the key can be deduced with a computed probability for any rounds of the target algorithm. The rest of the key bits can be found out by the exhaustive search, or by using some additional mechanisms which are very similar to the ones in the differential cryptanalysis. For instance, *Matsui* proved that some key bit guessing techniques can be adapted to the linear cryptanalytic attack which are known as *1R* and *2R-methods*, just similar to the *1R* and *2R-attacks* used in differential cryptanalysis. But this time, the mechanism is based on the identification of some bits of a linear approximation that depend for their value on a small subset of bits in the targeted key. Thus, it can be assumed that only by making a correct guess for these key bits, it' s sufficient to detect the anticipated bias in certain bits of the plaintext / ciphertext pair.[146]

3.2.2 Definitions and The Basic Model

In this section, the basic model of the linear cryptanalysis will be given briefly with all the necessary terms, theorems, definitions, etc. However, it should also be noted that this model is the one derived by its inventor, *Matsui*; but there are several approaches alternative to this model, such as using I / O sums for the generalization, applying non-linear approximations, and even combining with differentials such as differential-linear cryptanalysis, and so on. Since, linear cryptanalysis is a newer and more flexible attack type than differential cryptanalysis, the basic method is prone to some changes as time goes by. It should also be remarked that since the first implementations and mechanisms involving linear cryptanalysis were established for DES, the basic model and methodologies described here are also based on DES.

Throughout this section and for the rest of this chapter, some notations are used within some formulas and equations. These are described here so as to make it clear and easy to follow.

---

[145] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 291.

[146] Lars R. Knudsen, Matt Robshaw, "Non-linear Approximations in Linear Cryptanalysis", Advances in Cryptology - Proc. EUROCRYPT'96, p. 225, 1996.

For any data $\aleph$, where $\aleph$ can be: P for any plaintext block
C for any ciphertext block
K for any (sub)key;
X for any input block of f function
Y for any output block of f function
$Sk(x)$ output data x from any S-box that $k = 1,2 \ldots 8$

$\aleph_r$	stands for	$\aleph$ in the r th round
$\aleph^{[i]}$	stands for	the i^{th} bit of $\aleph$
$\aleph^{[i,j,k]}$	stands for	$\aleph^{[i]} \oplus \aleph^{[j]} \oplus \aleph^{[k]}$
$\aleph^{[i_1,i_2 \ldots i_n]}$	stands for	$\aleph^{[i_1]} \oplus \aleph^{[i_2]} \ldots\ldots \oplus \aleph^{[i_n]}$
$\aleph^{[i]} \succ \aleph^{[n]}$	stands for	$\aleph^{[i]}, \aleph^{[i+1]}, \aleph^{[i+2]}, \ldots, \aleph^{[n]}$

As mentioned previously, the first step in *Matsui*' s linear cryptanalytic attack is to find a linear approximation of DES cipher with some probability $p \neq 0.5$. This linear approximation can be expressed as below[147] ;

$$P^{[i_1,i_2 \ldots i_a]} \oplus C^{[j_1,j_2 \ldots j_b]} = K^{[k_1,k_2 \ldots k_c]} \qquad (1)$$

The equation (1) holds for $0 \leq a,b \leq 64$ and $0 < c \leq 56$, since in DES, data are encrypted in 64-bit blocks and the key size is 56 bits.

For the denoted equation (1), each of the left and right sides represent a single bit which suggests the probability of the equation to be true or false by a value of $|p - \frac{1}{2}|$.[148] The linear approximations for each of the S-boxes are derived in such a similar manner, including any combination of sets of bits that provide a probability value for each of the 64*16 entries in that S-box' s linear approximation table.[149]

Matsui derived two algorithms to extend the equation (1) to the whole of the encryption system and to calculate the required data so as to suggest correct guesses for right and left sides of the equation within a calculated success rate. The algorithms, equations and the relevant information given below are all extracted from the study of *Shahram Bakhtiari*.

Algorithm 1:

Let T be the number of plaintexts such that the left part of the equation (1) is 0 (having a zero parity);

Let N be the number of plaintexts;

---

[147] Shahram Bakhtiari, "Linear Cryptanalysis of DES Cipher", University of Wollongong, The Report for Master of CS Degree, p. 8, July 1, 1994.

[148] *ibid*, p. 8.

[149] Eli Biham, "On Matsui' s Linear Cryptanalysis", Technion- Israel Institute of Technology, Technical Report, p. 2, 1994.

brings the necessity of linear approximation tables of S-boxes and the linear approximations of S-boxes. By combining these linear approximations, the actual approximation for any n rounds will be delivered aftermath.

- **Linear Approximation of S-boxes and Linear Approximation Tables**

The linear approximation of S-boxes finds a linear relation between input bits and output bits of S-boxes.[150] These linear relations could be derived for each S-box by choosing a subset of the input bits and the output bits, calculating the XOR (parity) of these bits for each of the possible 64 inputs of that S-box and counting the number of inputs whose subset' s parity is 0. This implies that if the S-box is linear in the bits of the subset, all the inputs must have a zero parity of the subset, in other words the XOR combination of all those bits must end with a 0 value. Conversely, if the S-box has a non-linearity, is affline in the bits of the subset namely, all the inputs must have a parity 1. It should be mentioned that whenever the number of zeroes come closer to the number of ones within any subset, that subset is proven to be more non-linear.[151]

Matsui calculated the number of zero parities for each of the 64*16 entires, which are the possible subsets of the input and the output bits of each of the S-box. This calculation can be put into a formal definition as follows[152] ;

For any S-box Sk (k = 1,...8), $0 \le \alpha \le 63$ and $0 \le \beta \le 15$, it could be defined that $NSk(\alpha,\beta)$ is to be the total number of inputs out of 64 possible input values which suggests that an XORed value of the input bits masked by α is equivalent to an XORed value of the output bits masked by β. The formulation of this statement is given as below where • stands for the bitwise AND operation, x is the input to Sk;

$$NSk(\alpha,\beta) \Rightarrow \#\left\{x|0 \le x \le 64, \left(\bigoplus_{s=0}^{5}\left(x^{[s]} \bullet \alpha^{[s]}\right)\right) = \left(\bigoplus_{t=0}^{3}\left(Sk^{[t]}(x) \bullet \beta^{[t]}\right)\right)\right\}. \quad (3)$$

In other words, α and β are the bitwise vectors which are used to derive the scalar products with the S-box input and output bits via AND operation; and consequently, the outputs of these scalar products are the subsets of bits that will be XORed with each other to provide the necessary parities.[153] In fact, this definition gives the inner structure of the scheme generally preferred while implementing the linear approximations. Using this definition, tables within 64*16 entries can be constructed for each of the eight S-boxes and they are named as linear approximation tables.[154] (some refer as linear distribution tables.[155]) Therefore, it can be concluded

---

[150] Shahram Bakhtiari, "Linear Cryptanalysis of DES Cipher", University of Wollongong, The Report for Master of CS Degree, p. 8, July 1, 1994.

[151] Eli Biham, "On Matsui' s Linear Cryptanalysis", Technion- Israel Institute of Technology, Technical Report, p. 2, 1994.

[152] Shahram Bakhtiari, "Linear Cryptanalysis of DES Cipher", University of Wollongong, The Report for Master of CS Degree, p. 9, July 1, 1994.

[153] Ronald L. Rivest, Scott Contini, M. J. B. Robshaw, Yiqun Lisa Yin, "The Security of the RC6 Block Cipher", RSA Laboratories, Technical Report, p. 42, 1998.

[154] Eli Biham, "On Matsui' s Linear Cryptanalysis", Technion- Israel Institute of Technology, Technical Report, p. 2, 1994.

that the linear approximation table of any S-box describes all the possible values, hence the probabilities for all the possible subsets of that S-box.[156]

As an example, the linear approximation table of S5 is given in the Table 3.11. An important fact should be stated; which is that all the values for the entries in the linear approximation tables are subtracted from 32, valid for the table of S5 as well. This is done so as to represent the subsets' linearity in a simple manner by subtracting the number of half of the inputs (64/2 = 32) from the values of subsets. By this way, it can be easily analyzed from the tables that the 0-value entries suggest purely non-linear subsets, whereas high negative or positive values suggest linear (affline) or close to linear subsets.[157] In other words, for the basic linear cryptanalytic attack the difference between $NSk(\alpha,\beta)$ and the value 32 is the crucial point which provides the distance between $NSk(\alpha,\beta)$ and mid-point 32 (the distance of non-linearity), where only a few entries are proven to be far from 32 among all the S-boxes.[158]

Analyzing Table 3.11, some important properties of S5 related with linear cryptanalysis can be figured out. For instance, it can be seen that about 30% of the entries have value 0 implying that S-box S5 has entirely non-linear subsets which are useless for the linear cryptanalysis by a percentage of 30. It could be further analyzed from the same table that the highest difference entry value is *-20* which provides the subset with the highest bias or the highest linearity among S5. This is the entry (10_x, F_x) where 10_x is the input subset mask (vector) and F_x is the corresponding output subset mask of the bits for S5. In other words, this shows that out of the 64 possible XOR combinations of the subsets of 6 input bits and 4 output bits of the S5, only 12 of them have a bias when the input bits are ANDed with 10_x and the output bits are ANDed with F_x. This can be identified from the equation (3); when $\alpha = 10_x$ and $\beta = F_x$, only 12 different combinations out of the 64 possible subsets satisfy the equality. Henceforth, it can be concluded that only 12 out of the 64 inputs, the parity of the four output bits is the same as the value of the second input bit which is 1. In fact, this property of S5 was found out by *Shamir* long before the invention of linear cryptanalysis, however, nobody found a way to exploit this weakness at that time. Furthermore, this specific entry, besides having the highest linearity for S5, is proven to be the most linear entry among all the S-boxes in DES.[159]

It should be noted that in Table 3.11, the distance value is denoted as *32* for the entry (0_x, 0_x). But this entry is not taken into consideration because both all the input and output bits of the S-box is 0, which provides 64 possible combinations with parity, or this entry is possible with a probability of 1, thus it' s trivial and requires no cryptanalytic analysis.

---

[155] Shahram Bakhtiari, "Linear Cryptanalysis of DES Cipher", University of Wollongong, The Report for Master of CS Degree, p. 9, July 1, 1994.

[156] Eli Biham, "On Matsui' s Linear Cryptanalysis", Technion- Israel Institute of Technology, Technical Report, p. 2, 1994.

[157] *ibid,* p. 2.

[158] Shahram Bakhtiari, "Linear Cryptanalysis of DES Cipher", University of Wollongong, The Report for Master of CS Degree, p. 9, July 1, 1994.

[159] Eli Biham, "On Matsui' s Linear Cryptanalysis", Technion- Israel Institute of Technology, Technical Report, p. 2, 1994.

Table 3.11 The Linear Approximation Table of S5. [160]

Input	Output subset															
subset	0_x	1_x	2_x	3_x	4_x	5_x	6_x	7_x	8_x	9_x	A_x	B_x	C_x	D_x	E_x	F_x
0_x	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1_x	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2_x	0	4	-2	2	-2	2	-4	0	4	0	2	-2	2	-2	0	-4
3_x	0	0	-2	6	-2	-2	4	-4	0	0	-2	6	-2	-2	4	-4
4_x	0	2	-2	0	0	2	-2	0	0	2	2	4	-4	-2	-2	0
5_x	0	2	2	-4	0	10	-6	-4	0	2	-10	0	4	-2	2	4
6_x	0	-2	-4	-6	-2	-4	2	0	0	-2	0	-2	-6	-8	2	0
7_x	0	2	0	2	-2	8	6	0	-4	6	0	-6	-2	0	-6	-4
8_x	0	0	2	6	0	0	-2	-6	-2	2	4	-12	2	6	-4	4
9_x	0	-4	6	-2	0	-4	-6	-6	6	-2	0	-4	2	-6	-8	-4
A_x	0	4	0	0	-2	-6	2	2	2	2	-2	2	4	-4	-4	0
B_x	0	4	4	4	6	2	-2	-2	-2	-2	-2	2	0	-8	-4	0
C_x	0	2	0	-2	0	2	4	10	-2	4	-2	-8	-2	4	-6	-4
D_x	0	6	0	2	0	-2	4	-10	-2	0	-2	4	-2	8	-6	0
E_x	0	-2	-2	0	-2	4	0	2	-2	0	4	2	-4	6	-2	-4
F_x	0	-2	-2	8	6	4	0	2	2	4	8	-2	8	-6	2	0
10_x	0	2	-2	0	0	-2	-6	-8	0	-2	-2	-4	0	2	10	-20
11_x	0	2	-2	0	4	2	-2	-4	4	2	2	0	-8	-6	2	4
12_x	0	-2	0	-2	2	-4	-2	-8	4	6	4	6	-2	4	-6	0
13_x	0	-6	0	2	-2	4	2	0	4	-6	4	2	-6	4	-2	0
14_x	0	4	-4	0	0	0	0	0	-4	-4	4	4	0	4	-4	0
15_x	0	4	0	-4	-4	4	-8	-8	0	0	-4	4	8	4	0	4
16_x	0	0	6	6	2	-2	4	0	4	0	6	2	2	2	0	0
17_x	0	4	-6	-2	6	-2	-4	4	4	-4	-6	2	-2	2	0	4
18_x	0	6	0	2	4	-10	-4	2	2	0	-2	0	2	4	-2	-4
19_x	0	2	4	-6	0	-2	4	-2	6	8	6	4	10	0	2	-4
$1A_x$	0	2	2	-8	-2	4	0	2	-2	0	4	2	0	-2	-2	0
$1B_x$	0	2	6	-4	-6	0	0	2	6	8	0	-2	-4	-6	-2	0
$1C_x$	0	0	-2	2	4	0	-6	2	-2	6	-4	0	2	-2	0	0
$1D_x$	0	4	-2	6	-8	0	-2	2	10	-2	-8	-8	2	2	0	4
$1E_x$	0	-4	-8	0	-2	-2	-2	2	-2	2	-2	6	4	4	4	0
$1F_x$	0	-4	8	-8	2	-6	-6	-2	-2	2	-2	-2	-8	0	0	-4
20_x	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21_x	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22_x	0	-4	-2	2	-2	2	-4	8	-4	0	-6	6	2	-2	-16	-12
23_x	0	0	-2	-2	6	-2	-4	4	0	0	-2	-2	-2	6	4	-4
24_x	0	-2	6	4	0	6	-2	4	4	-6	-2	4	0	14	2	0
25_x	0	6	2	0	0	6	2	0	-4	-6	2	-8	0	-2	6	-4
26_x	0	2	4	-2	-2	0	2	-4	4	-2	-4	-2	6	0	-2	0
27_x	0	-10	0	-2	6	4	6	-4	0	6	-12	2	2	0	6	-4
28_x	0	4	-2	-2	0	4	-6	2	2	-6	4	0	6	-2	-4	0
29_x	0	0	2	6	0	0	6	2	2	-2	-8	0	-2	-6	0	0
$2A_x$	0	0	-4	-8	6	6	6	-6	6	2	-2	-2	-8	4	-4	4
$2B_x$	0	8	0	4	6	-2	-6	6	2	6	-2	6	-4	0	4	4
$2C_x$	0	2	4	-6	0	-6	0	6	-2	-4	2	-4	-2	4	6	0
$2D_x$	0	-2	-4	-2	0	-2	-8	2	-2	0	-6	-8	-2	0	-2	4
$2E_x$	0	6	2	-4	6	4	4	-2	-10	-8	0	-2	4	-2	2	0
$2F_x$	0	6	-6	-4	6	-4	4	-2	2	4	4	-6	0	2	-2	-4
30_x	0	2	-2	0	-4	-6	-2	-4	4	2	2	0	0	2	2	4
31_x	0	2	-2	0	0	-2	2	0	0	-2	-2	-4	0	2	2	4
32_x	0	6	0	-2	-2	8	2	4	0	10	0	2	-2	4	2	0
33_x	0	-6	0	10	2	0	-2	-4	0	6	0	-10	2	4	-2	0
34_x	0	0	-12	4	-4	0	4	-8	-4	0	-4	0	-4	-4	0	0
35_x	0	-8	0	0	8	-4	4	0	0	-4	-4	0	4	4	-4	4
36_x	0	4	-2	-6	-2	-2	8	0	4	-4	-2	-2	6	2	-4	0
37_x	0	-8	-6	-6	-6	6	0	4	12	0	2	-2	2	2	4	-4
38_x	0	2	4	-6	0	-2	4	-2	-6	4	-6	0	6	4	-2	0
39_x	0	-2	8	2	-4	6	-4	-6	-2	-4	2	4	-2	0	2	0
$3A_x$	0	6	-10	0	2	4	0	-2	6	-4	0	2	4	-2	-2	-4
$3B_x$	0	-2	-6	-4	-10	0	-8	-2	-10	4	4	-2	0	2	-2	4
$3C_x$	0	-8	-6	-2	0	-4	2	2	-6	2	4	0	10	-2	4	4
$3D_x$	0	4	2	2	4	4	-2	2	-2	10	0	0	2	2	4	0
$3E_x$	0	-4	4	-4	2	2	-2	2	2	-2	-2	-2	4	-4	0	4
$3F_x$	0	-4	-4	-4	14	6	-6	-2	2	-2	6	-2	0	0	-4	0

The probabilities of such linear approximations' validity for the S-boxes are required in order to continue the attack and this is trivial. The probability can be calculated as the distance from 1/2 (32/64) within that entry.[161] For example, the probability of the most linear entry mentioned previously is 12/64, since its distance value is -20, $p' = 1/2 - 20/64 = 12/64$. Therefore, an entry with value 0 has probability

[160] Eli Biham, "On Matsui' s Linear Cryptanalysis", Technion- Israel Institute of Technology, Technical Report, p. 3, 1994.

[161] *ibid*, p. 2.

$p' = 1/2$ which shows its pure non-linearity. Another example can be observed from Table 3.11, where the entry $(2_x, 1_x)$ has a probability $p' = 1/2 + 4/64 = 9/16$ since its distance is 4.

In other words, each element of the linear approximation tables gives a linear expression based on some bits of the input and output of the S-boxes, and some bits of the subkey for any round. Recalling from the equation (3), each linear expression can be accepted true with probability denoted as below[162];

$$p' = \frac{NSk(\alpha, \beta)}{64}.$$

It was also mentioned previously that the parity of the four output bits is the same as the value of the second input bit for the entry $NS5(10_x, F_x)$ with the probability $p' = 12/64$. Using this information and analyzing the *f*-function for any single round, an extended linear approximation for S5 can be derived with the addition of relevant subkey bits and output bits of the *f*-function. The four output bits of S5 are found to be the bits *12, 13, 14* and *15* of the entire 32-bit output block of the S-boxes. After P permutation, these bits become *7, 18, 24* and *29* respectively, as the output bits of the *f*-function. A further analysis of the round function gives the information that the 4th bit of the 6-bit input to S5 is also the 22nd bit of the 48-bit input block to S-boxes. In addition, since this 22nd bit is the XORed output of the 22nd bit of the subkey K_i with the 22nd bit of the output of expansion permutation, then the related subkey bit necessary for the linear approximation is easily extracted as bit *22*. Also, the related input data block (the right 32-bit input block, denoted as X_i) bit can be found as the 15th bit because this bit becomes the 22nd bit of the expanded data block after the expansion permutation. Thus, having made all these analyzes, the linear approximation of the *f*-function for any single round *i* can be formed as the one below[163];

$$X_i^{[15]} \oplus F_i^{[7,18,24,29]} = K_i^{[22]}. \qquad (4)$$

It should be stressed that the probability of this equation is true with 12/64, which is exactly the same probability of the entry of S5 itself.[164] This deduction is trivial because this equation is a simple extension of the linear approximation of S5, where no other additional approximation, or probabilistic property is used. The additional subset of bits are only correlated with no other S-box but S5.

For any linear approximation, the required number of plaintext / ciphertext pairs can be calculated easily by using the bias of that approximation.. The distance value of any entry in the linear approximation table divided by the total number of subset combinations gives the bias or the probability of the bias. For instance, the bias

---

[162] Shahram Bakhtiari, "Linear Cryptanalysis of DES Cipher", University of Wollongong, The Report for Master of CS Degree, p. 9, July 1, 1994.

[163] *ibid*, pp. 10-11.

[164] *ibid*, p. 11.

of the linear approximation for S5 is -20/64. In fact, the bias is the difference between 1/2 and the probability p'. If the bias is denoted by e_0 then it can be shown that[165]-[166];

$$e_0 = p' - \frac{1}{2}.$$

Similarly, the bias for the linear approximation of S5 and hence, of one-round *f*-function can be found as e_0 = 12/64 - 1/2 = -20/64. Consequently, *Matsui* proved that the amount of plaintext required to exploit this bias with a high success rate is $c * e_0^{-2}$ where c is a constant dependent on the style of the attack mounted. Later on, $c = 1$ is accepted commonly so as to simplify the proposition.[167] Thus, for any linear approximation with the bias e_0, the number of known plaintexts needed to carry out a successful attack is accepted to be e_0^{-2}. [168] For instance, the linear approximation of one-round *f*-function requires about $\frac{(64)^2}{(20)^2} \approx 10$ known plaintexts to be successful.

It can be easily seen that the greater the bias, the fewer the number of plaintext / ciphertext pairs required. It' s proven that the data requirement for a linear cryptanalytic attack is inversely proportional to the square of the bias of the linear approximation used in that attack. This is issued as a noticeable and interesting fact when compared to differential cryptanalysis; since in differential cryptanalysis the data requirements are proven to be inversely proportional to the probability of the differential.[169]

When considering the required amount of data for such an attack, it should be stressed that the corresponding ciphertexts of all the known plaintexts should be available to the cryptanalyst, so it would be better to say known plaintext / ciphertext pairs rather than known plaintexts. These plaintexts can be randomly chosen, however all these plaintexts must be encrypted under the same key. Another important fact is that each of these linear approximations only find out a single bit of the key within the involved known plaintext / ciphertext data.[170]

It should be noted that, many other linear approximations can be constructed with different probabilities within any single S-box, or several S-boxes. But whenever several S-boxes are combined, a theoretical assertion is required to calculate their compound probability. Similarly, these one-round linear approximations could be extended further to any number of rounds with the requirement of a method or formula so as to calculate the overall probability. For such iterations, there exists a method

---

[165] Eli Biham, "On Matsui' s Linear Cryptanalysis", Technion- Israel Institute of Technology, Technical Report, pp. 2-4, 1994.

[166] Ronald L. Rivest, Scott Contini, M. J. B. Robshaw, Yiqun Lisa Yin, "The Security of the RC6 Block Cipher", RSA Laboratories, Technical Report, p. 41, 1998.

[167] *ibid*, p. 42.

[168] Eli Biham, "On Matsui' s Linear Cryptanalysis", Technion- Israel Institute of Technology, Technical Report, p. 4, 1994.

[169] Ronald L. Rivest, Scott Contini, M. J. B. Robshaw, Yiqun Lisa Yin, "The Security of the RC6 Block Cipher", RSA Laboratories, Technical Report, p. 41, 1998.

[170] Eli Biham, "On Matsui' s Linear Cryptanalysis", Technion- Israel Institute of Technology, Technical Report, p. 4, 1994.

based on a theoretical lemma which is proven to be valid and useful. It' s named as *piling-up lemma* and will be explained in the following paragraphs.

- **Piling-up Lemma**

As stated previously, whenever more than one S-box is combined and used within a linear approximation, or any linear approximations with different rounds are combined together, an extra mechanism is required to achieve the resultant bias or probabilities properly. *Matsui* asserted the piling-up lemma which is a simple and basic formula based on theoretical facts so as to satisfy this requirement. It can be simply described as a method and the related formula that computes the bias and hence, the compound probability of the combination of linear approximations, up to any number. The lemma can be formulated as below where p is the compound probability, l is the total number of different linear approximations of the S-boxes or of the f-functions with any number of rounds, p_i' is the probability of each linear approximation[171] ;

$$p = \frac{1}{2} + 2^{l-1} \cdot \prod_{i=1}^{l} \left(p_i' - \frac{1}{2} \right).$$

This lemma is also sometimes denoted with a similar formula, but this time bias values of each approximation, e_i ' s are used where the compound bias is e. This is also given in the following formula[172] ;

$$e = 2^{l-1} \cdot \prod_{i=1}^{l} e_i .$$

A simple example of piling-up lemma can be given when two different S-boxes with different approximations, p_1' and p_2' namely, are to be combined to form a new linear approximation. By assigning 2 to l, the compound bias or probability of the new approximation can be obtained using the piling-up lemma. Thus the probability of the approximation with two S-boxes can be found as[173] ;

$$p' = p_1' \cdot p_2' + (1 - p_1') \cdot (1 - p_2')$$

Similarly the bias of their linear approximation can be formulated as[174]-[175] ;

$$e = 2 \cdot e_1 \cdot e_2$$

---

[171] Shahram Bakhtiari, "Linear Cryptanalysis of DES Cipher", University of Wollongong, The Report for Master of CS Degree, p. 12, July 1, 1994.

[172] Eli Biham, "On Matsui' s Linear Cryptanalysis", Technion- Israel Institute of Technology, Technical Report, p. 4, 1994.

[173] *ibid*, p. 2.

[174] *ibid*, p. 4.

[175] Ronald L. Rivest, Scott Contini, M. J. B. Robshaw, Yiqun Lisa Yin, "The Security of the RC6 Block Cipher", RSA Laboratories, Technical Report, p. 42, 1998.

Consequently, the linear approximation with any number of S-boxes, or any number of different approximations of several rounds can be combined and their probabilities can be found out. Another important fact is that with the existence of piling-up lemma, the linear characteristics can be developed and used in linear cryptanalysis. However, the applicability and validity of piling-up lemma whenever the issue of key-dependency is included in the attack is still being questioned and argued. But it' s also remarked that no other lemma or alternative method is found yet, which turns out the piling-up lemma to be the best and only mechanism for calculating concatenated probabilities amongst the linear cryptanalytic attacks.[176]

- **Multiple Linear Approximations**

Another technique in the linear cryptanalysis terminology is known as multiple linear approximations. Their use is proven to be enhancing a basic linear cryptanalytic attack in some occasions. The basic idea in multiple linear approximations is to reuse the data one already been set up in a different way. In fact, the plaintext / ciphertext data is exactly the same, but the difference comes from using a different linear approximation with different subsets of bits each time. Thus, it could be made possible to extract more information about the key being searched by using a variety of linear approximations together.[177]

The most important thing that must be taken into consideration is that this technique is mostly based on the bias of the different approximations; its success heavily depends on the biases achieved from different approximations. The higher the bias, the more successful and effective the attack will be. But, if the biases are much less for the newly developed ones than the original approximation, this technique becomes useless.[178]

The fact that how these different approximations with different biases might enhance the linear cryptanalytic attack is shown to lie behind the theoretical aspects of approximations and their biases. This can be simply explained here as follows:

Given n approximations A_i with biases e_i which hold for $0 \leq i \leq n\text{-}1$. It is also known that the amount of plaintext / ciphertext required for A_0 is proportional to e_0^2. Therefore, as a best-case attack it could be assumed that the amount of data required to successfully use all n approximations (A_i for $0 \leq i \leq n\text{-}1$) is proportional to $\left(\sum_{i=0}^{n-1} e_i^2\right)^{-1}$.[179]

For instance, by using n linear approximations all having the same bias the plaintext / ciphertext requirements for that attack might be reduced by a factor of n. In

[176] Ronald L. Rivest, Scott Contini, M. J. B. Robshaw, Yiqun Lisa Yin, "The Security of the RC6 Block Cipher", RSA Laboratories, Technical Report, pp. 42-43, 1998.
[177] *ibid*, p. 43.
[178] *ibid*, p. 43.
[179] *ibid*, p. 43.

general, it' s found out that the reduction rate in data needs is around $\frac{\left(\sum_{i=0}^{n-1} e_i^2\right)}{e_0^2}$.[180]

- **Characteristics**

The characteristics that are developed and used in the differential cryptanalysis, are one of the basic schemes used in linear cryptanalysis as well. The basic methodology and the purpose of the use of the characteristics in linear cryptanalysis is similar to the differential ones that was explained in section 3.1.2 of this chapter. However, it must be stressed that there are some remarkable differences in the design and implementation of the characteristics in linear cryptanalysis. First of all, the bit differences (XORs) of the pairs were used in the characteristics of differential cryptanalysis; but in linear cryptanalysis, the bits set in the characteristics denote the subset of bits whose parity is approximated. Whenever a linear characteristic is considered, the input and output halves must not be taken as neither the actual values of bits nor the XORs of the actual values of bits.[181] Other differences between differential and linear characteristics are the implementation of iteration and concatenation of characteristics, combined probability calculations, the criteria for the choice of input data, the effect of right and left input halves, etc. which will be discussed throughout this section.

As seen in differential cryptanalysis, the simplest fundamental characteristics with the highest probability are the one-round characteristics in linear cryptanalytic attacks. Thus, a one-round linear characteristic is defined as follows;

A one-round characteristic is a tuple $(\Omega_P, \Omega_T, \Omega_K, ½ + p)$ in which $(\Omega_P)_L = (\Omega_T)_L = A'$, $(\Omega_P)_R \oplus (\Omega_T)_R = a'$ and in which $½ + p$ is the probability that a random input block P, its one-round encryption C under a random subkey K satisfies $P \cdot \Omega_P \oplus C \cdot \Omega_T = K \cdot \Omega_K$ where Ω_P is the subset of bits of the data before the round, Ω_T is the subset of bits of the data after the round and Ω_K is the subset of bits of the key whose parity is approximated.[182]

Recalling from the linear approximation issue, the p value in $½ + p$ is in fact the bias of that characteristic, where $½ + p$ is sometimes denoted as p'.

Following the definition, it' s shown that one-round linear characteristics can easily be derived, as seen in differential cryptanalysis. The easiest way with the best probabilities for one-round characteristics is suggested to be constructed with one active S-box; which implies that one has to choose a non-zero entry in one of the S-boxes and the related subsets Ω_P, Ω_T, Ω_K.[183] For example, the one-round characteristic shown in Figure 3.17 has only one active S-box, S5 namely, and in order to maximize its probability, the characteristic uses the maximal entry for S5. This provides a one-

[180] Ronald L. Rivest, Scott Contini, M. J. B. Robshaw, Yiqun Lisa Yin, "The Security of the RC6 Block Cipher", RSA Laboratories, Technical Report, p. 43, 1998.
[181] Eli Biham, "On Matsui' s Linear Cryptanalysis", Technion- Israel Institute of Technology, Technical Report, p. 4, 1994.
[182] *ibid*, pp. 4-5.
[183] *ibid*, p. 5.

round characteristic with probability 1/2 - 20/64, which is 12/64. It can be seen from the Figure 3.17, that the input a' to the f-function is $(00\ 00\ 80\ 00)_x$, and the output of the f-function is $(00\ 00\ F0\ 00)_x$ which implies the maximal entry for S5 as $(10_x, F_x)$. This subset of input / output bits of S5 result with a -20/64 bias which was explained in the previous sections and was also given in Table 3.11.

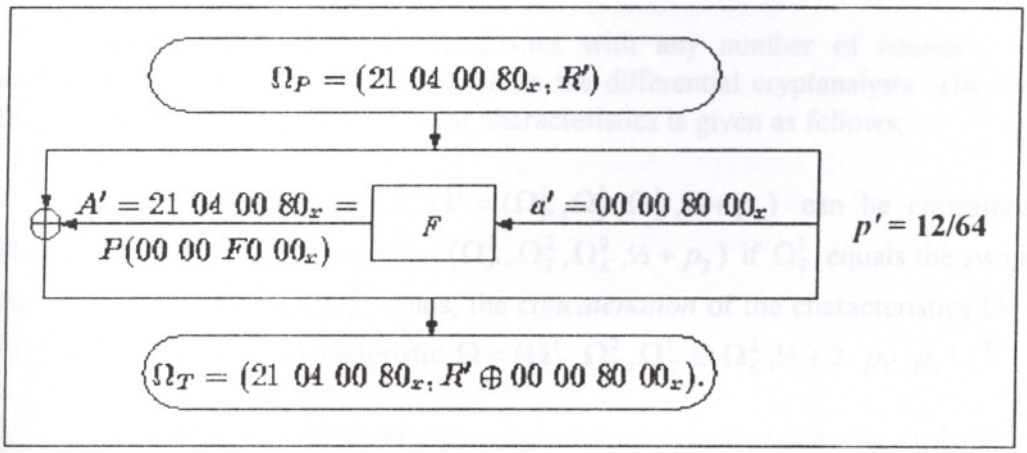


Figure 3.17 The one-round characteristic with a probability 1/2 - 20/64.[184]

In fact, this one-round characteristic is not the best among the linear ones. The best one-round characteristic is trivially known to be the one with no active S-box with a probability 1 and it' s denoted in Figure 3.18.

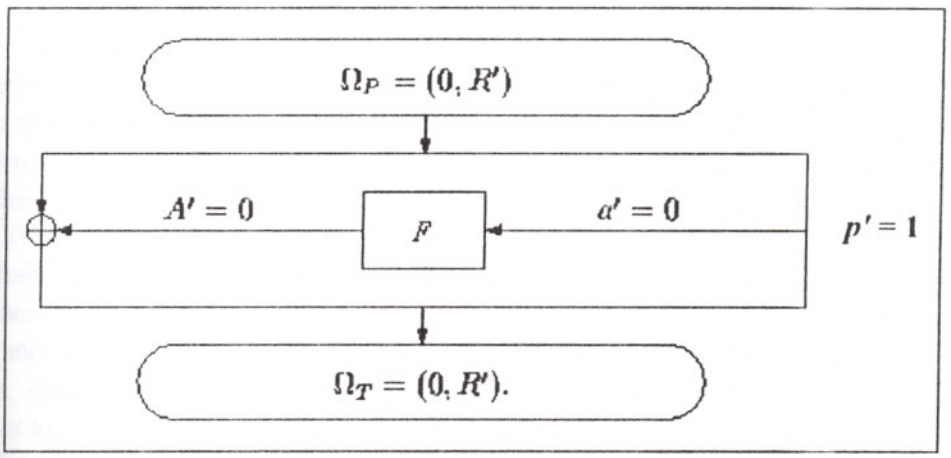


Figure 3.18 The one-round linear characteristic with a probability 1.[185]

It' s mentioned that one-round characteristics can also be constructed with more than one active S-box where the entries are chosen among two or more S-boxes and the resultant probability is calculated by the piling-up lemma easily. However, it should be stressed that unlike in differential cryptanalysis, one does not need to have the same values in common bits of both S-boxes; hence, if the bits common to two S-boxes are affected, this won' t imply that both these S-boxes are active in any linear cryptanalytic attack. Furthermore, it' s stated that if both of these S-boxes are active, the value of the common input bits become the XOR of their values from both S-

[184] Eli Biham, "On Matsui' s Linear Cryptanalysis", Technion- Israel Institute of Technology, Technical Report, p. 5, 1994.
[185] *ibid*, p. 5.

boxes, since the same bit is used twice in a linear equation, hence it (the repeated bit) cancels itself. In theory, it' s proven that the probability calculated by this way is the average between all the possible random keys. In DES, this probability is found to be true for all the keys when applied in practice and this is mentioned to be due to the design criteria of S-boxes.[186]

The concatenation of characteristics with any number of rounds can be established in linear cryptanalysis as well as the differential cryptanalysis. The formal definition of the concatenation of linear characteristics is given as follows;

An n-round characteristic $\Omega^1 = (\Omega_P^1, \Omega_T^1, \Omega_K^1, ½ + p_1)$ can be concatenated with an m-round characteristic $\Omega^2 = (\Omega_P^2, \Omega_T^2, \Omega_K^2, ½ + p_2)$ if Ω_T^1 equals the swapped value of the two halves of Ω_P^2. Thus, the *concatenation* of the characteristics Ω^1 and Ω^2 is the $(n + m)$ round characteristic $\Omega = (\Omega_P^1, \Omega_T^2, \Omega_K^1 \oplus \Omega_K^2, ½ + 2 \cdot p_1 \cdot p_2)$.[187]

When a total of l different characteristics are concatenated, the probability of the resultant characteristic, $½ + p$, can be found easily which is denoted below[188] (in fact, it' s the piling-up lemma);

$$½ + p = ½ + 2^{l-1} \cdot \prod_{i=1}^{l} p_i .$$

An interesting fact with the characteristics of linear cryptanalysis is the specific property of the structure within the rounds. For any single round of the n-round characteristic, the input subset of bits, a' namely, to the f-function is derived by XORing the subset bits of right half of the input to that round by the subset bits of right half of the output from that round. The output subset of bits from the f-function, A' is XORed with the subset bits of left half of the input to that round, to form the new right half input for the next round and the subset bits of right half of that round' s input becomes the new left half input for the next round. This causes the right halves of the input and output data of any round to be free of choice for any characteristic. In other words, since a' is the XOR of $(\Omega_P)_R$ and $(\Omega_T)_R$, then R' can be of any value, and without knowing the right halves, the cryptanalyst can freely continue the analysis with that characteristic. In contrary, the values of the $(\Omega_P)_L$ and $(\Omega_T)_L$ are of importance to the cryptanalyst and L' must be specifically chosen for the first round of any characteristic. Noticeably, this is just the opposite of the scheme in differential cryptanalysis; since for each single round of the characteristics in differential cryptanalysis the right half of the input XOR must be satisfying necessary values and a specific R' should be chosen whereas the left half of the input XOR, L', can be of any value. In fact this phenomena is stated to be one of the basic differences between linear and differential cryptanalysis.[189] This can also be analyzed from figures 3.2 and 3.18,

---

[186] Eli Biham, "On Matsui' s Linear Cryptanalysis", Technion- Israel Institute of Technology, Technical Report, pp. 5-6, 1994.

[187] *ibid*, p. 6.

[188] *ibid*, p. 6.

[189] *ibid*, p. 6.

where the free variable is at the left half in the differential cryptanalysis' characteristic and at the right half in the linear one respectively.

It should be stressed that this difference is due to the structure of the implementation in the two attacks. In differential cryptanalysis' characteristics, all the inputs to the rounds, *f*-functions and their corresponding outputs are all composed of the differences of data bits (XORs) whereas in linear cryptanalysis' characteristics, all these inputs and outputs are derived by the subsets of the data bits.[190]

Comparing the use of characteristics, another important difference between linear and differential cryptanalysis is pointed. It' s noted that this difference comes from the ability to use differentials in which only the values of Ω_P and Ω_T are in concern. In differential cryptanalysis, it' s seen that whenever several characteristics have the same values for Ω_P and Ω_T, they can be directly concatenated or iterated and consequently, they can be viewed as one differential in which the internal information can be ignored. However, in linear cryptanalysis, the internal data belonging to several characteristics contain the information about the subset of key bits participating in the linear relations and approximations. For instance, it' s proven that; if two characteristics with equivalent values of Ω_P and Ω_T and with similar probabilities exist, they might cancel the effect of each other if the parity of the subset of the key bits is not the same, or if their probabilities are the complement of each other while the parity of the subset of their key bits is the same. Thus the internal information cannot be ignored and therefore, it requires more attention and effort whenever one suggests or proposes for a linear characteristic.[191]

3.2.3 Linear Cryptanalysis of DES with Reduced Rounds

The linear cryptanalysis of DES for any number of rounds is successfully achieved with the use of necessary characteristics, linear approximations and additional schemes such as counting the key bits. In the following sections, some of the linear cryptanalytic attacks against DES reduced to some rounds will be discussed.

3.2.3.1 DES Reduced to Three Rounds

Using some linear approximations for the *f*-function of DES and by concatenating these approximations, the linear cryptanalytic attack to three-round variant can be carried out easily. The scheme denoted here uses the linear approximation given in equation (4) for the first and third rounds, and the linear approximation for the second round is derived from these two approximations. As a result, combining these linear approximations, the resultant linear approximation and the compound probability can be achieved.[192]

---

[190] Eli Biham, "On Matsui' s Linear Cryptanalysis", Technion- Israel Institute of Technology, Technical Report, p. 6, 1994.

[191] *ibid*, pp. 6-7.

[192] Shahram Bakhtiari, "Linear Cryptanalysis of DES Cipher", University of Wollongong, The Report for Master of CS Degree, pp. 11-12, July 1, 1994.

The linear approximation for the first and last of rounds of three-round DES is established as follows;

$$X_1^{[15]} \oplus F_1^{[7,18,24,29]} = K_1^{[22]}$$
$$X_3^{[15]} \oplus F_3^{[7,18,24,29]} = K_3^{[22]}$$

As can be analyzed from the Figure 3.19, in the first round, the output of the *f*-function (F_1) is XORed with the left half of the input plaintext (denoted as P_H in the figure) to become the input X_2 for the second round. Also, the right half of the input plaintext (denoted as P_L in the figure) is the input to the *f*-function, X_2 namely, for the first round. Similarly, the left half of the resultant ciphertext (C_H) is produced by XORing the output of the *f*-function in the third round (F_3) with X_2 where the right half of the resultant ciphertext (C_L) is also the input to the *f*-function in round three, X_2 namely. These can all be denoted as follows;

$$X_2 = F_1 \oplus P_H, \qquad \text{thus} \quad F_1 = X_2 \oplus P_H$$
$$C_H = F_3 \oplus X_2, \qquad \text{thus} \quad F_3 = X_2 \oplus C_H$$
$$X_1 = P_L$$
$$X_3 = C_L$$

These equalities can be derived easily by the very nature of the DES algorithm. Using these for the linear approximations of the 1st and 3rd rounds, these approximations can be rewritten as follows which discard the F_i ' s from the linear relations;

$$X_2^{[7,18,24,29]} \oplus P_H^{[7,18,24,29]} \oplus P_L^{[15]} = K_1^{[22]}$$
$$X_2^{[7,18,24,29]} \oplus C_H^{[7,18,24,29]} \oplus C_L^{[15]} = K_3^{[22]}$$

These two equations can be combined together via XOR operation which discards the X_2 variable and produces a single equation. Henceforth, this equation gives the linear approximation of three-round DES which is denoted as follows;

$$P_H^{[7,18,24,29]} \oplus C_H^{[7,18,24,29]} \oplus P_L^{[15]} \oplus C_L^{[15]} = K_1^{[22]} \oplus K_3^{[22]}$$

Using the piling-up lemma with the two probabilities (which is the same probability: 12/64) derived from the linear approximation table of S5, the probability of the linear approximation for three-round DES can be obtained as follows[193] ;

$$p = \frac{1}{2} + 2^{2-1} \cdot \prod_{i=1}^{2} \left(P_i - \frac{1}{2} \right) = \frac{1}{2} + 2\left(\frac{12}{64} - \frac{1}{2}\right)\left(\frac{12}{64} - \frac{1}{2}\right) = 0.6953124$$

Thus, the single bit of the keys for the first and third rounds can be obtained with a probability around *0.7*. This implies that the linear cryptanalytic attack against

[193] Shahram Bakhtiari, "Linear Cryptanalysis of DES Cipher", University of Wollongong, The Report for Master of CS Degree, p. 12, July 1, 1994.

three-round DES can be mounted successfully within the requirement of *25* plaintext / ciphertext pairs to find the single key bit. (Since the probability is 0.7, this implies a bias of 0.7 - ½ = 0.2, and the data requirement is $\frac{1}{(0.2)^2}$)

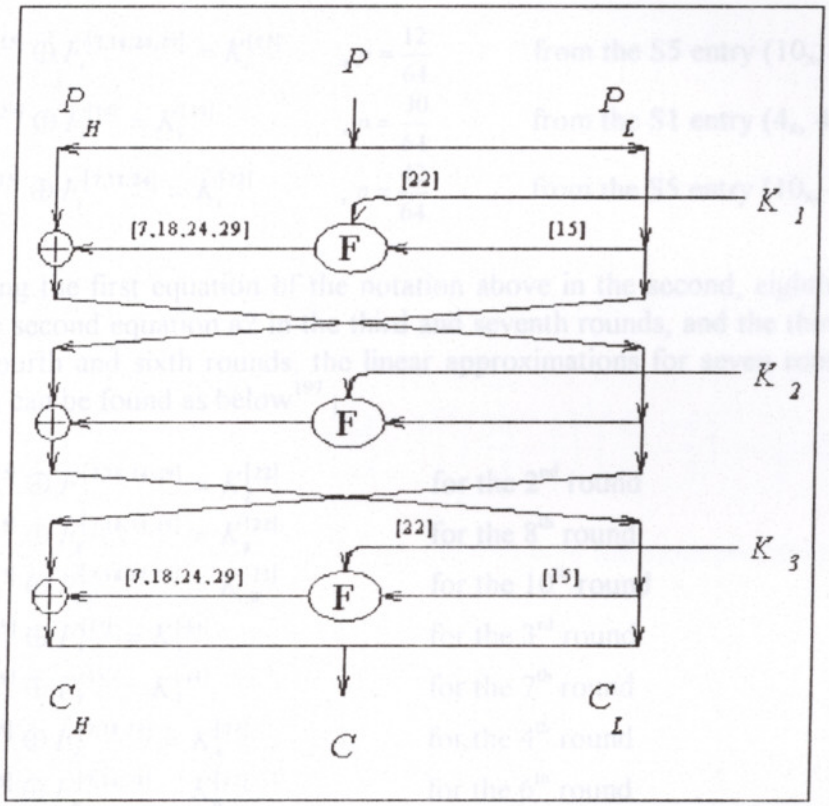


Figure 3.19 Linear Cryptanalysis of three-round DES.[194]

It' s worth to mention that the linear approximations denoted in the Figure 3.19 is, at the same time, a three-round characteristic for the linear cryptanalysis of DES reduced to three rounds.

3.2.3.2 DES Reduced to Ten Rounds

The linear cryptanalysis of ten-round DES can be obtained by using the similar logic and schemes explained in the attack to three-round DES. Some necessary linear relations are derived for some of the rounds which are proposed to be the best approximations achieved so far. By combining these approximations, a final linear approximation for the ten-round DES is constructed with a resultant probability. This 10-round linear approximation is also used for the linear cryptanalysis of 12-round DES, which will be explained in the following section. The linear cryptanalysis of 10-round DES is also included in the Figure 3.20, so it can also be analyzed from that figure[195], where the input, output and the key bits used in the linear approximations are denoted.

[194] Shahram Bakhtiari, "Linear Cryptanalysis of DES Cipher", University of Wollongong, The Report for Master of CS Degree, p. 11, July 1, 1994.
[195] *ibid*, p. 15.

The core of the 10-round linear approximation is shown to be three linear approximations derived from the S-boxes S1 and S5. Using the entries and the probabilities provided from the linear approximation tables of these two S-boxes, the following approximations can be derived[196] ;

a1: $X_i^{[15]} \oplus F_i^{[7,18,24,29]} = K_i^{[22]}$, $p = \frac{12}{64}$ from the S5 entry (10_x, F_x)

a2: $X_i^{[29]} \oplus F_i^{[15]} = K_i^{[44]}$, $p = \frac{30}{64}$ from the S1 entry (4_x, 4_x)

a3: $X_i^{[15]} \oplus F_i^{[7,18,24]} = K_i^{[22]}$, $p = \frac{42}{64}$ from the S5 entry (10_x, E_x)

Using the first equation of the notation above in the second, eighth and tenth rounds, the second equation a2 in the third and seventh rounds, and the third equation a3 in the fourth and sixth rounds; the linear approximations for seven rounds of 10-round DES can be found as below[197] ;

$X_2^{[15]} \oplus F_2^{[7,18,24,29]} = K_2^{[22]}$ for the 2nd round

$X_8^{[15]} \oplus F_8^{[7,18,24,29]} = K_8^{[22]}$ for the 8th round

$X_{10}^{[15]} \oplus F_{10}^{[7,18,24,29]} = K_{10}^{[22]}$ for the 10th round

$X_3^{[29]} \oplus F_3^{[15]} = K_3^{[44]}$ for the 3rd round

$X_7^{[29]} \oplus F_7^{[15]} = K_7^{[44]}$ for the 7th round

$X_4^{[15]} \oplus F_4^{[7,18,24]} = K_4^{[22]}$ for the 4th round

$X_6^{[15]} \oplus F_6^{[7,18,24]} = K_6^{[22]}$ for the 6th round

By the structure of the DES algorithm, the following deductions can also be made along the ten rounds;

$$F_2 = P_L \oplus X_3$$
$$F_i = X_{i-1} \oplus X_{i+1}, \quad \text{for } 3 \le i \le 9$$
$$F_{10} = X_9 \oplus C_H$$
$$X_{10} = C_L$$

Inserting these equalities in the former seven linear approximations, the linear approximations for these rounds can be rewritten as follows[198] ;

$$X_2^{[15]} \oplus P_L^{[7,18,24,29]} \oplus X_3^{[7,18,24,29]} = K_2^{[22]}$$
$$X_8^{[15]} \oplus X_7^{[7,18,24,29]} \oplus X_9^{[7,18,24,29]} = K_8^{[22]}$$
$$C_L^{[15]} \oplus X_9^{[7,18,24,29]} \oplus C_H^{[7,18,24,29]} = K_{10}^{[22]}$$
$$X_3^{[29]} \oplus X_2^{[15]} \oplus X_4^{[15]} = K_3^{[44]}$$

[196] Shahram Bakhtiari, "Linear Cryptanalysis of DES Cipher", University of Wollongong, The Report for Master of CS Degree, p. 13, July 1, 1994.
[197] *ibid*, p. 13.
[198] *ibid*, p. 14.

$$X_7^{[29]} \oplus X_6^{[15]} \oplus X_8^{[15]} = K_7^{[44]}$$
$$X_4^{[15]} \oplus X_3^{[7,18,24]} \oplus X_5^{[7,18,24]} = K_4^{[22]}$$
$$X_6^{[15]} \oplus X_5^{[7,18,24]} \oplus X_7^{[7,18,24]} = K_6^{[22]}$$

XORing all these seven approximations, the compound linear approximation for the ten-round DES is achieved as[199] ;

$$P_L^{[7,18,24,29]} \oplus C_H^{[7,18,24,29]} \oplus C_L^{[15]} = K_2^{[22]} \oplus K_3^{[44]} \oplus K_4^{[22]} \oplus K_6^{[22]} \oplus K_7^{[44]} \oplus K_8^{[22]} \oplus K_{10}^{[22]}$$

This is stated to be one of the best linear approximations for 10-round DES found so far. Using the piling-up lemma, the probability of this approximation can be calculated as[200] ;

$$p = \frac{1}{2} + 2^{7-1} \cdot \prod_{i=1}^{7} \left(p_i - \frac{1}{2} \right) = \frac{1}{2} + 2^6 \left(\frac{12}{64} - \frac{1}{2} \right)^3 \left(\frac{30}{64} - \frac{1}{2} \right)^2 \left(\frac{42}{64} - \frac{1}{2} \right)^2 = \frac{1}{2} - 1{,}53 \times 2^{-15}$$

It' s proven that another best approximation with the same probability can also be derived in a similar way exploiting the round symmetry of the DES algorithm. By applying the linear relation in a1 to the first, third and ninth rounds, a2 to the fourth and eighth rounds and a3 to the fifth and seventh rounds of DES and by making the replacements and combining the seven approximations via XOR, the second best linear approximation for 10-round DES can be found as[201] ;

$$C_L^{[7,18,24,29]} \oplus P_H^{[7,18,24,29]} \oplus P_L^{[15]} = K_9^{[22]} \oplus K_8^{[44]} \oplus K_7^{[22]} \oplus K_5^{[22]} \oplus K_4^{[44]} \oplus K_3^{[22]} \oplus K_1^{[22]}$$

The probability of this approximation is also calculated as $p = \frac{1}{2} - 1{,}53 \times 2^{-15}$ which is exactly equal to the probability of the first linear approximation for ten rounds. Thus, several bits of the keys for all the ten rounds can be retrieved by these two approximations. Any other linear approximations, but with less probabilities can be achieved using the similar scheme so as to guess the other bits of the keys.

3.2.3.3 DES Reduced to Twelve Rounds

The linear cryptanalysis of 12-round DES can be achieved by establishing the linear approximations for 12-round DES. When the basic methodology is used, in order to mount a successful attack for 12 rounds, the best linear approximations of 10-round DES are required. Since these were derived in the previously mentioned attack, the 12-round linear cryptanalytic attack can also be carried out. In other words, the linear cryptanalysis of DES reduced to 12 rounds explained here is the extension of the 10-round attack by adding a first and last round to the ten-round characteristic.

---

[199] Shahram Bakhtiari, "Linear Cryptanalysis of DES Cipher", University of Wollongong, The Report for Master of CS Degree, p. 14, July 1, 1994.
[200] *ibid*, p. 14.
[201] *ibid*, p. 14.

However, it must be stressed that all the former attacks and this attack can be carried out with different linear characteristics and schemes as well.

Both the two best 10-round linear approximations given in section 3.2.3.2 can be adapted for the rounds two to eleven to derive the linear approximations of the ten rounds of 12-round DES. Thus, the approximations are derived as follows[202] ;

$$X_2^{[7,18,24,29]} \oplus X_{12}^{[7,18,24,29]} \oplus X_{11}^{[15]} = K_3^{[22]} \oplus K_4^{[44]} \oplus K_5^{[22]} \oplus K_7^{[22]} \oplus K_8^{[44]} \oplus K_9^{[22]} \oplus K_{11}^{[22]}$$
$$X_{11}^{[7,18,24,29]} \oplus X_1^{[7,18,24,29]} \oplus X_2^{[15]} = K_{10}^{[22]} \oplus K_9^{[44]} \oplus K_8^{[22]} \oplus K_6^{[22]} \oplus K_5^{[44]} \oplus K_4^{[22]} \oplus K_2^{[22]}$$

These can also be analyzed from Figure 3.20 where the 12-round characteristic within the embedded 10-round characteristic denoting the relevant subsets of bits for each round is given. Exploiting the basic properties of DES algorithm, the following equalities, which can also be analyzed from Figure 3.20, are also shown to be valid;

$$X_2 = P_H \oplus F_1$$
$$X_{12} = C_L$$
$$X_{11} = C_H \oplus F_{12}$$
$$X_1 = P_L$$

By substituting these equalities in the two linear approximations for 12 rounds, and reconstructing the equations, the two best known linear approximations for the 12-round DES can be achieved as follows[203] ;

$$P_H^{[7,18,24,29]} \oplus F_1^{[7,18,24,29]} \oplus C_L^{[7,18,24,29]} \oplus C_H^{[15]} \oplus F_{12}^{[15]} =$$
$$K_3^{[22]} \oplus K_4^{[44]} \oplus K_5^{[22]} \oplus K_7^{[22]} \oplus K_8^{[44]} \oplus K_9^{[22]} \oplus K_{11}^{[22]}$$

$$C_H^{[7,18,24,29]} \oplus F_{12}^{[7,18,24,29]} \oplus P_L^{[7,18,24,29]} \oplus P_H^{[15]} \oplus F_1^{[15]} =$$
$$K_{10}^{[22]} \oplus K_9^{[44]} \oplus K_8^{[22]} \oplus K_6^{[22]} \oplus K_5^{[44]} \oplus K_4^{[22]} \oplus K_2^{[22]}$$

Each of these approximations are proven to be true with a probability of $1/2 - 1.53 \times 2^{-15}$, which is exactly equivalent to the probabilities achieved for the best linear approximations of 10-round DES.[204]

Shahram Bakhtiari, "Linear Cryptanalysis of DES Cipher", University of Wollongong, The Report Master of CS Degree, p. 14, July 1, 1994.

ibid, p. 16.

ibid, p. 16.

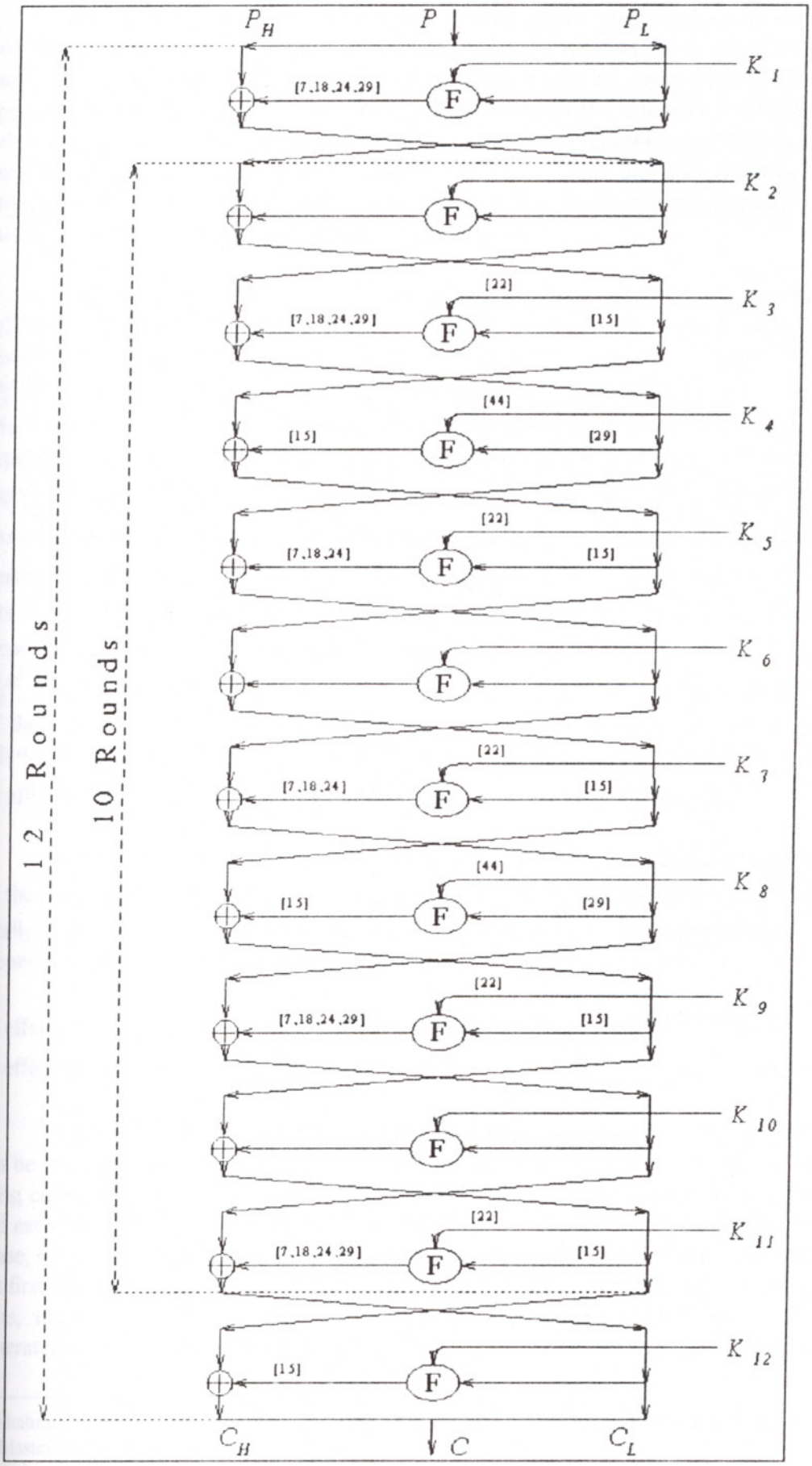


Figure 3.20 Linear Cryptanalysis of 12-round DES.

It' s stressed that in order to implement the attack successfully and efficiently, one should know how the plaintexts, ciphertexts and keys affect the result of these two linear approximations. This is necessary, because it can be seen from both of the approximations that some bits of the F are included in the equations. All the other subsets of bits in the left side of the equations are trivially known since they belong to input plaintext and output ciphertext halves. However, the F bits are unknown, which brings the necessity of implementing a mechanism to guess those bits in order to find out the key bits in the right side of the approximations.

The mechanism is derived by the use of effective bits. Effective text bits are defined as the bits that affect the left hand side of the linear approximations. For instance, considering the first 12-round linear approximation, $P_H^{[7,18,24,29]} \oplus C_L^{[7,18,24,29]} \oplus C_H^{[15]}$ can be considered as a one effective text bit. The effective bits for the unknown subsets of bits in the linear approximations can be found exploiting the structure of DES. Again, for the left side of the first 12-round linear approximation, $F_{12}^{[15]}$ is unknown. Analyzing the DES algorithm, this single bit can be found as the one output bit of S1. This implies that only the input to S1 affects $F_{12}^{[15]}$. Further analysis show that the input of S1 is derived from XORing $K_{12}^{[42]} \succ K_{12}^{[47]}$ and bits *42,43,..47* of the output of the expansion function. The input bits corresponding to those output bits of the expansion function are $X_{12}^{[27]} \succ X_{12}^{[31]}$ and $X_{12}^{[0]}$, hence, $C_L^{[27]} \succ C_L^{[31]}$ and $C_L^{[0]}$. Therefore, $C_L^{[27]} \succ C_L^{[31]}$ and $C_L^{[0]}$ are the other six effective bits of the left side of the equation. Moreover, the similar analysis can be made for $F_1^{[7,18,24,29]}$ which results with $P_L^{[11]} \succ P_L^{[16]}$ as the effective bits for $F_1^{[7,18,24,29]}$. Totally, *13* effective text bits are found for the first 12-round linear approximation.[205]

It' s proven that using the same method, the effective key bits for the right side of the same approximation can be derived as $K_{12}^{[42]} \succ K_{12}^{[47]}$ and $K_1^{[18]} \succ K_1^{[23]}$ which totally make up *12* effective keys. The same method can also be implemented for the second 12-round linear approximation which results with[206] ;

13 effective text bits: $C_L^{[11]} \succ C_L^{[16]}, P_L^{[0]}, P_L^{[27]} \succ P_L^{[31]}, C_H^{[7,18,24,29]} \oplus P_L^{[7,18,24,29]} \oplus P_H^{[15]}$
12 effective text bits: $K_1^{[42]} \succ K_1^{[47]}, K_{12}^{[18]} \succ K_{12}^{[23]}$

In practice, the method described for the linear cryptanalysis of 12-round DES can be implemented in two phases. The first phase is the data counting phase where using counters, the *13* effective text bits for the first linear approximation are derived and each bit' s value is computed. The second phase is the key counting phase. In this phase, first, new counters are used for finding the value of the *12* effective key bits of the first approximation. Then, for calculated number of required plaintext / ciphertext pairs, the key bits are guessed with the possible highest probabilities within the generated 12-round linear approximations.[207]

---

[205] Shahram Bakhtiari, "Linear Cryptanalysis of DES Cipher", University of Wollongong, The Report for Master of CS Degree, p. 16, July 1, 1994.
[206] *ibid*, p. 16.
[207] *ibid*, pp. 16-17.

It' s found that totally, $1.5*2^{13}$ counters are required so as to carry out such an attack. However, if both linear approximations are combined together then this value might increase to $1.5*2^{14}$. If the two phases mentioned previously are applied to both of the linear approximations, a total of *25* key bits are proven to be correctly guessed with a high probability. It' s also proven that the linear cryptanalysis of 12-round DES require 2^{32} pairs of plaintexts and ciphertexts when implemented with the method described here. The remaining 31 bits of the entire 56-bit key can be found by an exhaustive search. The required execution time for the complete linear cryptanalysis of 12-round DES by using the methodology described here was recorded as *120* hours when implemented on *SUN 4 SPARC* stations running *SOLARIS 2.3* operating system.[208]-[209]

3.2.4 Linear Cryptanalysis of the Full 16-Round DES

The linear cryptanalysis of the standard 16-round DES algorithm has been successfully implemented by *Matsui* et al. In some of these linear cryptanalytic attacks, the basic methodology described in the subsections of 3.2.3 are used, whereas some improved methods with additional schemes and mechanisms are derived and implemented as well.

The first known linear cryptanalytic attack against 16-round DES was implemented by *Matsui*. The basic idea in his attack was to derive a best 16-round approximation. However, rather than using best 14-round linear approximations and extending those to 16 rounds, he chose an eight-round iterative linear characteristic with a probability of $1/2 + 2^{-27}$, and concatenating this iterative characteristic with itself he managed to get a 16-round linear characteristic.[210] The eight-round iterative characteristic is denoted in the Figure 3.21.

After the extension of the eight-round characteristic to 16 rounds, *Matsui* also replaced the first and last round of the characteristic with locally better ones having higher probabilities. In the end, he managed to get a 16-round linear approximation with a probability around $1/2 + 2^{-24}$. The plaintext / ciphertext requirement for this attack is calculated to be around 2^{47}. *Matsui* claimed that the characteristic he developed for this attack was the best one without any restrictions. This assertion is based on the fact that, among the 16 rounds of the characteristic, there' s only one active S-box posing a single affected key bit at 12 rounds and no active S-box with no affected key bits at the other rounds. However, this attack is not considered so successful since only a single bit of the 56-bit key can be retrieved by this method.[211]-[212]

---

[208] S. Bakhtiari, R. Safavi-Naini, "Application of PVM to Linear Cryptanalysis", University of Wollongong, Technical Report, p. 5, July 25, 1994.

[209] Shahram Bakhtiari, "Linear Cryptanalysis of DES Cipher", University of Wollongong, The Report for Master of CS Degree, p. 18, July 1, 1994.

[210] Eli Biham, "On Matsui' s Linear Cryptanalysis", Technion- Israel Institute of Technology, Technical Report, p. 10, 1994.

[211] *ibid*, pp. 10-11.

[212] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 292.

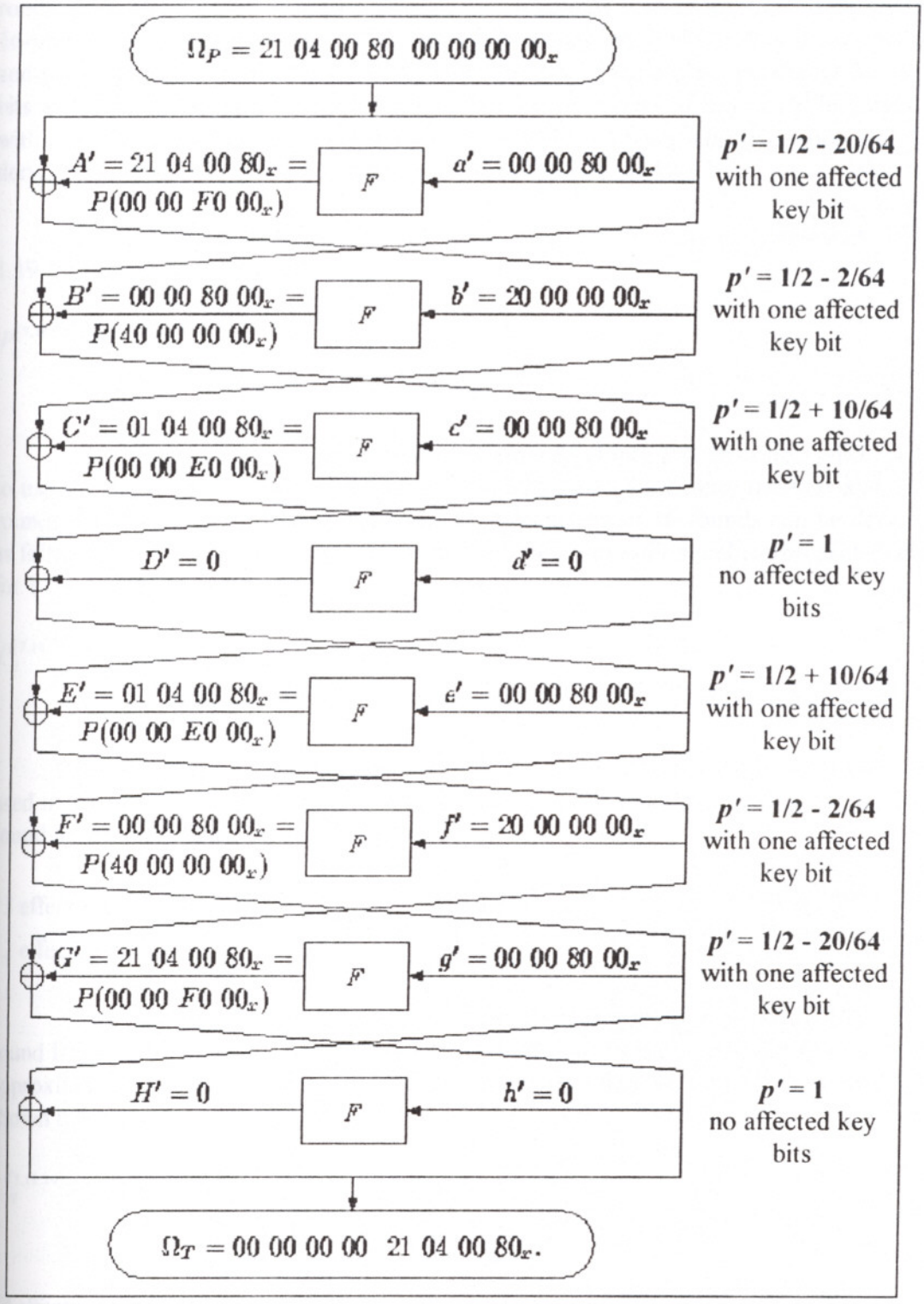


Figure 3.21 Eight-round iterative characteristic with probability $1/2 + 2^{-27}$.[213]

Later on, a much better and more powerful attack with less data requirements is successfully implemented, both in theory and practice. The basic methodology used in this refined attack is similar to the one implemented for 12-round DES. For 16-

[213] Eli Biham, "On Matsui' s Linear Cryptanalysis", Technion- Israel Institute of Technology, Technical Report, p. 10, 1994.

round DES, the best linear approximation for 14 rounds is used in order to derive the 16-round linear approximation. The effective text and key bits are derived aftermath, and using the same methodology in 12-round attack, counters are generated for data bits and key bits analysis in which some of the bits of the 56-bit key could be guessed with a high probability. The construction of the 16-round linear approximation and the derivation of the effective bits can be simply described as follows:

The best linear expression for 14-round DES, which has a probability of 1/2 - 1.19 * 2^{-21}, is denoted below[214] ;

$$P_L^{[7,18,24]} \oplus C_H^{[7,18,24,29]} \oplus C_L^{[15]} = K_2^{[22]} \oplus K_3^{[44]} \oplus K_4^{[22]} \oplus K_6^{[22]} \oplus K_7^{[44]} \oplus K_8^{[22]} \oplus K_{10}^{[22]} \oplus K_{11}^{[44]} \oplus K_{12}^{[22]} \oplus K_{14}^{[22]}$$

This 14-round linear approximation is embedded into full 16-round DES so as to use for the approximation of 2nd to 15th rounds. After combining the first and 16th rounds with this approximation, the linear approximation of 16 rounds can be derived as follows, with a probability of 1/2 - 1.19 * 2^{-21} being exactly equal to the probability for 14-round approximation[215] ;

$$P_H^{[7,18,24]} \oplus P_1^{[7,18,24]} \oplus C_L^{[7,18,24,29]} \oplus C_H^{[15]} \oplus P_{16}^{[15]} = K_3^{[22]} \oplus K_4^{[44]} \oplus K_5^{[22]} \oplus K_7^{[22]} \oplus K_8^{[44]} \oplus K_9^{[22]} \oplus K_{11}^{[22]} \oplus K_{12}^{[44]} \oplus K_{13}^{[22]} \oplus K_{15}^{[22]}$$

The basic logic in deriving the effective text and key bits is the same of the one used in the linear cryptanalysis of 12-round DES. Hence, the effective bits for the 16-round linear approximation can be found as[216] ;

13 effective text bits: $P_L^{[11]} \succ P_L^{[16]}, C_L^{[0]}, C_L^{[27]} \succ C_L^{[31]}, P_H^{[7,18,24]} \oplus C_L^{[7,18,24,29]} \oplus C_H^{[15]}$

12 effective text bits: $K_1^{[18]} \succ K_1^{[23]}, K_{16}^{[42]} \succ K_{16}^{[47]}$

Exploiting the symmetric structure of the DES algorithm, a second similar 16-round linear approximation with a probability equivalent to the one found for the first approximation can be derived as well as the related effective bits.[217] These are all shown below;

$$C_H^{[7,18,24]} \oplus P_{16}^{[7,18,24]} \oplus P_L^{[7,18,24,29]} \oplus P_H^{[15]} \oplus P_1^{[15]} = K_{14}^{[22]} \oplus K_{13}^{[44]} \oplus K_{12}^{[22]} \oplus K_{10}^{[22]} \oplus K_9^{[44]} \oplus K_8^{[22]} \oplus K_6^{[22]} \oplus K_5^{[44]} \oplus K_4^{[22]} \oplus K_2^{[22]}$$

13 effective text bits: $C_L^{[11]} \succ C_L^{[16]}, P_L^{[0]}, P_L^{[27]} \succ P_L^{[31]}, C_H^{[7,18,24]} \oplus P_L^{[7,18,24,29]} \oplus P_H^{[15]}$

12 effective text bits: $K_1^{[42]} \succ K_1^{[47]}, K_{16}^{[18]} \succ K_{16}^{[23]}$

---

[214] Shahram Bakhtiari, "Linear Cryptanalysis of DES Cipher", University of Wollongong, The Report for Master of CS Degree, p. 55, July 1, 1994.

[215] *ibid*, p. 55.

[216] *ibid*, p. 55.

[217] *ibid*, p. 55.

By implementing this methodology, it' s proven that 26 bits of the 56-bit key can be retrieved correctly with a high probability. The remaining 30 bits can be found via exhaustive search, or by some other special search algorithms for the key bits. The overall data requirement of this attack is calculated between 2^{43} - 2^{45} known plaintexts. In practice, it' s recorded that a software implementation of this method recovered the 56-bit key of the 16-round DES cipher in *50* days using twelve *HP9000/735* workstations. These results show that the linear cryptanalysis of 16-round DES using this method is much more efficient and successful and than the former method with eight-round iterative characteristics.[218]-[219]-[220]

The requirement of 2^{43} known plaintexts is also accepted to be the most successful performance among all the cryptanalytic attacks against standard DES so far. However, these values are still considered to be strongly effective in theory, not in practice; thus, the linear cryptanalysis of 56-bit DES still needs some significant improvements with a much higher feasibility as well as the differential cryptanalysis.

3.2.5 Linear Cryptanalysis of Some Other Cryptosystems

The linear cryptanalytic attacks are also carried out among some other symmetric block cryptosystems successfully which some of these will be discussed shortly in this section. The methods used, more or less, are shown to be similar to the basic methodology used in DES. However, in some of these implementations, additional schemes are derived, or different linear cryptanalysis techniques are used whenever necessary.

- **Linear Cryptanalysis of FEAL**

The basic method used in the linear cryptanalysis of FEAL is known to be originated by *Matsui*. Since FEAL is a DES-like cryptosystem, the fundamental model and the implementation used in the attack is proven to be similar to the *Matsui*' s attack to 16-round DES. Some characteristics with feasible probabilities are used in these attacks to find the key bits of FEAL for some of its rounds. It' s shown that there are *15* one-round linear characteristics in FEAL with probability $1/2 \pm 1/2$, based on the linearity of the least significant bits in the addition operation. It' s remarked that these characteristics are essential in the linear cryptanalysis, since they can be applied to any rounds of FEAL, or can be combined to form any *n*-round characteristics with probability either being 0 or 1. For instance, in his attack to FEAL-8, *Matsui* used two of these one-round characteristics to form a special three-round characteristic with probability 1, which is also denoted in the Figure 3.22. Then, this characteristic is used in rounds *3* up to *5* of the eight-round FEAL-8 in order to carry out the linear cryptanalytic attack. For the remaining rounds, *1*, *2*, and *6* up to *8*, the bits of the subkeys are revealed by using exhaustive search and mounting some other auxiliary search methods. By this way, the linear cryptanalysis of the eight-round FEAL-8 is achieved. It' s shown that this method requires 2^{28} known plaintexts with a

---

[218] Bruce Schneier, *Applied Cryptography - Second Edition*, p. 293.

[219] Shahram Bakhtiari, "Linear Cryptanalysis of DES Cipher", University of Wollongong, The Report for Master of CS Degree, p. 55, July 1, 1994.

[220] S. Bakhtiari, R. Safavi-Naini, "Application of PVM to Linear Cryptanalysis", University of Wollongong, Technical Report, p. 6, July 25, 1994.

computational complexity of 2^{50}, or 2^{15} known plaintexts with a computational complexity of 2^{64}.[221]

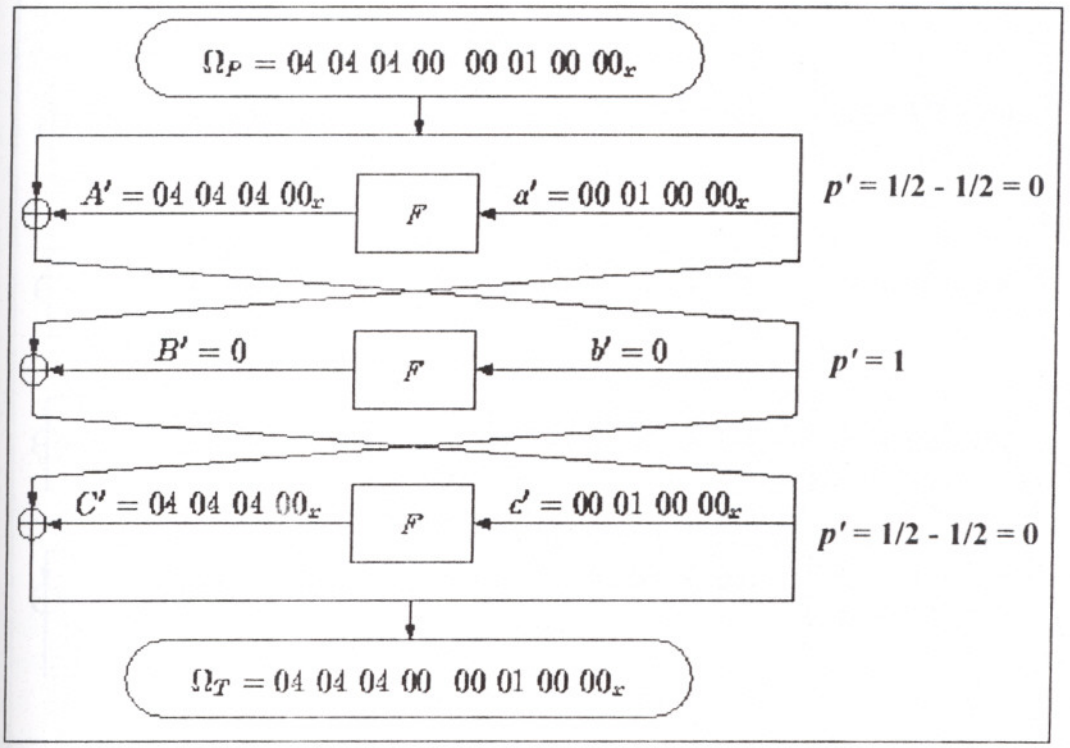


Figure 3.22 Three-round characteristic used in the linear cryptanalysis of FEAL-8.[222]

A refinement of this attack is established by *Biham* et al. This attack is basically similar to the *Matsui's*, however, instead of a three-round characteristic, they derived and used several iterative *n*-round characteristics such as a seven-round characteristic with probability $1/2 - 2^{-11}$. In fact, this characteristic is developed by iterating a special two-round iterative characteristic 3.5 times. A sample use of this two-round iterative characteristic is given in Figure 3.23, in which a four-round characteristic is formed. Using this iterative characteristic, the linear cryptanalytic attack to FEAL-8 is successfully achieved such that; 2^{24} known plaintexts are required with a success rate of 78%, and 2^{25} known plaintexts are required with a success rate of 97%. The seven-round linear characteristic mentioned is also used to attack FEAL-N up to *20* rounds successfully, requiring less amount of plaintexts and complexity than the brute-force attack.[223]

[221] Eli Biham, "On Matsui' s Linear Cryptanalysis", Technion- Israel Institute of Technology, Technical Report, pp. 11-14, 1994.
[222] *ibid*, p. 13.
[223] *ibid*, pp. 14-15.

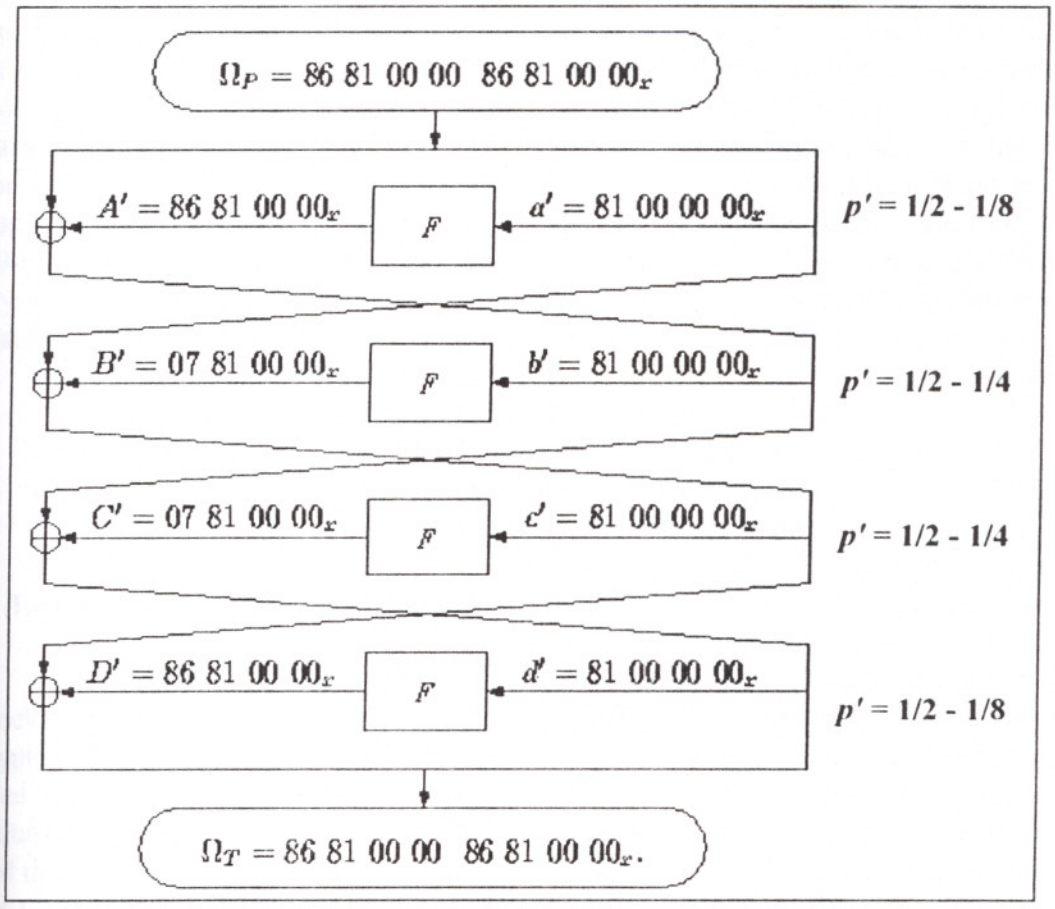


Figure 3.23 Four-round characteristic derived from the two-round iterative characteristic of FEAL-8.[224]

- **Linear Cryptanalysis of RC6**

The linear cryptanalysis of RC6 and also some of its variants such as RC6-I, RC6-NFR and RC6-I-NFR have been implemented successfully to some extent, but not better than the differential attacks. As mentioned previously in section 3.1.7, since the design and structure of RC6 algorithm is significantly different from DES, the basic methodology and the techniques used in linear cryptanalysis are different as well. The details of the design and implementation of the linear attacks are not given here; however, it should be stressed that two different types of linear approximation are used in the attacks to RC6 and its variants. *Type I* approximation uses a chosen particular type of linear approximation across the data-dependent rotation. On the other hand, *Type II* approximation involves a different style of approximation across the data-dependent rotation which leads to approximations across the fixed rotation and the quadratic function.[225]

According to the results achieved among the attacks to several variants of RC6, it' s stated that *Type I* approximations are far more efficient than *Type II*. However, some of the linear cryptanalytic attacks against several RC6 variants are proven to be

[224] Eli Biham, "On Matsui' s Linear Cryptanalysis", Technion- Israel Institute of Technology, Technical Report, p. 15, 1994.
[225] Ronald L. Rivest, Scott Contini, M. J. B. Robshaw, Yiqun Lisa Yin, "The Security of the RC6 Block Cipher", RSA Laboratories, Technical Report, p. 44, 1998.

much less efficient than brute-force or differential versions, yet it' s stressed that most of the RC6 variants with higher rounds are strongly resistive to linear cryptanalysis. For instance, using *Type I* approximation, the linear cryptanalysis of RC6 with the basic linear attack methodology require 2^{182} plaintexts for 20 rounds and 2^{222} plaintexts for 24 rounds. Similarly, using *Type I* approximation again but with multiple linear approximations this time, the attack can be implemented successfully 2^{171} plaintexts for 20-round and 2^{211} plaintexts for 24-round models of RC6. When the linear cryptanalytic attack is implemented using the *Type II* approximation, the following plaintext data requirements are observed for each of the RC6 variants[226] ;

RC6:	2^{302} for 20 rounds, 2^{342} for 24 rounds,
RC6-I:	2^{192} for 20 rounds, 2^{224} for 24 rounds,
RC6-I-NFR:	2^{182} for 20 rounds, 2^{212} for 24 rounds,
RC6-NFR:	2^{182} for 20 rounds, 2^{212} for 24 rounds.

- **Linear Cryptanalysis of LOKI 91**

LOKI91 is a DES-like symmetric block cryptosystem that operates on 64-bit blocks and uses a 64-bit key. It' s proven that LOKI91 is strongly resistive to linear cryptanalytic attacks with the basic methodology and techniques similar to the ones used in DES, by the studies of *Tokita* et al. For instance, the known plaintext requirements for 4, 7 and 10-round LOKI91 is found to be 2^{23}, 2^{40}, 2^{58}, respectively and these are shown to be not much better than other cryptanalytic attacks. Moreover, for 13 and 16-round LOKI91, all the regular linear cryptanalytic attacks are proven to be infeasible due to the fact that the plaintext requirements for linear cryptanalysis increase substantially whenever more rounds are added to the cipher.[227]

On the other hand, using an entirely different scheme named as *non-linear* approximations[228], the linear cryptanalysis of LOKI91 is proven to be achieved successfully with much less data requirements and computational complexity. In fact, the non-linear approximations are used instead of the 1R-attack and 2R-attack schemes used in the basic model of linear cryptanalysis so as to guess more number of bits with less plaintext / ciphertext pairs. It' s proven that by embedding the non-linear approximations into linear cryptanalytic attacks, *20* bits of the 64-bit LOKI91 key can be recovered instead of *13* bits with the equivalent probability of success, and the data requirements are significantly reduced as well. (non-linear approximations require less than ¼ of the plaintext data needed for the standard linear cryptanalysis.)[229]

---

[226] Ronald L. Rivest, Scott Contini, M. J. B. Robshaw, Yiqun Lisa Yin, "The Security of the RC6 Block Cipher", RSA Laboratories, Technical Report, pp. 48-59, 1998.

[227] Lars R. Knudsen, Matt Robshaw, "Non-linear Approximations in Linear Cryptanalysis", Advances in Cryptology - Proc. EUROCRYPT'96, p. 231, 1996.

[228] *ibid*, pp. 225-231.

[229] *ibid*, pp. 231-236.

Chapter 4

GENETIC ALGORITHMS

4.1 Introduction

Genetic algorithm (GA) technology has been around for over 30 years. However, applications based on this technology are just recently being produced. Genetic Algorithms (GA' s) were developed by *Prof. John H. Holland* and his students at the *University of Michigan* during the 1960s and 1970s. Essentially, they are a group of *breeding* computer programs and solutions for optimization or search problems by means of simulated evolution. In fact, these algorithms are based on the idea that seek to achieve systems which maintain a population of potential solutions, which have some selection process within fitness of individuals, and some recombination operators. Processes loosely based on natural selection, crossover, and mutation are repeatedly applied to a population of binary strings (or other data structures, whenever necessary) which represent potential solutions. Within the time of progress, the number of above-average individuals increases, and highly-fit *building blocks* are combined from several fit individuals to find good solutions to the problem at hand. In some references, however, such similar algorithms are named as *Evolutionary Programming* or *Evolutionary Algorithms*[1]. In fact all of these methodologies are based on the same evolutionary basics and disciplines, whereas, *Genetic Programming*[2] is a different and recently developing technology. In short, *Evolutionary Programs* (EP), can be used as a common term for all evolution-based systems.[3]

Genetic Programming (GP) is simply defined as GA' s applied to programs. Genetic Programming is more expressive than fixed-length character string GA' s, though GA' s are likely to be more efficient for some classes of problems. The first and basic model of GP has been founded by *J.R. Koza,* who proposed an evolution based system to search for the most fit computer program to solve a particular problem. The main difference between genetic programming and genetic algorithms is the representation of the solution since, genetic programming creates computer programs in the Lisp or scheme-like computer languages as the solution.[4-5]

---

[1] Zbigniew Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs,* pp. 1-2.
[2] Genetic Algorithms FAQ, Internet Document, http://www.cs.cmu.edu/Groups/AI/html/faqs/ai/genetic/part2/faq-doc-6.html, 1997.
[3] Zbigniew Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs,* pp. 1,8-9.
[4] Genetic Algorithms FAQ, Internet Document, http://www.cs.cmu.edu/Groups/AI/html/faqs/ai/genetic/part2/faq-doc-6.html, 1997.
[5] Jaime Fernandez, "The Genetic Programming Tutorial Notebook", Internet Document, http://www.geneticprogramming.com/Tutorial/index.html, 1997.

4.2 Definition of a Genetic Algorithm

4.2.1 Background

The Genetic Algorithm concept is formed upon the basics of Biology, Evolution, Genetics, and hence, nature. It is a model of machine learning which derives its behaviour from a metaphor of the processes of evolution in nature. This is done by the creation of a population of individuals within a machine that is represented by chromosomes, a set of character strings analogous to the base-4 chromosomes that is seen in our own DNA. (Since, in any individual DNA polynucleotide, 4 types of nucleotides exist: Adenine (A), Guanine (G), Cytosine (C), Thymine (T). These can be chained in any combination of sequence. Thus, this 4-letter alphabet can form 4^l different sequences of nucleotides of length l. This is the variability of DNA that enables the genetic material to exist in an almost infinite number of forms.[6]) The individuals in the population then go through a process of evolution.

It should be stressed that evolution (in nature or anywhere else) is not an intentional or directed process. That is, there is no evidence to support the assertion that the goal of evolution is to produce Mankind, in other words, the assertion, which once had been proposed by *J. B. Lamarck* that living beings change by consciously willing to change[7] is out of date. Also, *the inheritance of acquired characteristics*, a theory of evolution which once had been founded by *J. B. Lamarck*[8] has turned out to be false and unacceptable. Indeed, the processes of nature seem to converge into different individuals competing for resources in the environment in which some are better than others. Those that pose better characteristics do have a higher probability to survive and propagate their genetic material. This is, in fact, one of the basic theories of evolution, proposed once by *Charles Darwin*, and which is still valid today. Also, during evolution, in nature, there are various forces such as chemical agents, strains, contacts, temperature changes, etc. that act on an individual. As far as these forces work changes in the organism, the changes may be considered largely fortuitous or accidental, named as *physico-genetic*.[9]

In nature, what occurs at the molecular level is that a pair of chromosomes bump into one another, exchange chunks of genetic information and drift apart. This is the *Recombination* operation, which GA/GP researchers generally refer to as *Crossover* (also, in some Biology and Genetics books, it' s referred as *Crossing-over*) because of the way that genetic material crosses over from one chromosome to another.

---

[6] T. A. Brown, *Genetics - A Molecular Approach*, p. 31.

[7] Thomas E. Hart, "Lamarck and His Theory of Evolution", Internet Document, http://www.stg.brown.edu/projects/hypertext/landow/victorian/science/lamarck1.html, 1997.

[8] *ibid.*

[9] J. Mark Baldwin, "A New Factor in Evolution", American Naturalist 30, pp. 441-451, 536-553, June 1896, *reprinted in:* http://paradigm.soci.brocku.ca/~lward/Baldwin/BALD_002.html, 1998.

The crossover operation that occurs in an environment where the selection of who or which gets to mate is a function of the *Fitness* of the individual, ie. how good the individual is competing or how hard it tries to survive and reproduce in its environment.

4.2.2 Terminology and The Basic Model

GA is a process which stimulates the way biological evolution works. Like evolution, it operates on a population of individuals which represent potential solutions to a given problem. It then seeks to produce better (more fit) individuals (solutions) by combining the better of the existing ones (breeding). Using a *survival of the fittest tactic*, it discards the bad ones and tends to produce more of the good individuals. Not only does it produce more of the good solutions but also provides better and better solutions. This is due to the fact that it combines the best traits of parent individuals to produce superior children or the offsprings, in other words. This combination operator is called *crossover*. The term *genetic algorithm* comes from the fact that individuals are represented as strings of bits, integers, letters, floating numbers, etc. analogous to chromosomes and genes. In addition to recombination by crossover, these bit-strings are also thrown in a randomly process referred as *mutation*. This prevents the GA from getting stuck at good but non-optimal solutions. Some of the most important keywords and issues in GA can be summarized as follows;

Crossover - The genetic process by which genetic material is exchanged between individuals in the population, thus, in a computer program implementation, the data that represents the candidate solutions for a problem (ie. variables in a function optimization) is exchanged. There are various types of crossovers in GA which will be explained later on.

Reproduction - The genetic operation which causes an exact copy of the genetic representation of an individual to be made in the population.

Mutation - Another genetic operation which a single candidate chromosome (a bit sequence) is selected and some of its bit values is randomly changed.

Generation - An iteration of the measurement of fitness and the creation of a new population by means of genetic operations.

Selection - The process that determines which parents shall undergo crossover or reproduction operations. There are various selection methods used in GA, such as Fitness-proportionate selection; which a simple function of the fitness measure is used to select individuals via a probabilistic scheme that shall undergo genetic operations, or Tournament selection; where certain randomly selected individuals in a subgroup compete and the fittest is selected.

Inversion - A technique first proposed by *J. H. Holland*, which works by reversing the order of genes between two randomly chosen positions within the chromosome.[10]

Complement - Also, complement (ie. considering a binary representation, taking complement of each bit, each gene in other words, thus, taking inverse of the chromosome) of a whole chromosome can be added to the population to maintain diversity.[11]

Reordering - It' s another genetic operation that changes and reorders the sequence of bits, or genes on a chromosome. This greatly expands the search space, since, not only is the GA trying to find good sets of gene values, it is simultaneously trying to discover good gene orderings too.[12]

Some genetic algorithms use a simple function of the fitness measure to select individuals in a probabilistic manner that consequently undergo genetic operations such as crossover or reproduction (the propagation of genetic material unchanged). This is *Fitness-Proportionate* selection. There are several different sub-methodologies used in fitness-proportionate selection, such as, *Roulette-Wheel* or *Fitness-Ranking*. Other implementations use a model in which certain randomly selected individuals in a subgroup compete and the fittest is selected. This is called *Tournament Selection* and is the type of selection that is seen in nature. There are also other selection methodologies used, such as *Stochastic Universal Selection* developed by *Baker*[13]. The two processes that most contribute to evolution are crossover and fitness based selection / reproduction. As it turns out, there are mathematical proofs and computer tests[14]-[15] which indicate that the process of fitness proportionate reproduction is, in fact, near optimal in some manner. Also, in some complex cases, Stochastic Universal Selection outperforms the others significantly as well as being elegantly simple and theoretically perfect.[16]-[17]

Mutation also plays a role both in nature and in GA, though it is not the dominant role that is popularly believed to be the process of evolution, ie. random mutation and survival of the fittest. It cannot be stressed too strongly that the genetic algorithm (as a simulation of a genetic process) is not a random search for a solution (highly fit individual) to a problem. The genetic algorithm uses stochastic processes, but the result is distinctly

---

[10] David Beasley, David R. Bull, Ralph R. Martin, "An Overview of Genetic Algorithms - Part 2", University Computing, UCISA, pp. 3-4, 1993.
[11] L. Darrell Whitley (editor), *Foundations of Genetic Algorithms 2*, pp. 146-150.
[12] David Beasley, David R. Bull, Ralph R. Martin, "An Overview of Genetic Algorithms - Part 2", University Computing, UCISA, pp. 3-4, 1993.
[13] David Beasley, David R. Bull, Ralph R. Martin, "An Overview of Genetic Algorithms - Part 1", University Computing, UCISA, p. 11, 1993.
[14] L. Darrell Whitley (editor), *Foundations of Genetic Algorithms 2*, pp. 128-139.
[15] Zbigniew Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, pp. 58-60, 132-135.
[16] L. Darrell Whitley (editor), *Foundations of Genetic Algorithms 2*, pp. 128-139.
[17] David Beasley, David R. Bull, Ralph R. Martin, "An Overview of Genetic Algorithms - Part 1", University Computing, UCISA, pp.11-12, 1993.

non-random which is proven to be better than random. Also, there has always been an argument between GA researchers whether mutation or crossover, or both, play an important role in the performance of GA, and which is more or perhaps, the only necessary one.[18]-[19]-[20]-[21] There' s no exact and clear answer, since the performance of a GA depends on the nature of the problem, and thus, the usage of crossover and mutation and their effects vary among different problem-solution domains.

A sample genetic algorithm can be formulated as follows[22] ;

Let *P(t)* define a population of candidate solutions at time *t*:

$P(t) = \{x_1^t , x_2^t , ... x_n^t\}$

Then, a procedure for GA can be generated as:

```
procedure genetic algorithm;
 begin
 t := 0;
 initialize P(t);
 while termination condition not met do
        begin
          evaluate P(t);
          select pairs of solutions according to the quality of their evaluation;
          produce the offspring of these pairs using genetic operators;
          replace the weakest candidates with the offspring;
          t := t +1;
        end;
end;
```

Here, it should be noted that evaluation of candidate solutions assumes a fitness function, $f(x_i^t)$, that returns a measure of candidate' s fitness at time *t*. Using such a fitness function, a typical evaluation assigns each candidate solution a value as follows[23];

$$f(x_i^t) / m(P,t)$$

where $m(P,t)$ is the average fitness over all members of the population.

---

[18] L. Darrell Whitley (editor), *Foundations of Genetic Algorithms 2*, pp. 200-201.
[19] *ibid*, pp. 221-237.
[20] *ibid*, pp. 239-254.
[21] David Beasley, David R. Bull, Ralph R. Martin, "An Overview of Genetic Algorithms - Part 2", University Computing, UCISA, p. 6, 1993.
[22] George F. Luger, William A. Stubblefield, *Artificial Intelligence Structures and Strategies for Complex Problem Solving - Second Edition*, p. 528.
[23] *ibid*, p. 528.

In general, it' s stated that any implementation of a genetic algorithm or an evolutionary algorithm requires five basic components, which are grouped as below[24]-[25] ;

- a genetic representation for potential solutions to the problem
- a method for creating an initial population of potential solutions
- an evaluation function verifying fitness of each solution and rating these solutions
- genetic operators changing a gene' s contents in chromosomes (the composition of children) during reproduction
- some constant values for different parameters used by the algorithm (such as population size, probability of applying crossover or mutation, etc.)

It should be noted that except for some very specific and unique applications, the initial populations used in such algorithms are selected or created randomly. However, at the end of reproduction, the new population after the recombination and fitness ranking might be either derived randomly choosing among the new children, or via a selection mechanism among only new children or new children plus the old population.[26]-[27]

Another important remark is that the execution of a genetic algorithm is in fact based on two main stages. The first stage starts with the current population where the selection mechanisms are applied to this population in order to create the children (the intermediate generation) as candidates for the next generation. The second stage is the phase where recombination operators such as crossover and mutation are applied to the intermediate population and after this process, the new population is established.[28]

4.3 How and Why Genetic Algorithms work?

Before going into further details, several terms should be mentioned first which are used in GA terminology. In fact, genetic algorithms use a vocabulary borrowed from natural genetics. *Gene* (also referred to as a feature, character or detector) refers to a specific attribute that is encoded in the genotype' s chromosome (string). The particular values the gene can take are called its *Alleles*. The position of the gene in the chromosome is its *Locus*.[29] A common example taken from biology is that eye color is determined by a gene, the different eye colors are the gene' s alleles (blue, brown, etc.) and where the eye color gene happens to lie in the chromosome is its locus. This terminology is used among most of the GA researchers and the reason is that these researchers are coming from natural genetics domain rather than from computer science, whereas computer scientists

---

[24] Zbigniew Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, pp. 17-18.
[25] Leonard Bole, Jerzy Cytowski, *Search Methods for Artificial Intelligence*, p. 202.
[26] Enrique Alba, Carlos Cotta, "Introduction to Nature-Inspired Algorithmic Techniques", Internet Document, http://www.lcc.uma.es/personal/cotta/semEC/cap01/cap_1.html, 1997.
[27] David Beasley, David R. Bull, Ralph R. Martin, "An Overview of Genetic Algorithms - Part 1", University Computing, UCISA, pp. 6-7, 1993.
[28] L. Darrel Whitley, "A Genetic Algorithm Tutorial", Computer Science Department, Colorado State University, Technical Report, p. 4, November 10, 1993.
[29] Zbigniew Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, p. 15.

and software engineers, prefer the more computer-based terminology such as position, size, etc.

The characteristics an algorithm must have, in order to be considered as genetic will be discussed in the following sections. These characteristics will be described explaining the facts; how they are usually implemented and why they work in the desired way.

4.3.1 Fitness Function

Due to the nature of a genetic algorithm, it is easier to describe them as solving optimization problems which search for maxima. However, this does not prevent the researchers from using GA' s to find minimal values. A minimization problem can always be mapped mathematically into a maximization problem. To make it clear, an essential component of all GA' s - the fitness function (also called the objective function) will be explained.

The fitness or objective function is used to map the individual' s bit strings into a positive number which is called the individual' s fitness. There are two steps involved in this mapping; however it' s known that in some problems these two steps are essentially accomplished as one. The first step is referred as *decoding* and the second, *calculating fitness*. In order to understand the decoding concept, it can be simply stated that it helps to distinguish individuals into two parts commonly called the genotype or genome and the phenotype. These terms come from biology. The genotype, as its name implies, specifically refers to an individual' s genetic structure or for the software implementations, the individual' s bit string(s). The phenotype refers to the observable appearance of an individual. For our purpose, the phenotype is the desired result; the parameters of the problem that yield its fitness. Therefore, decoding refers to the process of mapping the genotype to the phenotype, the code to the parameters. For example, in the function optimization problem $f(x,y) = x^3 - xy^2$, a 60-bit string might be used to encode the parameters x and y (30 bits per parameter). Thus, the decoding process would refer to the mapping of these bit strings to real values (x,y) on some range of interest [M,N] (for example [-1.0, 2.0]). Using the genetic structure, x and y are the genes which occur at locus, or position, namely, *0* and *30* in the chromosome. These can be analyzed from the Figure 4.1. It should be stressed that, in this genetic implementation, the gene' s alleles range from -1 to 2.0 in increments of $(2 - (-1)) \times 2^{-30} = 3 \times 2^{-30}$.

The second step, fitness calculation, is trivial once the genotype has been decoded. A function is used to map the phenotype's parameter values into a positive number, which is referred as the fitness. In a function maximization problem, the fitness function is often shown to be the same as the function being optimized (provided that the output domain of the function is positive). If the domain contains negative values or whenever a minimization problem is the case, then the output of the function can be simply be mapped into the required form. It' s indicated that for negative values, this will involve

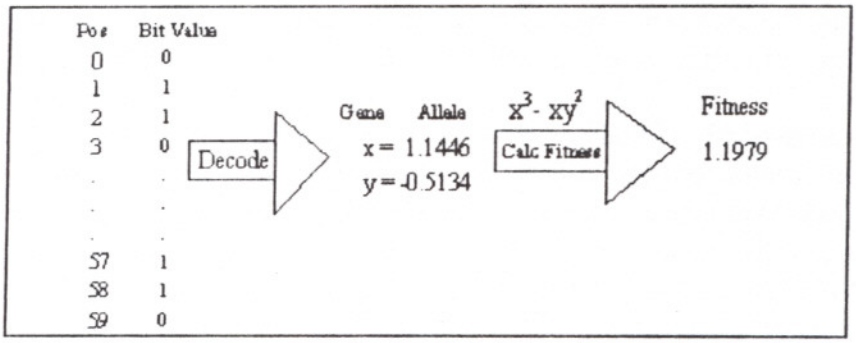


Figure 4.1 Example of Fitness Calculation for Parameter Optimization $f(x,y) = x^3 - xy^2$.

adding a constant to the output so as to shift it into the positive range. For minimization problems, the following deduction is used[30] :

$$Min\ (f(x)) = Max(g(x)) = Max(-f(x)).$$

This states that the minimum of a function, $f(x)$, is equivalent to the maximum of some other function $g(x)$ where one specific $g(x)$ for which $-1*f(x)$ is true. Again if, $-f(x)$ is negative (it might not be if the domain of $f(x)$ is negative) then an appropriate constant, C might be added as follows[31] ;

$$Min\ (f(x)) = Max(g(x) + C) = Max(-f(x) + C).$$

It must be stressed that the construction of a fitness function is shown to be very important and crucial for the correct execution of a GA with the satisfying results. Thus, the design and implementation of a suitable and proper fitness function for the problem is heavily dependent on the following issues which usually makes it very difficult for the designer[32] ;

- The fitness function differs with respect to the maximization or minimization criterion.
- The environment is susceptible to noise in the evaluations, ie. partial evaluations.
- The fitness function might change dynamically as the GA proceeds.
- The fitness function might be so complex that only approximations to fitness values can be computed.
- The fitness function should allocate very different values to strings in order to facilitate the process of selection mechanisms.
- The fitness function must be designed so as to consider the constraint of the problem properly by clearly distinguishing the unfeasible solutions.

[30] Zbigniew Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, p. 31.
[31] *ibid*, p. 31.
[32] Enrique Alba, Carlos Cotta, "Introduction to Nature-Inspired Algorithmic Techniques", Internet Document, http://www.lcc.uma.es/personal/cotta/semEC/cap02/cap_2.html, 1997.

- In some cases, the fitness function might form some different sub-objectives. It' s stressed that such a multi-objective function can cause non-conventional problems.
- The fitness function is like a black box for the GA. Phenotype is fed into this black box and the fitness value is derived as the output, but the internal mechanisms that enable this are not always straightforward. For instance, such mechanisms need to be achieved by complex mathematical functions, a complex simulator program or indeed by a human who himself makes the decision for good strings, ie. when GA is used in music or animation programs.

The operations of decoding and fitness calculation can also be applied to problems even when a function optimization is not the case. For instance, in the Traveling Salesman Problem, (a well known NP-Hard Problem), the genotype would be chosen in the way a tour is encoded in a string. The *parameters* or phenotype would be the actual tour being considered and the fitness would be a function of the length of the tour for a minimization problem.[33] The Traveling Salesman Problem and its GA implementation will be described in more detail in the subsequent sections.

4.3.2 Selection

Selection is one of the most important mechanisms in GA' s. Selection determines which individuals in the population will have all or some of its so-called *genetic material* passed on to the next generation of individuals. The object of the selection method employed in a GA is to give exponentially increasing trials to the fittest individuals. The most common scheme to accomplish this is the technique named as *roulette-wheel selection*, but it shouldn' t be considered as the best. This technique involves selecting the positive fitness values where higher values indicate greater fitness. Roulette wheel selection gets its name from the fact that the algorithm works like a roulette wheel in which each slot on the wheel is paired with an individual in the population. This is done in such a way that the size of each slot is proportional to the corresponding individual' s fitness. Due to this fact, this selection method is also referred as *fitness-proportionate selection*. It should be obvious that maximization problems fit directly into this paradigm where larger slot implies larger fitness. On the other hand, it' s remarked that negative values are not allowed because there' s no such a slot of negative size.[34]

A common way to implement roulette wheel selection is as follows:

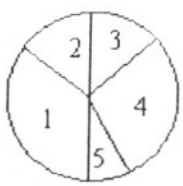
1. Sum up all the fitness values in the current population, call this value *SumFitness*. SumFitness is in effect the total area of the roulette wheel.
2. Generate a random number between 0 and 1, called *Rand*.
3. Multiply SumFitness by Rand to get a number between 0 and SumFitness which will be called *RouletteValue* (*RouletteValue* = *SumFitnesss x Rand*). Think of this value as the distance the imaginary roulette ball travels before falling into a slot.

[33] Zbigniew Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, pp. 25-26.
[34] *ibid*, pp. 32-33.

4. Finally sum up the fitness values (slot sizes) of the individuals in the population until an individual, which makes this partial sum greater or equal to RouletteValue, is reached. This will then be the individual that is selected.

It is not always intuitively obvious that this algorithm actually implements a weighted roulette wheel. To see this fact, some extreme situations should be considered. For instance, there can be an individual, whose fitness is equal to SumFitness (implying all other individuals have a fitness of zero).

It' s proven that no matter what number is generated for RouletteValue, someone will always throw the partial sum over the top, thus having a selection probability of 1. This corresponds to a roulette wheel with just one slot. On the other extreme, an individual, i, with fitness zero can never cause the partial sum to become greater than RouletteValue, so it has a zero probability of getting selected. It' s shown that this corresponds to a slot that does not exist on the wheel. All other individuals between these extremes will have a probability of throwing the partial sum over the top that is proportional to their size, which is exactly the expected behaviour of a weighted roulette wheel. An example of the roulette wheel selection is given in Figure 4.2 where for five individuals, their corresponding fitness values and the probability for each within the provided slot size percentages are denoted.



Roulette Wheel

Individual	Fitness	Slot Size %
1	30	35
2	10	12
3	15	18
4	25	29
5	5	6
SumFitness:	85	

Figure 4.2 Roulette Wheel Selection with five Individuals of varying Fitness.

This selection mechanism can also be denoted as follows;

f_i : the fitness value of each individual for i=1,2,...n
p_i : the probability of being selected for each individual

$$p_i = \frac{f_i}{\sum_{i=1}^{n} f_i}$$

Fitness-proportionate selection is not the best way to implement selection if both efficiency and fairness are taken into consideration. This method is not always fair, because it' s proven that random noises (referred to as stochastic errors) might cause the actual number of individuals of varying fitness to receive more or less than their expected preference. To overcome this drawback, *Stochastic Universal Selection (SUS)* has been

developed. It' s proven that in this method, sometimes also referred as stochastic universal sampling, the mean variation can almost be completely eliminated by using the reproduction rate in a more generic way than the fitness-proportionate selection, and by applying the special sampling mechanism. In contrast, the wheel used in *SUS* is divided into a number of equally spaced markers equal to the population size.[35]-[36]

There are also other selection and sampling mechanisms such as; *Tournament Selection, Truncation Selection, Linear Ranking Selection, Exponential Ranking Selection,* etc. which are also classified according to some specific properties they possess or some affects they pose to the population such that; *dynamic or static selection, extinctive or preservative selection, ellitist or pure selection, generational or steady-state selection.*[37]-[38]-[39]-[40]

The efficiency and performance of these selection schemes are proven to vary among the different GA models and implementations. It' s stressed that when one is to decide upon a selection scheme, three basic criteria should be concerned. These are *selection intensity, selection variance* and *the loss of diversity.* The selection intensity can be defined as the measure which gives information about the selection mechanism' s effect on the average fitness of the population. This effect can be obtained by computing the difference between the population average fitness after and before selection. The selection variance is the normalized expected variance of the fitness distribution of the population after applying the selection method to the fitness distribution. The loss of diversity is the proportion of individuals of a population that is not selected during the selection phase.[41]

Under these circumstances, it' s generally accepted that despite being the simplest, fitness-proportionate selection is unsuitable in most GA applications producing side-effects among the population, and degrading the search efficiency. It' s also stated that *SUS* is very powerful and strongly decisive but it has high computational complexity which might slow down the execution of the GA. On the other hand, when the highest selection variance is regarded as the main criteria, exponential ranking selection is proven to be the best method.[42]-[43]

It' s stressed that the common and crucial goal in all legitimate GA selection techniques is to reward better fit individuals by letting them reproduce more often. This is

---

[35] Zbigniew Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs,* pp. 56-57.
[36] Tobias Blickle, Lothar Thiele, "A Comparison of Selection Schemes used in Genetic Algorithms", Swiss Federal Institute of Technology, TIK-Report, No:11, pp. 43-45, December 1995.
[37] *ibid,* pp. 14-51.
[38] Zbigniew Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs,* pp. 56-58.
[39] Richard K. Belew, Lashon B. Booker (editors), *Proceedings of the 4th International Conference on Genetic Algorithms and their Applications,* pp. 92-99.
[40] L. Darrell Whitley (editor), *Foundations of Genetic Algorithms 2,* pp. 132-133.
[41] Tobias Blickle, Lothar Thiele, "A Comparison of Selection Schemes used in Genetic Algorithms", Swiss Federal Institute of Technology, TIK-Report, No:11, pp. 11-13, December 1995.
[42] *ibid,* pp. 49-52.
[43] L. Darrell Whitley (editor), *Foundations of Genetic Algorithms 2,* p. 134.

one of the important aspects of a GA which makes it different from random search or several *hill-climbing methods*. It should be stressed that when compared to other search and optimization algorithms, such as *random search, gradient methods* (ie., *random-mutation hill-climbing*), *iterated search* (ie., *steepest-ascent hill-climbing, next-ascent hill-climbing,*), *simulated annealing*, etc., GA' s outperform all the others in some cases or have better performance than some of the other algorithms in some other cases.[44]-[45]-[46]

4.3.3 Reproduction

The second important issue in all genetic algorithms is stated to be the fact that they contain some sort of recombination procedure. It' s the reproduction stage where two individuals selected in the previous step are allowed to mate to produce offspring. This mating is done by a genetic operator commonly called as *crossover*. As mentioned previously, crossover is the process by which the bit-strings of two parent individuals combine to produce two child individuals, or offsprings, namely.[47]-[48]

There are many ways in which a crossover can be implemented. Some of the ways are broadly applicable to all types of problems and others are highly problem-specific. The most primitive but also highly effective form of crossover that is commonly implemented is single-point crossover. Single point crossover starts by selecting a random position on the bit string, called a cut point or cross point. The bits from the left of the cut point on parent1 (P1) are combined with the bits from the right of the cut point in parent2 (P2) to form child1 (C1). Similarly, the opposite parts are combined to form child2 (C2). A simple example of single-point crossover is given in the Figure 4.3.

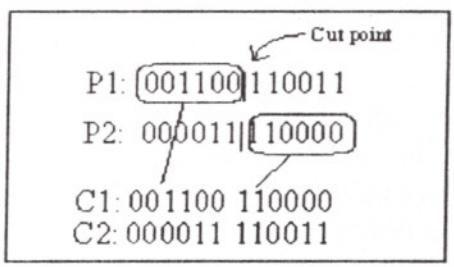


Figure 4.3 Example of a single-point crossover.

---

[44] David Beasley, David R. Bull, Ralph R. Martin, "An Overview of Genetic Algorithms - Part 1", University Computing, UCISA, pp. 4-6, 1993.

[45] Thomas Back, David B. Fogel, Zbigniew Michalewicz (editors), *Handbook of Evolutionary Computation*, pp. 1-5.

[46] Zbigniew Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, pp. 16-17, 26-29.

[47] David Beasley, David R. Bull, Ralph R. Martin, "An Overview of Genetic Algorithms - Part 1", University Computing, UCISA, p. 3, 1993.

[48] L. Darrel Whitley, "A Genetic Algorithm Tutorial", Computer Science Department, Colorado State University, Technical Report, p. 6, November 10, 1993.

Thus, child1 and child2 will tend to be different from either of their parents, but also shall possess some characteristics of both. If the parents each had high fitness then there would be a good chance that at least one of the children is as fit or better than either parent. If this is the case, then selection will favour this child's s existence and insertion into new population, if not, then selection will favour the child's extinction. There is, of course a small possibility that most or all of the crossovers produce children of less fitness. To overcome this possibility, a parameter, P_C, the probability of crossover, is introduced. Before crossover is applied, a random number is generated. If this number is less than or equal to P_C, then crossover is performed, if not, then the parents are passed into the next generation unchanged.[49] Since without crossover it' s proven that except some complex and special cases, no advancement can be observed in the execution, P_C is usually set high ($0.5 < P_C < 1.0$).

There are also other kinds of crossovers for bit-wise and real-coded applications, such as *Two-Point Crossover*, *Multi-Point Crossover*, *Uniform Crossover (UX)*, *Half-Uniform Crossover (HUX)*, *Parameterized Uniform Crossover*, *Parameter-bounded Blend Crossover (BLX-α)*, *Segmented Crossover*, *α Crossover*, *Randomized And/Or Crossover (RAOC)*, *Randomized Uniform Crossover (RUC)*, etc.[50-51-52-53-54-55] It' s proven that the performance, efficiency and applicability of such crossover schemes vary among the implementation, data structure and the algorithm chosen. Therefore, there can be no crossover model in general to be regarded as the best or the worst. One should decide upon the most suitable crossover scheme according to the structure of the problem and the corresponding GA he / she deals with.

As mentioned previously, another important and effective GA operator is shown to be mutation. Mutation can be simply defined as the operation on which the value of a single bit or some of the bits within an individual is changed in a random fashion so as to produce a new child with different genes. An example of a single-bit mutation is given in Figure 4.4 where the 3rd leftmost bit of P1 is mutated from 1 to 0 as it' s passed onto C1 within the same crossover denoted in Figure 4.3. It should be stressed that, in genetic algorithms, high mutation rates cause the algorithm to degenerate to random search and usually this is an undesired situation. The rate of genetic drift can be reduced by increasing the mutation rate. However, if the mutation rate is too high, the search becomes

---

[49] David Beasley, David R. Bull, Ralph R. Martin, "An Overview of Genetic Algorithms - Part 1", University Computing, UCISA, p. 3, 1993.

[50] Bill Keller, Rudi Lutz, "A New Crossover Operator for Rapid Function Optimisation Using a Genetic Algorithm", School of Cognitive and Computing Sciences, The University of Sussex, pp. 1-11.

[51] David Beasley, David R. Bull, Ralph R. Martin, "An Overview of Genetic Algorithms - Part 2", University Computing, UCISA, pp. 1-2, 1993.

[52] Zbigniew Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, pp. 68-70.

[53] L. Darrell Whitley (editor), *Foundations of Genetic Algorithms 2*, pp. 34-43.

[54] *ibid*, pp. 188-201.

[55] *ibid*, pp. 239-254.

effectively random, thus enabling to investigate new and unknown areas in the search space.[56]

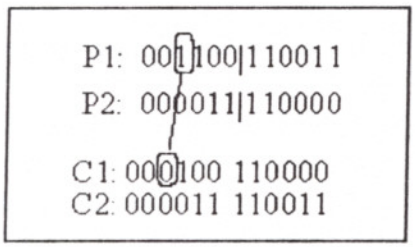


Figure 4.4 Example of a single bit mutation.

Unlike crossover, mutation is a unary operator, in other words, it only acts on one individual at a time. As bits are being copied from a parent individual to a child, usually a random number is generated. If, the probability of mutation, P_m namely, is greater than this random number, mutation is activated. P_m is usually chosen small ($0 <= P_m <= 0.1$).[57] It is remarked that there' s no optimum value for P_m, it depends on the algorithm and problem. P_m can be deduced heuristically or by some simple mechanisms. (ie. P_m = 1 / *Chromosome Length*.)

There' s also another genetic operator known as *inversion*. However, inversion is not used as often as crossover and mutation in most GA' s. Inversion is a process that shifts the locus of one or more gene in a chromosome from one point to another. This does not change the meaning of the genotype in the sense that a genotype before and after inversion will still decode to they same phenotype.[58] Consequently, this fact causes a question in mind wondering why designers would be bothered with inversion at all.[59] The theory behind inversion is that there are groups of two or more genes in a chromosome that work together to yield a high fitness. If these genes are physically close together than single point crossover is much less likely to disturb these groups. Although this argument seems reasonable, inversion in practice has achieved conflicting results. Another operator similar to inversion is known as *reordering*. Reordering changes the sequence of genes of any chromosome to any new valid sequences. It' s proven that reordering might be useful in implementations where the gene orderings of the individuals are effective as well as the gene values.[60] Especially, for the problems where the ordinal encoding is necessary such as graph partitioning problems or TSP. Besides establishing a proper order, reordering can help to expand the search space for the candidate solutions. In many GA applications inversion is ignored whereas in some applications for which the fitness function might

---

[56] David Beasley, David R. Bull, Ralph R. Martin, "An Overview of Genetic Algorithms - Part 1", University Computing, UCISA, p. 8, 1993.
[57] L. Darrel Whitley, "A Genetic Algorithm Tutorial", Computer Science Department, Colorado State University, Technical Report, p. 6, November 10, 1993.
[58] Zbigniew Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, pp. 52-53.
[59] David Beasley, David R. Bull, Ralph R. Martin, "An Overview of Genetic Algorithms - Part 2", University Computing, UCISA, pp. 3-4, 1993.
[60] *ibid*, pp. 3-4.

change dynamically over time, special GA operators such as inversion and reordering can be considered useful.

4.3.4 Alternative GA Operators and Schemes

So far, the only format applied to genotypes is bit string encoding. However, there is an increasingly large number of GA' s being researched and implemented which do not use bit encoding at all. Some researchers who are purists or conservative in terminology would claim that these are not true genetic algorithms. The others oppose to this and claim that non-binary representations are better suited to complex optimization problems. The term evolutionary program is sometimes applied to algorithms which are in the soul of GA' s but use other kinds of data structures.[61]

Two common forms of genetic encoding alternatives are real or floating numbers and ordinals. Real encoding uses strings (arrays) of real numbers rather than bits. Thus, instead of encoding each parameter in a genetic algorithm as many bits, a single real number can be used. With a real numbered scheme a decoding step is not necessary since the *genes* are already in the form they will be processed.

It' s stressed that when using a real valued encoding with several genes, traditional single point crossover can certainly be used. However, one of the reasons of the trend to real encoding is to exploit the ability to manipulate the chromosomes directly as strings of numbers. This suggests that new forms of crossover might be more useful and efficient. One possible technique can be crossing by averaging the two parents. Another technique might use *weighted averages*, weighting each child toward a different parent.[62] Still others may combine features from single point crossover with weighted crossover. Imagination and experimentation can often lead to varying results based on the nature of the problem.

Mutation in real encoding also presents the designer with a variety of choices. The simplest method might be to replace a real gene completely with another randomly generated one. One can also average a random number into the existing gene. Another good alternative is to add or subtract a small noise factor into the gene's value. Or, swapping two numbers in the chromosome string, known as *structural mutation*. Again the choice and results usually depend on the problem in hand.[63]-[64]

An example of a domain where real encoding is useful is in engineering design optimization. Here, there are usually a large number of real parameters which describe the various design choices that can be made. Finding a near optimal design for a given application is often an arduous task. Because of the larger parameter space, bit strings would probably yield poor results. In addition, real encoding would be more intuitive to

---

[61] Zbigniew Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, pp. 3-6, 75-82.
[62] David Beasley, David R. Bull, Ralph R. Martin, "An Overview of Genetic Algorithms - Part 2", University Computing, UCISA, pp. 2-3, 1993.
[63] Zbigniew Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, pp. 87-88.
[64] *ibid*, pp. 213-214.

the engineer and allow him to incorporate other known algorithms into the optimization process. This is known as *hybridizing* the genetic algorithm.[65]

Another class of optimization problem that requires a different encoding scheme is combinatorial optimization. In combinatorial optimization problems, people are concerned with optimizing the order of some operation rather than a set of control parameters. An example of this is the *Traveling Salesman Problem (TSP)* which is previously mentioned. In *TSP*, we have a salesperson who must make a tour throughout the country visiting various cities and then returning to home base. The salesperson can not go to any city more than once. The goal is to schedule a travel plan (a tour) which minimizes the total distance travelled. It is not obvious that this algorithm can be mapped onto a bit-string encoding, although it has been done. A more natural and hence, efficient encoding is to use a string of ordinals. These are simply positive integers which describe the order in which the cites should be visited (assuming each city is associated with a number). Thus, the solution of the problem might be achieved via a hybridized genetic algorithm, in an evolutionary sense rather than a standard genetic algorithm model. There are actually several methods of solving the *TSP* via the use of a GA.[66]-[67]-[68] The one illustrated in Figure 4.5 uses a representation generally known as path encoding.

For the example denoted in Figure 4.5, there are ten cities in the tour, and the tour order given in that figure is one of the possible schedules, which is [1,4,3,2,5,6,8,9,10,7]. In fact, this is a vector-based representation where each city travelled at a unit time is each element of this vector, thus an individual stands for the complete touring path. For instance, for the vector mentioned here, at time t, the salesman is at city 1, then at time $t+1$, he goes to city 4, and so on. Hence, for the GA implementation, the population is composed of several possible and legal (no cycles, etc.) individuals that represent a complete tour with all the cities travelled, path encoding, in other words.

The evaluation of these individuals, ordinal number encoded chromosomes, is done by calculating the overall cost of the complete tour. Since the cost of travel between each city pair is given initially in the problem, the calculation process is trivial.

The aim of the GA is to search and find the best ordering of the cities in the tour, thus to find the chromosome that gives the minimum cost. Thus, by establishing a proper fitness function which calculates each vector' s cost and gives a fitness value to each chromosome that suggests the fittest individuals with the minimum costs, the problem can be solved via a GA approach. The important fact is that since the encoding and the structure of the genes and the chromosomes is of a special case in *TSP*, the recombination and reproduction operators, crossover, mutation, etc. should also be designed and adapted in a suitable manner regarding the ordinal encoding and the order of genes.

---

[65] Zbigniew Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, pp. 7-9, 75-78.
[66] *ibid*, pp. 165-191.
[67] Lawrence Davis (editor), *Genetic Algorithms and Simulated Annealing*, pp. 43-60.
[68] Leonard Bole, Jerzy Cytowski, *Search Methods for Artificial Intelligence*, pp. 211-217.

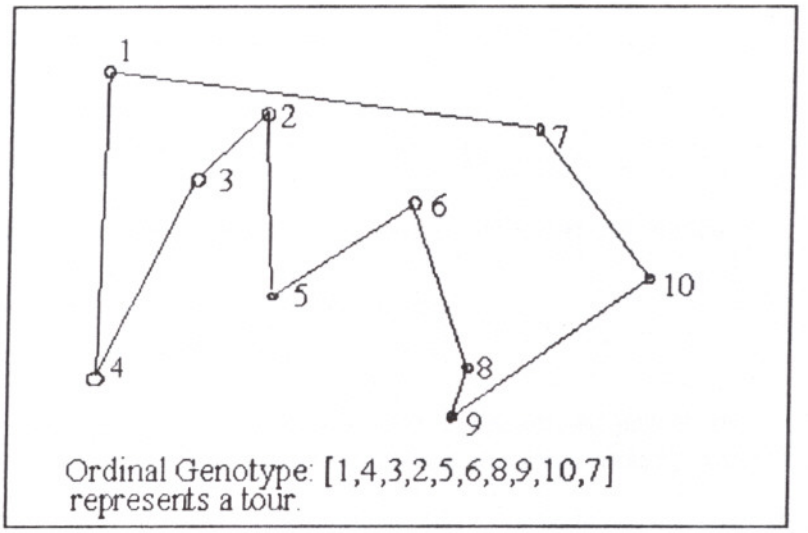


Figure 4.5 Traveling Salesman Example with an Ordinal Encoded Individual.

This is due to the fact that the types of operators that have been considered for bit strings and real numbers are shown to be problematic when applied to ordinals. The reason is that operators like single-point crossover or simple mutation are likely to cause the generation of invalid individuals. It' s strictly remarked that an ordinal chromosome must not contain duplicate ordinals and must have every ordinal in its range represented, in other words, the order of genes within a chromosome and the genotype do have essential and critical importance in implementations such as *TSP*.

For instance, *partially-mapped crossover (PMX)*, *order-crossover (OX)*, *cycle-crossover (CX)* crossover methods and mutation operators such as inversion, insertion, displacement, reciprocal exchange can be used.[69]-[70] For example, the order-crossover can be processed as follows[71] ;

Let two parents representing ordered path for the complete tour be:

p_1 = [1 2 3 4 5 6 7 8 9 10]
p_2 = [4 5 2 1 8 7 6 9 3 10]

The order-crossover can be applied to these parents by choosing a subsequence of a tour from one parent and preserving the relative order of cities from the other parent. If, the subsequence is chosen as the middle four cities in the tour, then the corresponding offsprings will be produced in the following way: First the segments between cut points for the four cities in the middle are copied into offsprings. Thus;

[69] Zbigniew Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, pp. 172-176.
[70] *ibid*, p. 213.
[71] *ibid*, pp. 173-174.

the parents with the cut points denoted:

$$p_1 = [1\ 2\ 3 \mid 4\ 5\ 6\ 7 \mid 8\ 9\ 10]$$
$$p_2 = [4\ 5\ 2 \mid 1\ 8\ 7\ 6 \mid 9\ 3\ 10]$$

the corresponding offsprings with the tour between the cut points:

$$o_1 = [x\ x\ x \mid 4\ 5\ 6\ 7 \mid x\ x\ x]$$
$$o_2 = [x\ x\ x \mid 1\ 8\ 7\ 6 \mid x\ x\ x]$$

Next, beginning from the second cut point of the second parent, the cities from the first parent are copied in the same order, omitting the ones already present in the first offspring. Thus,

the sequence of p_2 : 9 3 10 4 5 2 1 8 7 6
removing the ones present in o_1 : 9 3 10 2 1 8

This new sequence is then placed in the first offspring, starting from the second cut point, so that the new offspring is formed as;

$$o_1 = [2\ 1\ 8\ 4\ 5\ 6\ 7\ 9\ 3\ 10]$$

The similar procedure can be applied for the second offspring, thus;

$$o_2 = [3\ 4\ 5\ 1\ 8\ 7\ 6\ 9\ 10\ 2]$$

The other crossover operators use methodologies basically similar to the order-crossover. The common goal in all these operators is to derive the offsprings with no cycles or anomalies in the tours they represent. However, other types of crossover operators are used for the GA implementation of TSP, where rather than focusing the positions and the sequence of the cities within a tour, the links between the cities are considered; hence, the encoding of the chromosomes and the crossover operators are implemented via matrix representations. *Edge combination crossover (ERX)*, *matrix crossover* are such of these models.[72]

As well as crossover, some various mutation schemes can be applied to the TSP implementation. However, these mutation operators must be designed so as to be well-suited for the ordinal encoding of the chromosome. Some of these mutation operators can be summarized as: *insertion*, selecting a city and inserting in a random place; *displacement*, selecting a subtour and inserting in a random place; *reciprocal exchange*, swapping two cities.[73]-[74] For instance, the displacement mutation can be applied as follows;

---

[72] Zbigniew Michalewicz, *Genetic Algorithms + Data Structures – Evolution Programs*, pp. 176-189.
[73] *ibid*, p. 176.
[74] Leonard Bole, Jerzy Cytowski, *Search Methods for Artificial Intelligence*, p. 215.

individual:	[3 1 5 2 6 8 4 7]
selected subtour:	{5 2 6}
random position:	before city 7
individual after mutation:	[3 1 8 4 5 2 6 7]

Any of the selection methods mentioned previously can be used for the GA implementation of the *TSP*. Once the fitness values established properly for each chromosome, any selection mechanism can be applied involving no extra adaptation for the ordinal encoding.

4.4 Theory of Genetic Algorithms

In the following sections, a short analysis of the theoretical and mathematical basis that genetic algorithms rely on, will be given with necessary examples and notations.

4.4.1 The Fundamental Theorem of Genetic Algorithms

In section 4.3, it' s been stated that GA's work because a GA gives exponentially increasing trials to the fittest individuals. However, it' s shown that this is not completely accurate. For instance, if a GA only worked at the level of the individual, then it would be doubted whether it could be sufficient to explore spaces as large as $2^{100} \approx 10^{30}$ or greater. Certainly, it' s stated that in order to be efficient, a GA must limit its population size to some manageable number. This number is taken usually in the range of 10^2 to 10^5. Then in contrast, anyone might suspect how could a few hundred or even a few thousand individuals could explore a space with a trillions of different points. The answer is proven to lie in a more accurate definition of the underlying GA theory. Instead of saying that individuals receive exponentially increasing trials, the fundamental theorem of genetic algorithms states that each time an individual is processed, the GA is not simply sampling a single point in the space but too many points. In order to understand how this can be achieved, the concept of schemata should be known, which will be discussed in the following section.

4.4.2 Schema Theorem and The Building Block Hypothesis

It' s stressed that whenever bit string genotypes are chosen for the implementation, by no doubt, each position in the string is limited to the characters *0* and *1* only. In other words, the representational alphabet domain for the strings is {0,1}. Throughout this discussion an additional third symbol will be used which is *, and will stand for the *don't care* property. This produces the new alphabet as {0,1,*}.[75-76] A string from this alphabet

[75] Melanie Mitchell, Stephanie Forrest, John H. Holland, "The Royal Road for Genetic Algorithms: Fitness Landscapes and GA Performance", Proceedings of the First European Conference on Artificial Life, Cambridge, pp. 1-2, 1991.

[76] Thang Nguyen Bui, Byung Ro Moon, "Genetic Algorithm and Graph Partitioning", IEEE Trans. On Computers, vol. 45, No. 7, pp. 843, 847-848, July 1996.

might look like the following: 00**11. The meaning of the don't care symbol, is that whatever value (0 or 1) is preserved at that position, is not taken into consideration, it can be neglected. Another way of stating this is that the sample string, 00**11 matches all of the strings in the following set {000011, 000111, 001011, 001111}. Strings which include the don't care symbol are called *schemata* or *similarity templates*. In order to understand the importance of schemata, the following hypothetical population of bit-strings and their associated fitness is given as an example;

String	Fitness
10011	361
00110	36
11001	576
01110	196
00111	158

Some patterns between the strings and their fitness can be directly notified from this example, without even knowing how the strings decode to yield their fitness values. One pattern that seems to emerge is that strings which begin with 1 do have significantly higher fitness values than those which begin with zero. Another deduction that holds for this example is that strings with more 1' s than 0' s tend to have higher fitness. Thus, by only looking at 5 strings out of a possible 32, a lot of information can be extracted. If one had to make a guess for a string that would give even better fitness, he' d confidently guess 11111, given these two observations. The first observation noted is equivalent to stating that strings which match the schema 1**** have higher fitness than those that match the schema 0****. The second observation is that; strings which match schemata from the set {1111*, 111*1, 11*11, 1*111, *1111, 111**, 11**1, 1**11, **111} will have greater average fitness than those which match schemata in the set {0000*, 000*0, 00*00, 0*000, *0000, 000**, 00**0, 0**00,**000}. Also, it should be noted that a chromosome of length n can be viewed as an instance of 2^n schemas. For instance, a chromosome or string 101 is an instance of $2^3 = 8$ schemas: ***, 1**, *0*, **1, 10*, 1*1, *01 and 101.[77]

Therefore, the success of GA' s, can be concluded to be due to the fact that they implicitly search for patterns (schemata) with higher fitness. In a bit string of length *5* there are only $2^5 = 32$ individuals but $3^5 = 243$ schemata. Thus, it can be clearly stated every time a GA processes a single individual it is actually processing many schemata. According to the schema theorem first worked out by *John Holland*, in a population of size N, the number of schemata effectively processed is in the order of N^3. This assertion is based on a specific property known as *implicit parallelism* or sometimes referred as *intrinsic parallelism*. In fact, the implicit parallelism property is remarked to be one of the basic reasons that explain the good performance and power of GA' s.[78]-[79]-[80]

---

[77] Thang Nguyen Bui, Byung Ro Moon, "Genetic Algorithm and Graph Partitioning", IEEE Trans. On Computers, vol. 45, No. 7, pp. 843, 847-848, July 1996.

[78] *ibid*, p. 843.

[79] Zbigniew Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, p. 52.

This fact can be simply denoted by an example. In any GA application, if a population of *1000* individuals each of length *100* exists, then it is necessarily sufficient to explore a space of 10^{30} points. The number of schemata processed in each generation of the GA is approximately $1000^3 = 10^9$. Moreover, bad schemata can be quickly discarded from the exploration by the selection mechanisms, which results with a fairly powerful search procedure with an optimized search space and efficiency.

As a conclusion, *John Holland* provided the *Schema Theorem* which is stated as follows;

Short, low order, above-average schemata receive exponentially increasing trials in subsequent generations of a genetic algorithm.[81]

This theorem is sometimes called as Schemata Theorem as well.

A related observation that emerges when one considers the relationship between schemata and the crossover operator is referred as the *Building Block Hypothesis* and is stated as follows;

A genetic algorithm seek near-optimal performance through the juxtaposition of short, low-order, high-performance schemata, called the building blocks.[82]

Similar to Schema Theorem, the building block hypothesis states that the reproduction, selection and exploration schemes in GA' s are the source for a powerful search. Both in these two definitions, the term short indicates the length of the defining portion of a schema. This is the distance between the first 0 or 1 and the last 0 or 1 which defines the compactness of information contained in that schema. For example, in 1***0*, the defining length is *4* and in **111* it is only *2*. Order (as in low-order) refers to the number of fixed positions (0 or 1) in the schema. For example, the order of the schemata **0*1*1 is *3*, and the order of the schemata 00***00*0 is *5*. It can be seen that schemas of low order match a larger number of individuals than those of high order where a high performance schema is the one that matches individuals of high fitness.[83]

Another fact of the Schema Theorem is that the schemata can also be defined as the bit-wise combinational representations of *hyperplanes*. Thus, a schema of order-3 can be used to implement a *hypercube*. Thus, each binary-encoded chromosome of length l is a member of the 2^l -1 different hyperplanes where 3^l -1 hyperplane partitions can be defined over the entire search space.[84]-[85] Consequently, it' s stated that the implicit parallelism is

---

[80] David Beasley, David R. Bull, Ralph R. Martin, "An Overview of Genetic Algorithms - Part 1", University Computing, UCISA, p. 7, 1993.

[81] Zbigniew Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, p. 51.

[82] *ibid*, p. 51.

[83] *ibid*, pp. 51-53.

[84] L. Darrel Whitley, "A Genetic Algorithm Tutorial", Computer Science Department, Colorado State University, Technical Report, pp. 6-7, November 10, 1993.

[85] Gregory J.E. Rawlins (editor), *Foundations of Genetic Algorithms*, p. 222.

derived from the fact that many hyperplanes are sampled when a population of chromosomes are evaluated.[86]

The derivation of the Schema Theorem and the related assertions can also be explained in a more mathematical form as follows[87]-[88] :

Let H be any schema in a population $P(t)$ at any generation t, where the number of strings or chromosomes of H is denoted by $m(H,t)$. If during any reproduction stage, chromosomes $\{x_1, x_2, \ldots x_n\}$ are regenerated according to the fitness function, then a chromosome x_k of the new population will be reproduced with a probability

$$p_k = \frac{f_k}{\sum_{j=1}^{n} f_j}$$

where f_j indicates the fitness value of the j^{th} chromosome. Thus, the average fitness for the whole population can be denoted as

$$f^* = \frac{\sum_{j=1}^{n} f_j}{n}.$$

Consequently, if an average fitness function value for a schema is denoted by $f(H)$, then the following equation can be stated;

$$m(H,t+1) = m(H,t) \cdot \frac{f(H)}{f^*}. \qquad (1)$$

If, the schema H remains below or above the average of a constant threshold value c, which is independent of t, then the following statement holds;

$$m(H,t+1) = \frac{f^* + c \cdot f^*}{f^*} \cdot m(H,t) = (1+c) \cdot m(H,t). \qquad (2)$$

Therefore, the cardinality of the schema H during any reproduction can be stated by the equation below;

$$m(H,t) = m(H,0) \cdot (1+c)^t \qquad (3)$$

---

[86] L. Darrel Whitley, "A Genetic Algorithm Tutorial", Computer Science Department, Colorado State University, Technical Report, pp. 7-9, November 10, 1993.

[87] Leonard Bole, Jerzy Cytowski, *Search Methods for Artificial Intelligence*, pp. 206-207.

[88] Zbigniew Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, pp. 45-50.

The three equations (1), (2), (3) denote the effects of fitness-proportionate selection on the schema H. On the other hand, if the defining length of H is denoted by δH and the length of each chromosome is given as l, then the probability that a random crossover point is between the defining bits of the schema can be deduced as follows;

$$p_d(H) = \frac{\delta(H)}{l-1}$$

The $p_d(H)$ value is also referred as the probability of disruption of the schema. Thus, a schema marked as disrupted will survive with a probability denoted as below;

$$p_s(H) = 1 - \frac{\delta(H)}{l-1}$$

However, this can directly occur whenever this schema also exists in the other parent. Since only some of the chromosomes can undergo crossover, the probability of a schema survival becomes (the selective probability of crossover is p_c);

$$p_s(H) = 1 - p_c \cdot \frac{\delta(H)}{l-1}$$

On the other hand, since it' s proven that a schema H can be generated by crossing over two strings that do not possess H. Thus, the formula for the probability of schema survival is denoted as;

$$p_s(H) \geq 1 - p_c \cdot \frac{\delta(H)}{l-1}$$

which implies that

$$m(H, t+1) \geq m(H, t) \cdot \frac{f(H)}{f^*} \cdot \left[1 - \left(p_c \cdot \frac{\delta(H)}{l-1}\right)\right] \qquad (4)$$

The equation (4) imposes the crossover effects on the schemata. From this equation it can be deduced that the expected number of chromosomes matching a schema H in the next generation is a function of the actual number of strings having the schema, relative fitness of the schema and its defining length. As stated previously, the crossover probability is mostly set as $p_c < 1$.

Since another effective basic operator is mutation in the population then the similar assertion can be derived for the mutation as well. In general, mutation is applied to all the bits of all the strings with a very low probability $p_m << 1$. It' s also shown that a schema H will be preserved with no alteration if all of its $\delta(H)$ specified positions are unchanged. The

probability of a single bit being prevented from mutation is trivially 1 - p_m. Hence, the probability of a schema that survives mutation can be denoted as follows, where each single bit mutation is independent from the other mutations;

$$p_s(H) = (1 - p_m)^{\delta(H)}$$

since $p_m << 1$, this equation can be rewritten as;

$$(1 - p_m)^{\delta(H)} \approx 1 - \delta(H) \cdot p_m \qquad (5)$$

Finally, combining all the selection, crossover and mutation effects denoted in the equations (1), (2), ...(5) a new form of the reproductive schema growth equation can be derived as below, which is the formal definition of the Schema Theorem;

$$m(H, t+1) \geq m(H, t) \cdot \frac{f(H)}{f^*} \cdot \left[1 - \left(p_c \cdot \frac{\delta(H)}{l-1} \right) - \delta(H) \cdot p_m \right]$$

From this formula, it could also be seen that if only fitness-proportionate selection is applied with no crossover or mutation, generation after generation, a schemata with above average fitness will be sampled by an increasing number of chromosomes (or conversely, a schemata with below average fitness will be observed by a decreasing number of chromosomes).

Considering that crossover tends to cut off schemata as it recombines individuals, it can be firmly stated that short schemata have a higher chance of surviving crossover intact. Low order schema effectively process more individuals and they are also more likely to remain intact.[89]-[90]-[91] Moreover, it' s proven that the positions of the bits in the schema are important in determining the likelihood that those bits will remain together during crossover, whenever single-point crossover is used.[92]

On the other hand, it should also be stressed that; for a number of schemas to combine with each other via any crossover and to construct higher order schemas, all of them must be kept through that crossover. This brings another important fact among GA' s; the importance of *disruptivity* of a schema.[93] Disruption can be defined as the loss of some or all of the schema in the offsprings after a recombination operator, mainly crossover, is applied to the parents preserving that schemata. For instance, if there exists a schemata in a population as (1********1), then whenever the string having that schema

---

[89] L. Darrell Whitley (editor), *Foundations of Genetic Algorithms 2*, pp. 77-79.

[90] *ibid*, pp. 226-228, 230.

[91] Zbigniew Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, pp. 52-53.

[92] L. Darrel Whitley, "A Genetic Algorithm Tutorial", Computer Science Department, Colorado State University, Technical Report, p. 13, November 10, 1993.

[93] Thang Nguyen Bui, Byung Ro Moon, "Genetic Algorithm and Graph Partitioning", IEEE Trans. On Computers, vol. 45, No. 7, p. 843, July 1996.

undergoes a single-point crossover with any string such as (*********0), then the disruption will occur with a probability 1 (recalling that the computation of the probability of the disruption of schema was indicated in this section previously). Because, no matter which bit-location is chosen for the single-point crossover between these two chromosomes, the schema will be lost. This provides an important fact: When using 1-point crossover, the bit-positions of the schema are crucial in determining the probability of the survival of the schema.[94] Another important deduction based on this fact, which can also be analyzed from the example, is that the shorter the schema' s defining length, the less likely it is to be disrupted through crossovers[95]. It' s also proven that the disruption rate of crossover varies among different types of crossover schemes such as *n*-point crossover, parameterized uniform crossover, etc.[96] For instance, for the example given here, any 2-point crossover scheme has a disruption probability (rate) < 1, thus might be a better choice than 1-point crossover for that schemata.

It' s proven that the building block hypothesis has application in both the design of new kinds of crossover operators and in determining how to best encode a problem.[97]

4.5 Potential Problems for GA Implementations

Besides the difficulties that lie in the design and implementation of proper and successful evaluation and fitness functions, there are some other potential problems and shortcomings that can be faced for any GA implementation or in any GA application. Also, in some cases, the designer should consider some of the important facts related with the special properties in the structure of GA' s; because sometimes it might become too difficult for him / her to decide upon which mechanisms to use in the design and implementation. Some of these facts and the potential problems will be mentioned shortly throughout this section.

- **Exploration vs Exploitation**

It' s stated that any efficient optimization algorithm, like GA, should use two basic techniques to find a global maximum; *exploration* and *exploitation*. Exploration is defined as the investigation of new and unknown areas in the search space. On the other hand, exploitation makes use of knowledge found at points previously visited to help finding better points. The crucial issue is that these two requirements are contradictory, when one is kept high in the search, the other becomes low-utilized. Thus a good genetic algorithm must be implemented so as to find a balance between the two, because the power and efficiency of a GA lies in this balance.[98]-[99]

---

[94] L. Darrel Whitley, "A Genetic Algorithm Tutorial", Computer Science Department, Colorado State University, Technical Report, p. 13, November 10, 1993.
[95] Thang Nguyen Bui, Byung Ro Moon, "Genetic Algorithm and Graph Partitioning", IEEE Trans. On Computers, vol. 45, No. 7, p. 843, July 1996.
[96] L. Darrell Whitley (editor), *Foundations of Genetic Algorithms 2*, pp. 222-229.
[97] *ibid*, p. 230.
[98] Zbigniew Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, p. 15.

For instance, a purely random search is good at exploration, but does no exploitation. However, a purely hill-climbing method is good at exploitation but has very little exploration. Making a good balance is the best approach, but this must be carefully implemented in a genetic algorithm' s design. In theory, combining these two techniques in an optimal way at the same time for a GA is shown to be true, whereas some difficulties and inevitable problems are faced in practice. These are shown to be due to the facts that; the population size is not infinite, the fitness function accurately reflects the utility of a solution and the genes in a chromosome do not interact significantly.[100]-[101]

- **Premature Convergence**

There are some common problems observed in GA' s due to the variation in fitness range throughout the execution of a GA. One of them is known as *premature convergence*. The premature convergence is the domination of a few number of highly fit, but not optimal chromosomes throughout the population very rapidly. Since, the population is converged, the GA loses the ability to search for better solutions and sticks on the same type of chromosomes with the same values covering the entire population. Except mutation, all the other reproduction operators produce almost the identical offsprings in the next populations. This turns out the GA into a non-efficient, slow random search.[102]

It' s remarked that as a result of the schemata theorem, premature convergence occurs, because the population size is not infinite. It' s shown that, in order to overcome premature convergence, the selection mechanisms must be rearranged in a way that enables the compression of the range of fitness and the prevention of any super-fit individuals' dominance.[103]

- **Difficulty in Finding The Global Maximum**

Another problem caused by the variation in fitness range is shown to be the difficulty in finding the global maximum, in other words, slow finishing. After many generations, the population will be largely converged, but may still have not precisely located the global maximum (have not found the exact and best solution in optimization problems). Since, the average fitness has reached a high value, there will probably be little difference between the best and average individuals, thus the GA will have difficulty in reaching the global maximum and will wander through the local maxima. This is a very crucial problem when the exact optimum solution is required in time-critical applications. This problem might be overcome by expanding the range of fitness in the population by

---

[99] David Beasley, David R. Bull, Ralph R. Martin, "An Overview of Genetic Algorithms - Part 1", University Computing, UCISA, p. 8, 1993.
[100] *ibid*, p. 8.
[101] L. Darrell Whitley (editor), *Foundations of Genetic Algorithms 2*, p. 231.
[102] David Beasley, David R. Bull, Ralph R. Martin, "An Overview of Genetic Algorithms - Part 1", University Computing, UCISA, p. 10, 1993.
[103] *ibid*, p. 10, 1993.

some special mechanisms such as *explicit fitness remapping* and *implicit fitness remapping*. Also, some random mechanisms can be added to the selection and recombination operations so as to introduce new individuals that might help reaching the global maximum.[104]

- **Crossover / Mutation Balance**

It' s mentioned previously that in general, for any GA, crossover and mutation operators are used together in recombination phase, where crossover probability is significantly high and mutation probability is too low. However, some researchers suggest completely discarding the mutation operator but some others advise keeping mutation probability high, even higher than the probability of crossover in some occasions. No matter what kind of problem is dealt, and which GA model is chosen, the designer should be aware of some of the facts concerning the crossover / mutation balance.

It' s shown that mutation can provide any level of disruption that crossover can provide. Moreover, it' s proven that crossover has no advantage over mutation in terms of the amount of exploration that can be performed. However, considering the type of exploration, crossover is proven to guarantee the preservation of common alleles, while mutation does not. Another fact is that, crossover can exhibit high simultaneous levels of preservation, survival and construction since it can share information between the fit individuals. But since mutation is often implemented with a parameter that is constant during the search, no information can be shared by mutation unless it' s implemented in a variable fashion. However, it' s proven that crossover implicitly concerns the interaction among the genes when generating new instances, a property not valid for the standard mutation operator. This property is shown to be the source of crossover' s power as a search operator.[105]-[106]

When optimizing the balance between exploitation and exploration in a GA is concerned, it' s stressed that neither mutation nor crossover must be dismissed. Instead, they should be both used with an optimized balance between their probabilities. It' s proven that mutation plays an important role in the exploration strength of a GA, while crossover heavily affects the exploitation quality.[107]

4.6 Other GA Models

The standard GA model previously described, sometimes is proven to be insufficient in some applications, especially whenever the evaluations are very time-consuming and the populations need to be held extremely large. For such situations, different GA models have been successfully derived and yet still being constructed. One of

---

[104] David Beasley, David R. Bull, Ralph R. Martin, "An Overview of Genetic Algorithms - Part 1", University Computing, UCISA, pp. 8-12.
[105] L. Darrell Whitley (editor), *Foundations of Genetic Algorithms 2*, pp. 226-232.
[106] *ibid*, pp. 200-201.
[107] *ibid*, p. 231.

these is known as *Parallel Genetic Algorithms (PGA)*. It' s shown that PGA' s are not only an extension of the standard sequential GA via parallelism, but also they represent a new class of algorithms that have a different search mechanism. There are various types of PGA models, depending on; the distribution of the population (*Centralized, Distributed*, etc.), depending on the implementation (*Farming, Island, Cellular*) and depending on the Parallel Model (*Coarse-Grained, Fine-Grained*, etc.).[108]-[109]-[110]

Another GA model is known as *CHC* (*Cross-generational elitist selection, Heterogeneous recombination, Cataclysmic mutation*), which is a non-traditional genetic algorithm. It processes sequentially as the standard GA' s, but it combines a conservative selection strategy that always preserves the best individuals found so far with a highly disruptive recombination operator that produces offsprings that are maximally different from both parents. The CHC algorithms are shown to perform better than the standard GA' s in some specific implementations.[111]-[112]

There are also other different GA models such as *messy genetic algorithms (MGA), Delta Coding, The Vose and Liepins models, Genitor, Dynamic Parameter Encoding*, etc. These all differ from the traditional GA' s in some or all of the schemes, such as representation, operators, selection, phases of the evolution process.[113]-[114]-[115]

4.7 Applications of Genetic Algorithms

In this section some GA applications are summarized among a broad range of application areas that; GA' s are proven to be successful, or might be useful in the near future.

- **Scheduling**

Scheduling can be defined as a problem of finding the optimal sequence to execute a finite set of operations such that a certain set of constraints are not violated. A schedular usually attempts to maximize the utilization of individuals or machinery and minimize the time required to complete the entire process being scheduled. Conflicts can arise when an individual or machine is scheduled for more than one operation at a given time or when

---

[108] Lawrence Davis (editor), *Genetic Algorithms and Simulated Annealing*, pp. 129-140.
[109] Gregory J.E. Rawlins (editor), *Foundations of Genetic Algorithms*, pp. 316-335.
[110] L. Darrel Whitley, "A Genetic Algorithm Tutorial", Computer Science Department, Colorado State University, Technical Report, pp. 32-34, November 10, 1993.
[111] *ibid*, p. 30.
[112] Gregory J.E. Rawlins (editor), *Foundations of Genetic Algorithms*, pp. 265-283.
[113] L. Darrel Whitley, "A Genetic Algorithm Tutorial", Computer Science Department, Colorado State University, Technical Report, pp. 27-31, November 10, 1993.
[114] Zbigniew Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, pp. 65-72, 135-138.
[115] David E. Goldberg, Kalyanmoy Deb, Hillol Kargupta, Georges Harik, "Rapid, Accurate Optimization of Difficult Problems Using Fast Messy Genetic Algorithms", University of Illinois, Urbana-Champaign, IlliGAL Report No. 93004, pp. 1-17, February 1993.

the schedule utilizes more resources than are available. Other constraints include priority of some tasks over others and precedence of certain tasks with respect to others. GA' s can be applied successfully and do show a great performance in several scheduling areas like job-shop scheduling, transportation and assignment problems (*GENOCOP*), resource scheduling (in laboratories, several good stocks, etc.), ERP, MRP, control systems *(GAFOC)*.[116]-[117]

- **NP-Complete and NP-Hard Problems**

The solutions for various forms of complex problems such as NP-Complete (Non-Deterministic Polynomial Complete) and NP-Hard (Non-Deterministic Polynomial Hard) have been successfully implemented using GA' s. These include the traveling salesman, map coloring, timetable assignment and graph partitioning problems.[118]-[119]-[120]

- **Engineering and Design**

Optimizing a design that includes many different parameters is a time consuming and difficult task, even for expert engineers. GA' s can assist in design by quickly heading toward near-optimal designs, given certain user specified design constraints. Aircraft and space shuttle designs can be given as examples. GA' s also offer a great aid in various kind of industrial engineering applications, control systems, etc.[121]-[122]-[123]

- **Machine Learning**

Machine learning is mainly based on building computer programs that are able to construct new knowledge or to improve already possessed knowledge by using the input information. There are several models of machine learning in which GA' s have been successfully applied. For instance, GA' s have been applied to classifier systems where the GA tries to evolve a set of *if ...then* rules to deal with some particular condition or

---

[116] Zbigniew Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, pp. 99-126, 203-209.

[117] Christian Bierwirth, Herbert Kopfer, Dirk C. Mattfeld, Ivo Rixen, "Genetic Algorithm Based Scheduling in a Dynamic Manufacturing Environment", Department of Economics, University of Bremen, Germany, Technical Report, pp. 1-6.

[118] Zbigniew Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, pp. 165-200, 209-214.

[119] Thang Nguyen Bui, Byung Ro Moon, "Genetic Algorithm and Graph Partitioning", IEEE Trans. On Computers, vol. 45, No. 7, pp. 841-854, July 1996.

[120] Kenneth A. De Jong, William M. Spears, "Using Genetic Algorithms to Solve NP-Complete Problems", ICGA-89 The Third International Conference on Genetic Algorithms, pp. 1-9, June 4-7, 1989.

[121] Sigurd A. Nelson, "Strain Gage Selection in Loads Equations Using a Genetic Algorithm", NASA Contractor Report 4597, pp. 1-20, 1994.

[122] George W. Ryan, "A Genetic Search Technique for Identification of Aircraft Departures", NASA Contractor Report 4688, pp. 1-16, 1995.

[123] James L. Rogers, Collin M. McCulley, "Integrating a Genetic Algorithm into a Knowledge-Based System for Ordering Complex Design Processes", NASA TM-110247, pp. 1-13, April 1996.

problem. GA' s are also used in other learning models, such as connectionist network models and symbolic systems.[124-125-126]

- **Telecommunications**

Routing of information through a complex communications network is an extremely important problem in today' s world filled with computer networks, cellular networks, packet switching networks and the like. This is a tough problem that some researchers have addressed the solution using GA' s. And also, the optimization of network channels in terms of speed, load balances, etc. can be achieved via GA usage.[127]

- **Security in Computer Systems**

GA' s can also be used efficiently in several areas of security in computer systems. For example, a very successful implementation is achieved for the security audit trail analysis problem, where, GA enables both an automotive efficient tool for audit trail processes and also a simulation system for various kinds of attacks and threats to a computer system, an intrusion detection system namely. A tool and program has been developed for this aim which is known as *GASSATA*. (a Genetic Algorithm tool for Simplified Security Audit Trail Analysis) It' s been developed by *M.E. Ludovic* and used in *IBM AIX* systems successfully with proven reliability and efficiency.[128]

---

[124] W. F. Punch, E. D. Goodman, Min Pei, Lai Chia-Shun, P. Hovland, R. Enbody, "Further Research on Feature Selection and Classification Using Genetic Algorithms", ICGA93, pp. 1-8, 1993.

[125] Zbigniew Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, pp. 215-229.

[126] Lawrence Davis (editor), *Genetic Algorithms and Simulated Annealing*, pp. 129-140.

[127] Vladimir N. Davidenko, Victor M. Kureichik, Victor V. Miagkikh, "Genetic Algorithm for Restrictive Channel Routing Problem", ICGA97 The Seventh International Conference on Genetic Algorithms, pp. 1-6, July 19-23, 1997.

[128] M.E. Ludovic, "Genetic Algorithms, an Alternative Tool for Security Audit Trails Analysis", Internet Document, http://www.supelec-rennes.fr/rennes/si/equipe/lme/these/oakland95/oakland95.html, 1997.

Chapter 5

CRYPTANALYSIS OF SYMMETRIC CIPHERS USING GA' S

5.1 Previous Studies

It' s been proven that there haven' t been much study on the cryptanalysis of symmetric ciphers using genetic algorithms (GA' s), especially when differential or linear cryptanalysis is the case. However, in conjunction with this issue, there are some remarkable works or projects that has been formerly established or being constructed during the writing time of this thesis. But, against all my efforts, I have been able to find only a few information about these studies. Only one of these projects, *The Decryption of Rotor Ciphers Using Genetic Algorithms*[1] namely, had detailed and accessible information via Internet, and it will be explained in the following paragraphs.

The first known studies combining the genetic algorithms and the cryptanalysis is shown to be derived by *Prof. Richard Jay Spillman*. In one of his studies, with the assistance of some other researchers, he successfully established a genetic algorithmic method to break the keys of mono-alphabetic simple substitution ciphers as an alternative to brute-force type cryptanalytic attacks.[2] In a reference[3] concerning this study, it' s been mentioned that randomly chosen keys were evaluated for fitness using letter and diagram frequencies and the fittest candidates were then *mated* and subjected to mutation to provide the next generation of keys. The mating operation used a selective crossover, in which the best character of each key was passed on.

Following this research, *Spillman*' s another successful study is shown to be the cryptanalysis of *Knapsack Ciphers* using genetic algorithms.[4] It' s mentioned that in his study, he described the design and use of a genetic algorithm to attack *small trap do* or *knapsacks* effectively and gave performance data to prove that GA-based search performed *50 - 100* times faster than exhaustive search.[5]

Another researcher who made several studies based on the usage of GA' s in cryptanalytic attacks is *R. A. Matthews*. In one of his studies, he established a genetic system for the cryptanalysis of simple transposition ciphers.[6] In this system, candidate column orders were assessed for fitness using *digram* frequencies, and the best of them were used to breed a new generation of candidates, using column rotations and swaps

---

[1] Emergent Technologies Inc., "On the Decryption of Rotor Ciphers Using Genetic Algorithms", Internet Document, http://www.sys.uea.ac.uk/~ajb/rotor.html, September 1997.

[2] R. Spillman, M. Janssen, B. Nelson, M. Kepner, "Use of a genetic algorithm in the cryptanalysis of simple substitution ciphers", Cryptologia, v. 17, No. 1, pp. 31-44, January 1993.

[3] Cypherpunks, "Re: Coding and Nnet's", Internet Document, http://www.inet-one.com/cypherpunks/dir.95.11.08-95.11.14/msg00209.html, May 1998.

[4] R. Spillman, "Cryptanalysis of knapsack ciphers using genetic algorithms", Cryptologia, v. 17, No. 4, pp. 367-377, October 1993.

[5] Cypherpunks, "Re: Coding and Nnet's", Internet Document, http://www.inet-one.com/cypherpunks/dir.95.11.08-95.11.14/msg00209.html, May 1998.

[6] R. A. Matthews, "The use of genetic algorithms in cryptanalysis", Cryptologia, v. 17, No. 2, pp. 187-201, April 1993.

as mutation operators. The algorithm was proven to be successful in finding partial anagrams to aid manual solution.[7]

One of the most impressive and outstanding application among the cryptanalytic attacks with GA' s is shown to be the cryptanalysis of rotor ciphers. The design and implementation of this study will be discussed shortly in the following paragraphs.

A rotor cipher is a mechanical time-varying substitution cipher. In the times of First World War, rotor ciphers were mostly preferred in cryptography. Also, its variants such as *Enigma Cipher* were used in the Second World War. However, as the power and the applicability of computer technology increased, the security of the rotor ciphers decreased significantly and were not used any more after the 1950' s. But it should be stressed that nowadays, a rotor cipher is still considered as a good challenge to test a cryptographic method' s breaking capacity. Failure to decrypt a rotor cipher would be indicative of a failure in the algorithm of the cryptanalytic attack, or low performance in the cryptanalysis of any rotor cipher via any kind of cryptographic method imposes that method' s inefficiency.[8]

The original rotor cipher was a physical device made of an insulating disk which had a number of electrical contacts around the perimeter of each face. The contacts on one side of the disk are connected randomly to the contacts of the other side. Thus, if each letter of the input alphabet is assigned to a contact on one face, the output contacts would be a monoalphabetic substitution cipher. By connecting several rotors face to face, a product of substitutions can be achieved. The complexity of the cipher is obtained by rotating some of the rotor disks in a certain fashion after each character is encrypted. By this way, a highly complex polyalphabetic cipher with a moderately long key length can be derived.[9] (the so-called *certain fashion* used in rotation after each encryption provides the key itself.) This cipher is proven to be ported easily to computers, either in hardware or software.

There are various cryptanalytic methods previously invented to break the rotor cipher efficiently such as *the iterative technique*.[10] A recent alternative approach is implementing the cryptanalytic attack to several rotor cipher models by the aid of genetic algorithms, which has been successfully established by *A. J. Bagnall* et al. It' s shown that a genetic algorithm can be used to successfully search the very large discrete key-space of a rotor machine, of magnitude up to $(26!)^3$, using a simple measure of suitability. The method involves finding the last rotor of a three-rotor machine using a GA and then solving the remaining two rotors by the iterative

---

[7] Cypherpunks, "Re: Coding and Nnet's", Internet Document, http://www.inet-one.com/cypherpunks/dir.95.11.08-95.11.14/msg00209.html, May 1998.

[8] Emergent Technologies Inc., "On the Decryption of Rotor Ciphers Using Genetic Algorithms", Internet Document, http://www.sys.uea.ac.uk/~ajb/rotor.html, September 1997.

[9] *ibid.*

[10] A. J. Bagnall, "The Applications of Genetic Algorithms in Cryptanalysis", MS Thesis, University of East Anglia, pp. 85-92, 1996.

technique.[11] The cryptanalysis of two-rotor machines by using GA' s is also proven to be more efficient than the other attack methods where it' s also suggested that the four-rotor and Enigma ciphers might also be cracked using enhanced GA' s.[12]-[13]

For the GA implementation, the rotor is converted into an array of m integers, where each integer is a 32-bit chromosome, representing the wiring for each position. By this way, the subset of the key-space containing the correct keys for an m-rotor cipher can be represented with a population of chromosomes. Several selection methods were tried and roulette selection was proven to be the most suitable for the implementation, producing faster convergence to an optimal solution (the correct key value). Also, since *PMX* performed the best results, it was chosen as the basic crossover operator. Two mutation operators were used; randomly swapping two rotor wirings as a reciprocal exchange and randomly shifting selected substring of rotor wirings to random position. Either one of these operators were chosen randomly and operated on the offspring through each reproduction. It was found that the optimum probability level for mutation was *0.5* for all the attacks. For the construction of the new population, only the offsprings that had better fitness than the worst member of the current pool and that were different from the chromosomes in the current pool were added to the new population replacing the worst ones.[14]-[15]

The core and the most essential part of the GA implementation was pointed to be the fitness function because, unless a proper fitness function is derived so as to choose the candidate keys among the entire key-space, the cryptanalytic attack with any GA would be useless. The basis for the fitness evaluation was that plaintext enciphered by a monoalphabetic substitution would still exhibit characteristics of the original distribution. A periodic polyalphabetic substitution system with period P can be reduced to a series of monoalphabetic substitution ciphers. Therefore, a likelihood function can be used as the fitness measure for which the search space of an m rotor machine is m permutations and a chromosome p' has fitness $L(p'|y)$ as[16] ;

$$L(p'|y) = P(y_0|p') \times P(y_1|p') \times \ldots \times P(y_{n-1}|p') \qquad \text{for a ciphertext of size } n$$

However, it was shown that using such a likelihood function as a fitness measure caused a problem due to the fact that; for any reasonably large string size the likelihood would be a very small value, yet with an extreme range of possible values. Thus, if roulette selection was used, poor solutions could possibly dominate the

---

[11] A. J. Bagnall, G. P. McKeown, V. J. Rayward-Smith, "The Cryptanalysis of a Three Rotor Machine Using a Genetic Algorithm", ICGA97 The Seventh International Conference on Genetic Algorithms, p. 1, July 19-23, 1997.

[12] A. J. Bagnall, "The Applications of Genetic Algorithms in Cryptanalysis", MS Thesis, University of East Anglia, pp. 130-139, 1996.

[13] Emergent Technologies Inc., "On the Decryption of Rotor Ciphers Using Genetic Algorithms", Internet Document, http://www.sys.uea.ac.uk/~ajb/rotor.html, September 1997.

[14] A. J. Bagnall, G. P. McKeown, V. J. Rayward-Smith, "The Cryptanalysis of a Three Rotor Machine Using a Genetic Algorithm", ICGA97 The Seventh International Conference on Genetic Algorithms, pp. 3-5, July 19-23, 1997.

[15] A. J. Bagnall, "The Applications of Genetic Algorithms in Cryptanalysis", MS Thesis, University of East Anglia, pp. 121-126, 1996.

[16] *ibid*, p. 126.

population because their fitness would be much greater than the other chromosomes, but still much smaller than the optimum. It was proven that taking the logarithmic values of each side of the equation in the fitness function and consequently, using *-log(L(p'|y))* as the fitness measure could overcome this problem. Thus, the objective is to find the minimum of the negative log likelihood that corresponds to the optimum solution. The revised fitness function is denoted as follows[17] ;

$$\log\left(L\left(p'|y\right)\right) = \log\left(P\left(y_0|p'\right)\right) + \log\left(P\left(y_1|p'\right)\right) + \ldots + \log\left(P\left(y_{n-1}|p'\right)\right)$$

Using the genetic mechanisms and functions described previously, the GA was coded and executed. The program generates new populations and executes through an evolution cycle in an iterative manner until a chromosome with the required optimum, thus a correct or nearly correct rotor is found. This also means that the key for that rotor is broken.

After the tests and several studies it was proven that by using a GA-based attack, the single rotor cipher was broken much more rapidly and efficiently than all the other known cryptanalytic techniques. For instance, a GA could efficiently search the discrete key-space of a size about 10^{44} and could find the key by examining only 10^5 - 10^6 candidates. It was shown that an unknown plaintext decryption of a single ciphertext enciphered in a 32-contact rotor could be achieved by a GA at an average of 20 minutes on the *12 Mhz 80286* platform. Thus, when the GA is embedded to a standard cryptanalytic attack, the overall cryptanalytic performance was proven to be better than all the other attack types on two and three-rotor models. Moreover, it was also shown that the GA embedded attack technique outperformed other search mechanisms such as simulated annealing and hill-climbing when these were embedded to the cryptanalytic attack in a similar manner. In all these studies as a conclusion, it was firmly stated that a very basic implementation of a GA could be used to solve a fairly hard cryptographic system.[18]-[19]-[20]

Besides these former studies that have been mentioned previously, there are also currently ongoing projects based on the cryptanalysis of different types of cryptosystems using genetic algorithms. For instance, one of these studies aims to derive a general mechanism to enhance key searches in cryptanalytic attacks using GA-based techniques; by making the best use of good global search capability of GA' s.[21] Another project that has been recently founded and is still being developed; is the cryptanalysis of DES via transformation of the DES algorithm into an instance of an *NP-Hard* combinatorial problem and attacking with an appropriate combinatorial

---

[17] A. J. Bagnall, "The Applications of Genetic Algorithms in Cryptanalysis", MS Thesis, University of East Anglia, pp. 126-127.

[18] Emergent Technologies Inc., "On the Decryption of Rotor Ciphers Using Genetic Algorithms", Internet Document, http://www.sys.uea.ac.uk/~ajb/rotor.html, September 1997.

[19] A. J. Bagnall, G. P. McKeown, V. J. Rayward-Smith, "The Cryptanalysis of a Three Rotor Machine Using a Genetic Algorithm", ICGA97 The Seventh International Conference on Genetic Algorithms, pp. 5-6, July 19-23, 1997.

[20] A. J. Bagnall, "The Applications of Genetic Algorithms in Cryptanalysis", MS Thesis, University of East Anglia, pp. 130-140, 1996.

[21] ADFA Computational Intelligence Group Projects, "Applications of Evolutionary Algorithms in Cryptanalysis", Internet Document, http://www.cs.adfa.oz.au/research/CIG/projects.html, April 1999.

algorithm.[22] In this project, one of the possible and promising alternative for the combinatorial algorithm is recommended as GA.

5.2 Proposed Theoretical Model Differential / Linear Cryptanalysis using Genetic Algorithms

The implementation of any differential or linear cryptanalytic attack via any genetic algorithmic approach is the core of this thesis. However, due to some important and strict aspects and shortcomings, this approach has been in a proposed theoretical or hypothetical fashion rather than an implementation; where these reasons will be explained later on.

Having made a detailed analysis both in differential / linear cryptanalysis and GA' s, my basic approach was to find a suitable methodology to combine these two issues which might possibly result with an exploitable effort. I also included the knowledge and concepts involved with the previous cryptanalytic attacks achieved via the usage of GA' s. Using these former information, and combining these with my analyzes and ideas, I tried to establish several alternative models related with the differential or linear cryptanalytic attacks using GA' s. I tried to generate alternative models each having a different view of standpoint in the use and application of GA' s among these cryptanalytic attacks.

- **New Characteristics in Differential Cryptanalysis using GA' s**

As mentioned previously, there are several best *n-round* characteristics derived for the differential cryptanalysis of DES and DES-like symmetric cryptosystems. However, as stated by *Biham* and *Shamir*, there might be better differential characteristics than the best ones found so far for the DES algorithm.[23] Since, it' s shown that the derivation of better characteristics in any type of differential cryptanalytic attack is the heart of this attack methodology; better characteristics shall improve the performance of the attacks significantly. For instance, if better characteristics could be established than the one used in the differential cryptanalysis of 16-round DES, having a higher probability than $2^{-47.2}$, then the overall computational complexity of the attack and the chosen plaintext requirements could be reduced too.

Therefore, if a suitable GA can be adapted so as to search and find better differential characteristics than the known ones, then the differential cryptanalysis can be improved with the aid of GA' s. This might be achieved by constructing a GA which takes several chosen plaintext pairs that produce the good and the best characteristics known so far and several random plaintext pairs as input. These pair values will be encoded so as to be the chromosomes of the GA population. The fitness function will be trivial, the probability of the characteristic provided within these chromosomes. Thus, the more fit chromosomes with higher probabilities will be used for reproduction, and if any chromosome having a near-optimum (best known characteristic) or equal-optimum or indeed, a better-than-optimum is produced, then

[22] Eric Michael Cordian, "The DES Analytic Crack FAQ", Internet Document, http://www.cyberspace.org/~enoch/crakfaq.html, December 1998.
[23] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, p. 28.

its ciphertext pairs will be checked. If these are different from the previously derived characteristics, they will be used as new and best (if any) characteristics in the differential cryptanalysis. However, the reproduction operators, selection and replacement processes and other necessary genetic mechanisms can be decided and properly adapted whenever this hypothetical model is transformed into an implementation. This fact holds for the other methods described in the following subsections as well.

It should be stressed that since any characteristic' s probability is found by checking the output differential of ciphertext pairs and the input differential plaintext pairs; in the GA, for each new chromosome (the input pairs) the encryption algorithm should be executed and the corresponding ciphertext differentials must be derived. Another important issue is that, since the best differential characteristics vary with respect to the number of rounds, this must also be distinguished in the GA search. Thus, for any specific number of rounds, the GA' s must be implemented and executed separately so that each GA focuses on finding the better differential characteristics of the cryptosystem for that specific n-round version.

- **Enhancement of the Data Analysis Phase in Differential Cryptanalysis using GA' s**

As discussed previously in Chapter 3, both for the differential cryptanalysis of DES reduced to some rounds and 16-round DES, some special look-up tables, searching schemes and key-counting mechanisms were used to find some portion of the key bits apart from the key bits found by the characteristics, which was referred as the data analysis phase of the differential cryptanalysis. If, a GA-based search can be used instead of the standard look-up mechanisms, then the key bits would be extracted in less computational time. With the aid of a GA' s, less number of decrypted ciphertext samples can be compared with the expected XORed plaintext values in the data analysis phase. Thus, the ciphertext data will be mapped into chromosomes for the GA model, and the fitness measure will be the result of the comparison. Another GA enhancement might be in the key-bit counting used for subkey bits in any specific S-boxes. By encoding the chromosomes for the subkeys and transforming the selection and checking mechanism used in key-bit counting into a fitness function, some subkey bits might presumably be broken faster. Since, the basic approach in all parts of the data analysis phase is checking all the possible combinations of data and some specific key bits instead of using heuristic exploration mechanisms, GA' s can be used providing more efficiency and less execution time.

- **New Characteristics in Linear Cryptanalysis using GA' s**

A similar approach to the one used in the differential cryptanalysis might also be helpful for the linear cryptanalytic attacks. Since, the characteristics are also the core of the linear cryptanalysis, GA' s can be used in deriving better linear characteristics than the known best ones. In linear cryptanalysis of DES and DES-like cryptosystems, as stated in the differential case, there might be better characteristics with better probabilities for any number of rounds. The general model can also be designed similar to the GA model used for differential cryptanalysis. However, this time the chromosomes had to be composed of some bits of plaintext, ciphertext and

the key due to the structure of linear cryptanalysis. For each chromosome, the linear approximation is applied and thereafter, its corresponding probability and thus, the linear characteristic can be generated. The fitness function checks the probability of each newly generated characteristic within that chromosome and selects the fittest ones among the highest probabilities, in other words, the best linear approximations. As similar to the model for the differential version, the GA for the linear characteristics tries to converge or find new linear characteristics with higher probabilities than the known ones. Thus, these characteristics can be used to improve the linear cryptanalytic attacks decreasing the known plaintext / ciphertext requirements and hopefully, finding out more number of key bits with good possibilities. For each different n-round linear characteristics of the cipher, a different GA implementation should be applied and executed separately.

- **Using GA' s for the Generalization of Linear Cryptanalysis**

As discussed in Chapter 3, there are several methods which enhance the standard linear cryptanalysis such as generalization techniques. GA' s might be useful in implementing these techniques since some of these generalization methods are NP-hard or NP-complete problems.

For instance, non-linear approximations is proven to be one of these methods. By using non-linear approximations in DES or DES-like cryptosystems, the generalization of the linear cryptanalysis for these cipher models are proven to be established successfully.[24] The non-linear approximations can be derived for any cryptosystem by analyzing the non-linearity between the plaintext, ciphertext and the key bits. However, this analysis is proven to be more complex and hard to implement successfully sometimes. Exploiting the search and the optimum finding capacity in non-linear problems, GA' s can be adapted to non-linear approximation techniques so that the cryptanalytic performance might be improved. It should be noted that non-linear approximations try to find non-linear properties within the cipher with the best possible estimated probabilities where an adaptive GA can find these properties much more efficiently. The chromosomes could represent the relevant data and key bits where each chromosome value consume the parameters for various non-linear approximations. Thus, calculating the non-linear approximations as the problem functions of the GA, the problem can be solved for the ones having the most fit values, in other words, the highest probabilities. Several GA' s might be derived for several different non-linear properties and their approximations where each fitness measure corresponds to the solution of the non-linear equation.

It should be noted that if any linear or non-linear technique can be useful for the generalization of linear cryptanalysis, which results with the improvement of the linear cryptanalytic attack performance; then that technique might be transformed into a GA-based generalization. Such an approach should possibly improve the performance of the attack further more, by decreasing both the computational complexity and the data requirements.

---

[24] Lars R. Knudsen, Matt Robshaw, "Non-linear Approximations in Linear Cryptanalysis", Advances in Cryptology - Proc. EUROCRYPT'96, p. 224, 1996.

- **Enhancement of the Bit Counting Phase in Linear Cryptanalysis using GA' s**

Within a typical linear cryptanalytic attack, some of the (sub)key bits are found by data and key-bit counting schemes after the linear approximations and characteristics. By analyzing the effective key and data bits, some bits of the key can be deduced after the extraction of these effective bits. However, so far it' s proven that only 25 key bits of 16-round DES can be guessed by the bit counting mechanisms.[25] Since it' s shown that analysis of the effective bits and their counting is a tedious and time consuming effort (all the possible bit combinations are used by making a search among those), a better method might improve the efficiency. Using a GA-based search mechanism, this counting method can be improved significantly where the same number of bits can be deduced with equivalent probabilities, but in less time. The chromosomes could represent the effective data and key bits, and by using the equalities with given probabilities, the possible key bit values could give the fitness measure of each chromosome where the best results suggesting the given probabilities yield to the fittest ones, thus the possible key bit values. The fitness function only tries to select which key-bit values satisfy the equations with the given probabilities. It should be stressed that for the GA implementation of effective bit analysis and counting; the effective bits, the equations and their probabilities should already be ready in hand.

5.3 Remarks and Future Work

It should be remarked that due to some difficulties in the implementation and due to the infeasibilities and inapplicability, the models discussed in this chapter were limited to theoretical assumptions. As explained in Chapter 3, section 3.1.1 and in some other previous sections, the differential and linear cryptanalytic attacks require chosen or known plaintext / ciphertext data to be generated in sufficient amounts so as to develop any further studies among such cryptanalysis techniques. However, these attacks are proven to be largely theoretical so far because very huge amount of time and data requirements made it beyond the reach of almost everyone as well as me. The achievement of the necessary data is proven to be requiring several years' time. Besides the collection of the data, the analysis and computational requirements bring extra overhead. Moreover, even these requirements were fulfilled, the hardware requirements were too costly and beyond our capacity which made my study inapplicable. For instance, the minimal storage consumption for a linear cryptanalytic attack to DES shall yield to $2*8*2^{43} = 2^{47}$ bytes ($\approx 10^4$ Gigabytes) and similarly for a differential cryptanalytic attack to DES should be to $2*8*2^{47} = 2^{51}$ bytes ($\approx 2^4*10^4$ Gigabytes) which are extreme values for today. Thus, even having the data ready would not be sufficient so as to make the study applicable.

On the other hand, I had no means of accessing or granting the known or chosen plaintext / ciphertext pairs formerly developed. Therefore, since I had no accessible data; and had not any sufficient time, computational power and storage medium to create and use such data for the differential and linear cryptanalysis, the genetic algorithmic approaches to these attack types could not be implemented. In

[25] S. Bakhtiari, R. Safavi-Naini, "Application of PVM to Linear Cryptanalysis", University of Wollongong, Technical Report, p. 5, July 25, 1994.

order to successfully establish any applications involved with differential / linear cryptanalysis using GA' s among DES and DES-like ciphers, these requirements must be satisfied first.

I should also point some important facts for any future studies involved with genetic algorithmic approaches to differential / linear cryptanalysis. First of all, it' s been mentioned throughout this thesis that; for the studies related with differential / linear cryptanalysis, a remarkable amount of the key bits were found using exhaustive key-bit search, especially for the linear cryptanalytic attacks. But, since it' s proven that any statistical attacks among DES and DES-like ciphers were not efficient, and no direct dependencies or statistical relations could be found for the key bits and data bits of these cipher models, then any GA model would not be meaningful if it focuses on improving the exhaustive key search part of the cryptanalytic attack.[26] (This phenomenon will be given in more detail in Chapter 6, section 6.3.) Unless a statistical attack is possible among any cipher system, any GA-based key search better than the exhaustive key search cannot be adapted. Therefore, for any genetic algorithmic approach one should focus on improving the techniques for the parts other than exhaustive key-bit search of differential / linear cryptanalysis.

Another important fact is since GA' s are implementational rather than theoretical due to their very nature, any innovative ideas, new visions or techniques for the GA' s might be necessary for the differential / linear cryptanalysis. As stated in my theoretical models, the best suitable and the most efficient reproduction, selection, replacement operators could only be decided whenever the attacks are implemented and the necessary algorithms are executed. Thus, any researcher involved with implementations of differential / linear cryptanalytic attacks using GA' s, he / she should also be open to new ideas, methods so as to derive new selection, reproduction operators, etc. which might be more successful and suitable than the ones found so far. I specifically remark this fact because differential / linear cryptanalysis is a very unique and new era for the genetic algorithmic implementations which seems basically different from all the other application areas of GA' s.

The parallelization of the genetic algorithmic approach to differential / linear cryptanalysis might be another future work topic. Since both differential / linear cryptanalytic attacks and some GA' s were successfully implemented in parallel, then a parallel model for any GA-based differential / linear cryptanalysis could be constructed as well. Since the parallel implementations were proven to improve the performance of any differential or linear cryptanalytic attack significantly, then a parallel GA model could be adjusted to such implementations which might further improve the performance of the parallel differential / linear cryptanalysis.

[26] A. J. Bagnall, "The Applications of Genetic Algorithms in Cryptanalysis", MS Thesis, University of East Anglia, p. 95, 1996.

Chapter 6

CASE STUDY

In this chapter, a simple case study concerning the usage of genetic algorithms (GA' s) in cryptanalysis will be given. This study' s aim is to analyze and validate the performance of a cryptanalytic attack to a simple symmetric block cipher using a genetic algorithm. I generated this sample cryptosystem which might be considered as a very simplified model of DES and DES-like ciphers. Then, a brute-force attack using a single plaintext / ciphertext pair is applied to this cryptosystem to find out the keys used for several encryption tests. Finally, a GA-based cryptanalytic attack is applied to the same cipher for the same keys. Thus, making a comparison between the results of the exhaustive key search and the key search using GA, the efficiency and effectiveness of my genetic algorithmic approach to the cryptanalysis of the sample cryptosystem is evaluated. It should be remembered that any cryptanalytic attack method must perform at least better than an exhaustive search in order to be accepted as a valuable and suitable technique.

The first phase of my study was the design of the cipher model and the model of the cryptanalytic attacks, both the brute-force and the genetic algorithm versions, which will be applied to the cryptosystem. The second phase was the implementation of these models, by generating the necessary algorithms and executing these algorithms via software codes written in C language.

6.1 Design of the Model

For the first part of the design, I chose a symmetric block cryptosystem. For the sake of simplicity and the ease of implementation, this system takes a plaintext data as input, encrypts in k-bit size blocks with a k-bit initial random key for which each data block will be enciphered independently known as the ECB mode. The decryption process is exactly the symmetric inverse of the encryption. Each ciphertext block will be decrypted with the same initial key and the same encryption algorithm, producing the original plaintext without error and at the same process rate.

The inner structure of encryption and decryption algorithms are composed of only a simple substitution operation and will be applied in an iterative manner for any arbitrary number of m rounds. For each round, the initial key will be shifted n bits to the left and the corresponding k-bit subkey will be used in the substitution operation within the k-bit data block. The n value can be any integer ranging from 1 to $k-1$, and can be any different value for each round. The k, m and n could be any integer value, however, in the implementation, k was chosen 32 due to some necessary criteria which will be explained later on.

To keep the model simple, there were no S-boxes, complex substitution functions and transpositions in the cryptosystem. This was a crucial and necessary design criteria because the memory and processing requirements for the cryptanalysis phase had to be kept limited. Since, the aim in this case study is the comparison of an exhaustive key search with a GA-based attack, the security quality or the complexity of the cryptosystem itself is not an essential issue. In my model, the cryptosystem serves

only as a test-pad for the comparison of attack techniques in a fairly and feasible fashion as well as providing the basic common features of any iterated symmetric block cryptosystem in a primitive manner.

The design of the cryptosystem is also denoted in the figures 6.1 and 6.2 where in the second figure, the inner structure of the encryption algorithm is given. The scheme in the decryption algorithm is exactly the same as the encryption algorithm; the only difference is that the input at the first round is the ciphertext block.

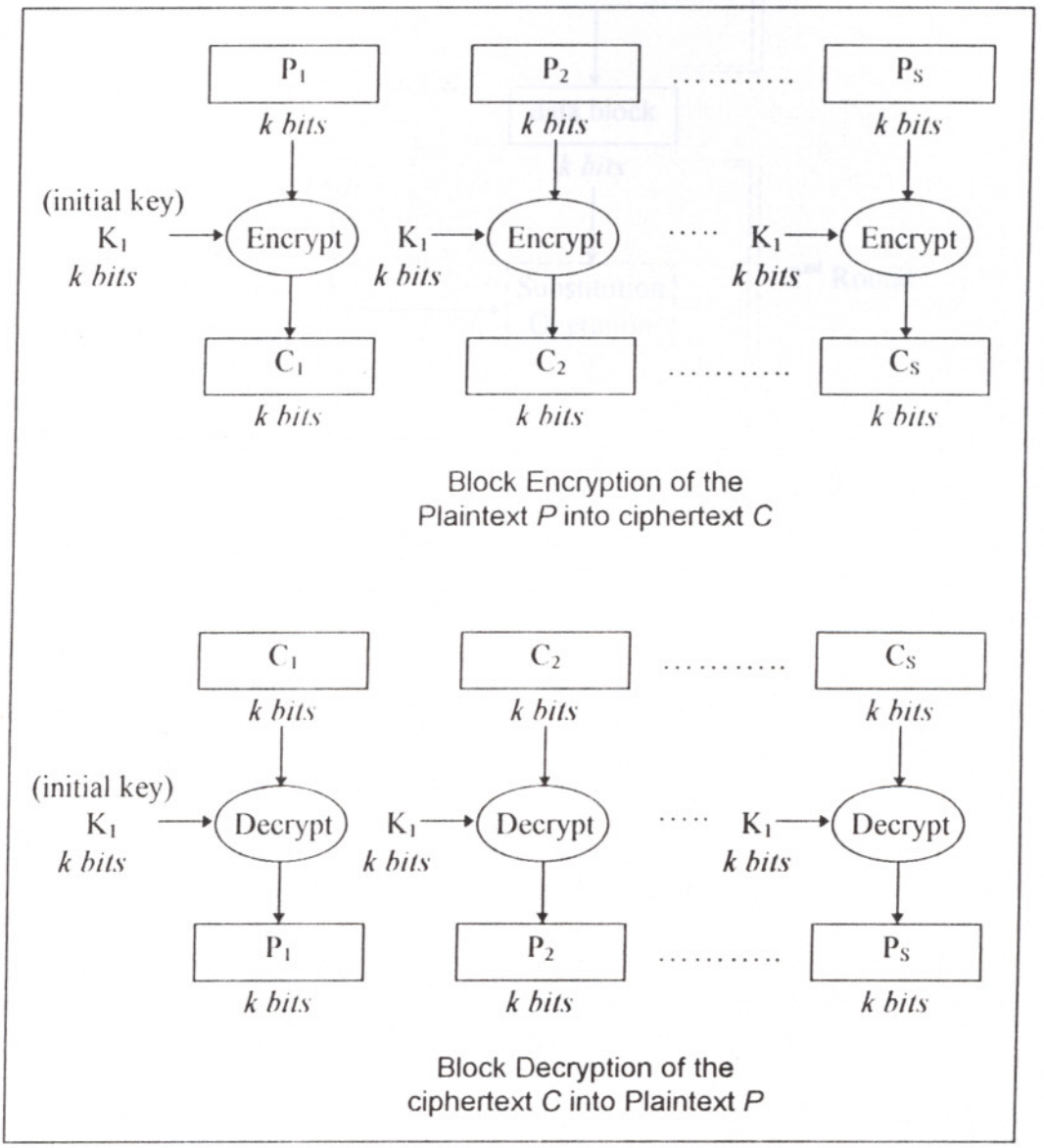


Figure 6.1 The Symmetric Cryptosystem used in the Case Study.

The second part of the design was the construction of the test system model for the comparison of cryptanalytic attacks. Since the study was involved in implementation and analysis of a GA-based cryptanalytic attack, one of the attack types was necessarily this GA-based cryptanalytic model itself. I chose the brute-force attack with a known plaintext / ciphertext pair as the other model in testing and making comparisons.

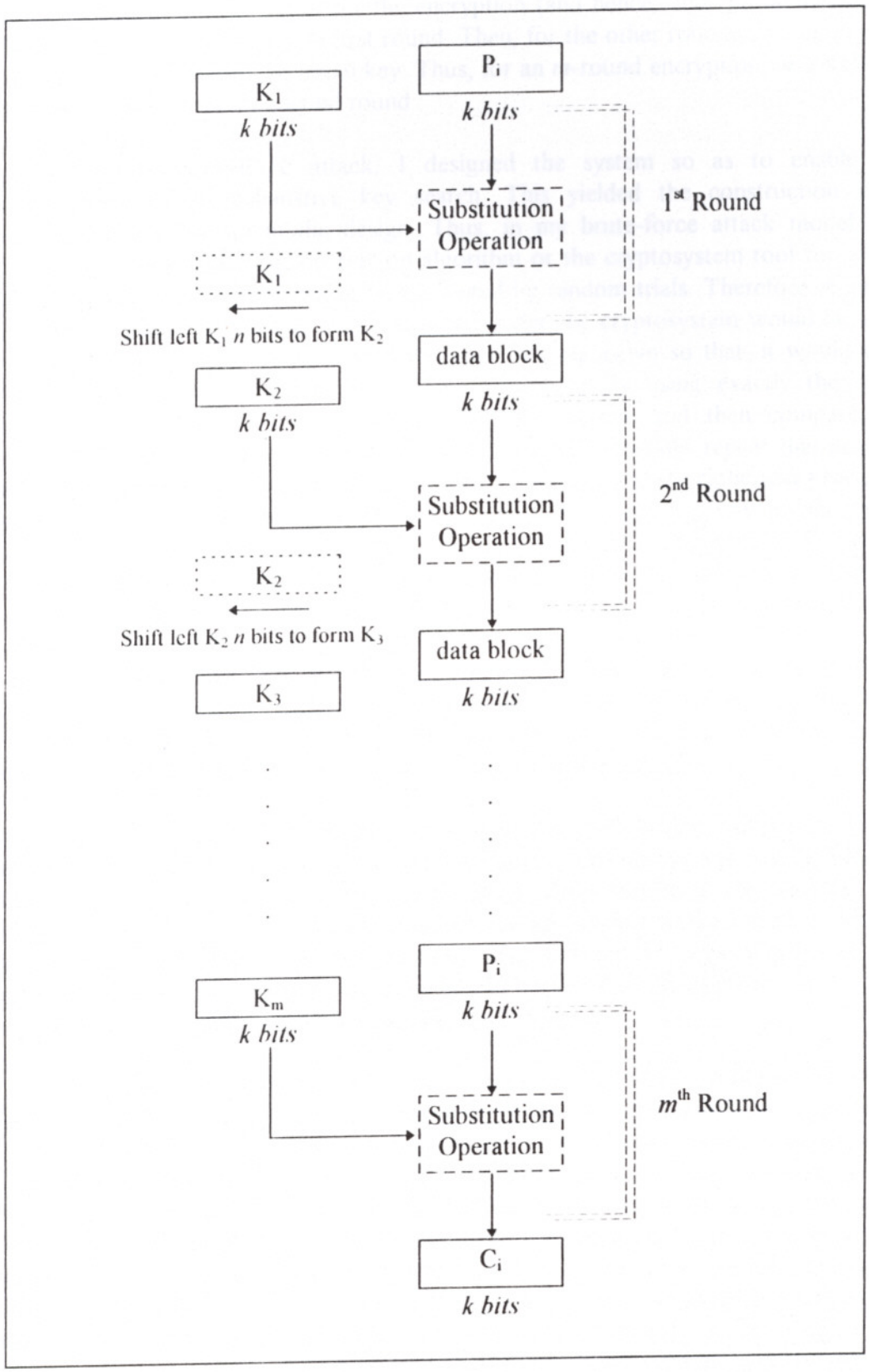


Figure 6.2 The inner structure of the encryption operation for the cryptosystem model.

It should be noted that for the encryption (and hence, decryption) algorithm, the initial key K_1 is used for the first round. Then, for the other rounds, the subkeys K_i are generated by shifting the initial key. Thus, for an *m*-round encryption, *m-1* subkeys are used starting from the second round.

For the brute-force attack, I designed the system so as to enable the establishment of an exhaustive key search. This yielded the construction of a tamperproof mechanism in the design. Thus, in my brute-force attack model, the cryptanalyst could reach the encryption algorithm or the cryptosystem tool for which the keys used in encryption might be recovered by random trials. Therefore, a single plaintext / ciphertext pair that was encrypted under the cryptosystem would be used with a brute-force attack algorithm. I designed this algorithm so that; it would take that plaintext, encipher under any random key value by using exactly the same encryption algorithm that was designed for this system, and then compare the produced ciphertext with the original one. The algorithm would repeat this process until the correct key value is found (the original ciphertext and the ciphertext produced from the attack exactly match each other). This attack model surely insists on an exhaustive *k-bit* key search with a theoretical computational complexity of 2^{k-1} up to 2^k. This is the most important reason why I chose the exhaustive key search attack in comparison with a GA-based attack besides the feasibility, simplicity and accuracy norms. This attack technique typically gives the threshold value for the performance of any alternative cryptanalytic attack model. If the GA-based attack model performs worse than the average performance of exhaustive search for any number of trials with random keys, then this will show the inapplicability and inefficiency of GA' s in the cryptanalysis of the cipher that was designed for this case study.

The GA-based cryptanalytic attack was designed in the manner so as to enhance the brute-force attack technique. The logic in this design was to establish the same brute-force attack, but this time, rather than searching the correct key exhaustively, GA' s would be used to deduce the key exploiting the optimum finding and search capabilities of such algorithms. If a proper GA was adapted in the cryptanalytic attack, the sought key value would be found more efficiently and in less computational time than randomly searching the key-space.

I designed the GA-based attack as follows: As similar to the brute-force attack, the same plaintext / ciphertext pair is input to the system. This plaintext / ciphertext pair would be derived by tamperproofing the same cryptosystem version used for the brute-force attack tests. Then, using a population composed of candidate key values, the ciphertexts would be produced within each generation by using the same encryption algorithm of the designed cryptosystem. Comparing these ciphertexts with the input ciphertext value, the key values that give the fittest results would be included for the next generation. With the aid of GA' s exploitation and exploration strength, after some generations, the possible key values are presumed to converge to the correct key value. The GA is to be executed until the global optimum, in other words, the correct initial encipherment key is found exactly.

Since the cryptosystem and the cryptanalysis is involved in operating on bit values of the data, I preferred to use bit values for the genotypes in the GA. Each *k-bit* chromosome represented the candidate *k-bit* key values for the correct solution. Due

to the nature of the GA' s, I only designed the general structure of the GA-based attack. However, which crossover or mutation techniques with what probability, which selection mechanisms, etc. had to be clarified during the implementation and the trials. But, I designed the system as much flexible and adaptive as possible. The GA used in the cryptanalytic attack was designed so as to benefit from both crossover and mutation.

The most essential issue and the core of the design for the GA-based attack was to find a suitable way to construct a fitness measure and function. Since the aim of the GA was to find the correct key value, then the relation between each key value and the corresponding ciphertext' s correctness would directly impose the fitness of that chromosome. Thus, the fitness measure had to be devised so as to establish a relation between the candidate key value and its resultant ciphertext data. The more fit chromosomes should have to be the ones which could give the more number of correct bits for the required ciphertext. Therefore, the global optimum fitness value would imply the exactly correct ciphertext with no single bit error, which could identify the correct key value being sought. But it should be stressed that not only the total number of correct bits, but also the positions of these bits in the ciphertext are important. For instance, two different key values might produce equal number of correct bits in the ciphertext, however, one of these ciphertexts could be much more near to the correct ciphertext due to the positions of the bits and hence, the ASCII character values of the data. Therefore the fitness function should be designed so as to have a measure on the bit positions of the data as well. The relation between the key value and the ciphertext depends on both the amount of correct bits and the positions of the correct bits. The fitness function should properly find a mapping between the positions of the correct key bits and correct ciphertext bits with their proportional weighted values. This correlation between the bits and the corresponding measure varies due the cryptosystem model (number of rounds, key and block size) and since different variants of the cryptosystem could be used with various number of shift bits and various number of rounds, each fitness function should be designed according to the implementation of the cipher.

For each key value and n-round variant of the cipher, the execution times until breaking the key would be recorded both for the brute-force attack with exhaustive key search and GA-based key search attack. After several trials for different keys, an analysis could be made for the breaking performance and the average time complexities of both attack types for each cipher variant by comparing the results. Thus, the aim of the design criteria could be fulfilled.

6.2 Implementation of the Model

The implementation of the model was straightforward. I first established the required algorithms for the cryptosystem, brute-force attacks and attacks using GA' s, then implemented these algorithms in software and executed them using C language. After the executions, the corresponding results were recorded and analyzed.

For the implementation of the cryptosystem, I chose the key and data block length as *32* bits. After several trials with different key sizes, this value was observed to be legitimate and fairly sufficient because the exhaustive search for higher key

lengths were extremely time consuming which could turn the case study infeasible. Since the data block length was designed to be the same of key length, the plaintext and ciphertext data were also processed in 32-bit blocks. In the algorithms, the plaintext is first divided into 32-bit blocks and each block is encrypted one by one in a batch fashion. This is the simplest and most straightforward implementation of any ECB mode encryption via a sequential algorithm.

For the test studies, I decided to use two different variants of the designed cryptosystem; one with a 3-round and one with a 5-round model. Thus, I first implemented the cryptosystems for both versions by deriving the necessary algorithms and the coding. I made several tests on both cipher variants by using some sample random plaintexts and random key values. I first encrypted a plaintext and then decrypted it using the corresponding ciphertext. For both cryptosystems, all the data were encrypted and decrypted correctly without any errors. As described in the design section, the cryptosystem for any rounds was proposed to be symmetric by the structure of the algorithm. In the implementation, I verified this assertion, because both the encryption and decryption algorithms were exactly the same except their input data. Thus, using the same initial key and the same algorithm, any enciphered plaintext could be decrypted to its original by the symmetric structure of the cryptosystem.

In fact, this symmetric property was due to the substitution operation that I chose and used within each round. I applied the XOR operation for the substitution operation. Thus, for each round both in encryption and decryption, the data is XORed with the key for that round so as to produce the output for the next round. Since DES and DES-like cryptosystems are all designed to use XOR operations in several parts of the algorithm, I also followed the same approach in the implementation. Bits from *0* to *31* of the 32-bit data block is XORed with corresponding 32 bits of the key directly, and no additional mechanism or operation was used in the substitution in order to make the cipher as simple as possible.

For the 3-round cipher, the initial encryption key was used in the first round. Then the second key for the 2^{nd} round was formed by shifting the initial key left *2* bits. And for the last round, the key in the 2^{nd} round (thus, the first subkey) was shifted *4* bits to the left so as to derive the subkey. The key-shifting bit values could have been some other value, but I heuristically chose these in the implementation. Similarly, for the 5-round cipher variant the left key-shifting bit values for the rounds 2 to 5 were chosen *2, 4, 4, 2*, respectively so as to derive the subkeys. Also, for the decryption process, the same key-shift values were used for each round of the each cipher variant.

The encryption algorithms for the 3-round and 5-round variants are denoted as follows;

Encryption algorithm for the 3-round version:

```
Begin ;
        Generate a 32-bit random initial key K1 ;
        K2 = shift_left ( K1 , 2 bits ) ;
        K3 = shift_left ( K2 , 4 bits ) ;
        Set ciphertext C = null ;
```

```
        Get any random plaintext P ;
        Divide P into 32-bit blocks: Pi , where i = 1,2...,s ;
        i = 1 ;
        while ( i <= s ) do {
              Di = ( Pi XOR K1 ) ;
              Di = ( Di XOR K2 ) ;
              Ci = ( Di XOR K3 ) ;
              Concatenate Ci at the end of C ;
              i = i + 1 ;
        } loop ;
End ;
```

Encryption algorithm for the 5-round version:

```
Begin ;
        Generate a 32-bit random initial key K1 ;
        K2 = shift_left ( K1 , 2 bits ) ;
        K3 = shift_left ( K2 , 4 bits ) ;
        K4 = shift_left ( K3 , 4 bits ) ;
        K5 = shift_left ( K4 , 2 bits ) ;
        Set ciphertext C = null ;
        Get any random plaintext P ;
        Divide P into 32-bit blocks: Pi , where i = 1,2...,s ;
        i = 1 ;
        while ( i <= s ) do {
              Di = ( Pi XOR K1 ) ;
              Di = ( Di XOR K2 ) ;
              Di = ( Di XOR K3 ) ;
              Di = ( Di XOR K4 ) ;
              Ci = ( Di XOR K5 ) ;
              Concatenate Ci at the end of C ;
              i = i + 1 ;
        } loop ;
End ;
```

For the implementation of the cryptanalysis phase of the study, I first derived the algorithms and made the studies concerning the standard brute-force attack with the exhaustive key search using a single known plaintext / ciphertext pair. In fact, the implementation of the attack was composed of two parts. In the first part, I chose any random plaintext and for different random key values, the corresponding ciphertexts were enciphered and stored. Then, in the second part of the attack, for each plaintext / ciphertext pair, the attack was established using the same encryption algorithm. Each key value was tested until the correct ciphertext could be reached within the same plaintext. All the correct key values and the computation times were recorded within each attack. I first applied this attack to the 3-round cipher with several random key values, and then similarly to the 5-round version with the same initial key values. The implemented algorithm of the brute-force attack for both variants of the test cipher can be denoted as follows;

Brute-force attack algorithm - Part 1:

```
Begin ;
        Get any random plaintext P ;
        Encrypt P under an unknown key K ;
        Retrieve corresponding ciphertext C ;
End ;
```

Brute-force attack algorithm - Part 2:

```
Begin ;
  Extract 32-bit portion of P as PB ;
  Extract 32-bit portion of C as CB ;
  loop {
        Generate 32-bit random key k ;
        If ( k is not a pre-generated value ) then
          Encrypt PB under k and retrieve corresponding ciphertext block CB_X ;
        end if ;
  } Until ( CB_X = CB ) ;       /* the key is found */
   Record k ;
End ;
```

The key k that satisfies $C_{B\_X} = C_B$ will be exactly equal to the searched key K, hence the initial encryption key will be broken. It should be noted that since the cryptosystem in this study handles each 32-bit data block independently (ECB mode), I dealt with only the first 32-bit block of the plaintext and the ciphertext. However, in the second part of the algorithm, for each randomly generated key value the storage of each previously generated keys and a look-up process is required. In order to eliminate this overhead and simplify the search process, I revised the second part of the algorithm as follows and used this scheme in all of the implementations;

Brute-force attack algorithm - Part 2 (revised version):

```
Begin ;
  Extract 32-bit portion of P as PB ;
  Extract 32-bit portion of C as CB ;
  k = 0 ;
  loop {
        Encrypt PB under k and retrieve corresponding ciphertext block CB_X ;
        If ( CB_X = CB ) then          /* the key is found */
          Record k ;
          exit loop ;
        end if ;
        k = k + 1 ;
  } Until ( k > 2^32 ) ;
End;
```

This revised version also exhaustively searches the key, but rather than randomly generating k values, it starts from an initial key value and tests each key one by one.

For the key search attack using GA' s, I first constructed the genetic algorithm with the necessary criteria, then applied the algorithm to the cryptanalytic attacks of both cipher variants for 3 and 5 rounds. Since, the cryptosystem was implemented with an encipherment key length of 32 bits, the chromosome size was set to 32 in all GA implementations. Since, the correct key is searched in the attacks, the candidate key values would be the possible solutions to the problem, thus the chromosomes would represent the key values. This also suggested the bit-wise representation for the chromosomes, as similar to the structure of the key. Thus, each chromosome is made up of 32-bit strings, where each bit of the chromosome is a gene either having value 0 or 1 and consequently, the genotype of the chromosome directly gives the 32-bit key value itself.

For both attacks, I set the population size to *100*, which could be a generic parameter for any variant of this cryptosystem whenever the key size is held 32-bits. Recalling from the Schemata Theorem, a population size of 100 yields a key search space of 100^3 for each generation. Since the key size, hence the chromosome size is 32, the total key-space is 2^{32}, or approximately 10^{10}. Thus, it could be stated that a key search among 10^4 chromosomes within each generation would be sufficient for the GA, which was also stated in the *Bagnall*' s study, in which the chromosome size was again 32 and the population size was set to 100.[1] On the other hand, for the initial population, all the candidate key values were generated randomly, thus a total of 100 individuals having different key values was gathered. For instance, any individual of the population would be a chromosome of 32-bit array as;

[0 0 1 1 0 1 . . . . . 0 1]

where, the left-most bit would stand for bit-0 of the key, the next one would be bit-1 of the key, and so on. The 32nd bit of the chromosome would correspond to the 32nd bit of the key, or the right-most key bit, bit-31.

Before porting the actual attack to the both cipher variants, I made some tests so as to decide upon which GA mechanisms should be used and how they would be used. I chose some specific 32-bit integer values, and generated some different genetic algorithms to find these values via alternative reproduction, selection operators. This was necessary due to the structure of GA' s. Since, GA is an implementation-based technique rather than a theoretical system, the efficiency and performance of its operators vary among different problems. So, in order to decide upon the most adaptive and best-suited GA for the cryptanalytic attack, I made several tests. After these tests, some mechanisms were proven to be better than others, so I used these in the GA-based brute-force attacks.

For both of the attacks to the 3 and 5-round cipher variants, the single-point crossover was used in the GA' s. However, I used a variable crossover rate which was between 0.7 and 1. Thus, within each generation, two individuals were selected to undergo crossover with a probability of $0.7 \leq p_c \leq 1$. A random value was generated to

[1] A. J. Bagnall, "The Applications of Genetic Algorithms in Cryptanalysis", MS Thesis, University of East Anglia, p. 124, 131, 1996.

set the crossover probability for each chromosome duo. Then, another random number was generated, and if this number was equivalent to or below this threshold p_c, the crossover was applied. The similar mechanism was used for the mutation operator, where a mutation was applied to a chromosome within a probability of $0.2 \leq p_m \leq 0.5$. If a random value was less or equal to p_m, then the mutation was applied to a randomly selected bit of the chromosome. However, I also generated an additional technique for the mutation mechanism. The mutation was applied to not a single bit for each chromosome, but instead it could be applied to n different bits of each selected chromosome. This n value was a random value which was dependent on the p_m in an iterative manner. In other words, the mutations were applied on the same chromosome until a random value below the new threshold p_m was generated. Then, it was passed to the following chromosome for the mutation. It should be stressed that a control mechanism was included so as not to let any bit of a chromosome to be re-mutated.

However, there was an additional mechanism used in the reproduction. In fact, this mechanism was a specially designed operator for this case study which I named as *shift-bits recombination*. This was a very important point in the implementation of GA' s for the key search attacks, since the fitness function was not efficient alone. For the 3-round cipher, the shift-bits recombination operator processes as follows: Before the GA begins the cycles of generation, 2^6 different chromosome values are generated. These chromosomes are formed by changing the *left-most* 6 bits for all the 64 combinations while keeping the other 26 bits set to a constant random value. For each of these key values, the corresponding fitness values are calculated and the chromosomes are ordered with respect to their fitness values. The left-most 6 bits of the eight of the fittest different chromosomes are then recorded. These eight 6-bit strings will be used then within each reproduction. After the crossover and mutation operations are applied to the population, the fittest four different offsprings are selected. Then each of these 6-bit strings are replaced with the left-most 6 bits of each of the four chromosomes so as to form 32 additional chromosomes. This operation is defined as the shift-bits recombination, since the left-most 6 bits are the total key bits shifted left during each encryption process of the 3-round cipher. These additional chromosomes are also input to the new population which will be processed through the replacement operation afterwards. This recombination operation is repeated within each generation, and it should be remarked that the fitness values of these 32 additional chromosomes are also checked out in every stage. I adapted this mechanism for the 5-round cipher model as well. This time, the left-most 12 bits were used to generate 2^{12} chromosomes and the best 16 strings were recorded where the shift-bits recombination was applied to these within the fittest four chromosomes so as to generate 64 additional offsprings. The purpose in the use of shift-bits recombination was to check, manage and include the effect of some specific bits that were shifted in each cycle of the encipherment algorithm; thus the exploitation strength of the GA could be improved significantly. This was necessary due to the difficulty of establishing a proper fitness function which could impact the correlation and dependency of the shifted key bits among the ciphertext.

For the fitness function of the GA' s in both cipher attacks, a proper and effective implementation was necessary so as to estimate and represent the relation between the correct bits of the key and the corresponding ciphertext. I first implemented the function as a raw fitness function which only looked for the total

number of correct bits in the ciphertext. So, each chromosome value is processed through the encryption algorithm of the attacked cipher model, and then the produced ciphertext is compared with the ciphertext enciphered with the unknown key. This can be denoted as follows, where c_i stands for the chromosome representing the candidate key value, and X_j corresponds to the j^{th} bit of the ciphertext enciphered by that chromosome, and T_j is the j^{th} bit of the initial ciphertext that was enciphered by the sought key value;

$$f(c_i) = \frac{\sum_{j=0}^{31}(R_j)}{32} \quad \{R_j = 1 \text{ only if } X_j = T_j, \text{ else } R_j = 0; \text{ for } 0 \le j \le 31\}$$

Thus, if all of the bits of the ciphertext X are equal to the original ciphertext T, then that chromosome has fitness equivalent to 1, which would be the global optimum. In other words, a chromosome having a raw fitness of 1 implies the exact unknown key. The more fit chromosomes having fitness values around 1 would be passed onto the next generations so as to converge to the genotype having the raw fitness as 1.

However, this raw fitness measure only focused on the correctness of the ciphertext which could not exploit the correct bit positions of the key, ciphertext, the correlation between the key bits, or the ciphertext bits, or the dependency or interrelationship between the key bits and the ciphertext bits. For instance, two different key values with different number of correct key bits could give the same or near fitness values. (ie, a key with a total of 31 correct bits and another one with 25 correct bits could both have a fitness of 26/32 = 0.8125.) These both chromosomes would be equally treated in the GA due to the raw fitness function, although one of them is more converged towards the correct key value. Indeed, in some cases, the candidate chromosomes would be equivalent in the total number of correct bits while their fitness measure do not say so. (eg, a chromosome only had a single wrong bit for the bit-5 position of the key (the 6^{th} bit from the left), but its fitness value was computed as 22/32 = 0.6875; yet another chromosome had a single bit error at the bit-15 position of the key, but provided a fitness value of 27/32 = 0.84375. I observed this type of diversity especially whenever any one or more bits of the left-most 6 bit positions of the key were incorrect.)

All these were due to the avalanche effect and the substitution complexity of the cipher. In my observations, a noticeable fact was that; for the 3-round cipher, a single bit error in the left-most 2 bits (bit-0 and bit-1) of the key affected the first and fourth blocks of the ciphertext for each 32-block portion. Furthermore, any single bit error for the bit-positions 2 to 5 from the left provided errors in all of the four blocks of the ciphertext. In fact this was a reasonable fact, because in the 3-round encryption algorithm, the left-most 2 bits were shifted in the second round to develop the subkey and the following 4 bits of the subkey were shifted left in the next round. Thus, these left-most 6 bits of the key were more disruptive and effective on the ciphertext blocks than any other bits of the key. The same fact was also observed in the 5-round cipher, where the single bit errors in the left-most 2 bits affected the first and fourth blocks of the ciphertext data and any single bit error among the other 10 bits used in the shifting affected all the block of the ciphertext. Further analyzes could possibly find correlations and internal dependencies between the specific positions of the key and the

ciphertext such that; when several bits in the specific positions of the key were incorrect, which bits of the ciphertext would be affected, and so on.

However, these impose very intensive and extended studies of each of the cipher variant, requiring statistical cryptanalysis and analyzes. Unless such studies were established, the derivation of a new fitness function rather than the raw one would be meaningless. But since the case study was designed so as to focus on brute-force attack types, I used the raw fitness function as the general fitness function in all tests. On the other hand, the effect of left-most 6 and 12 key bits in both ciphers were obvious, so I decided to exploit this aspect in the GA implementation. It was this reason that I derived the shift-bits recombination operator which could reflect the shifted key bits' impact on the ciphertext and reverse the negative effect of these bits in the fitness measure.

The roulette-wheel selection mechanism was used in order to select which chromosomes would undergo crossover and mutation. The standard method of roulette-wheel that was explained in Chapter 4, section 4.3.2 was used straightforwardly.

The replacement and the generation of the new population was achieved by replacing the weaker chromosomes with the newly generated offsprings. The fittest ones were ordered among the offsprings and these were replaced with the worst ones from the current population. I devised this mechanism so that for any generation throughout the execution of the GA, the population size was strictly held constant being equivalent to the initial value, 100.

The complete implementation of the exhaustive key search attack using GA's can be denoted by the algorithms as follows;

GA-based attack for the 3-round cipher:

```
Proc init_encrypt ( ) ;
Begin ;
        Get any random plaintext P ;
        Encrypt P under an unknown key K ;
        Retrieve corresponding ciphertext C ;
End ;

Proc main ( ) ;
Begin ;
   Extract 32-bit portion of P as P_B ;
   Extract 32-bit portion of C as C_B ;
   Initialize a random population of size 100 having each chromosome as a 32-bit array ;
   Generate 64 specific chromosomes sc_i for the shift-bits recombination ;
   for ( i = 0 to 63 ) {
      Proc Fitness ( sc_i ) ;
   } next i ;
   Select the fittest 8 sc_i and store their left-most 6-bits in an array SA[s_left_i] ;
```

```
while ( True ) do {        /* iterative loop until global optimum is reached */
    for ( i = 0 to 99 ) {
        Proc Fitness ( c_i ) ;
    } next i ;
    Roulette-wheel selection to select n chromosomes as the more fit c_i's ;
    for ( i = 0 to n ) {
        Apply crossover among c_i's ;
        Apply mutation among c_i's ;
        Proc Fitness ( c_i ) ;
        Add c_i to offspring population as o_i;
    } next i ;
    Select the fittest 4 offsprings o_i ;
    for ( i = 0 to 7 ) {
        for ( j = 0 to 3 ) {
            Apply shift-bits recombination among SA[s_left_i] and o_i to form o_ij ;
            Proc Fitness ( o_ij ) ;
        } next j ;
    } next i ;
    Insert o_ij's to the offspring population ;
    Reorder the offspring population ;
    Replace the weaker individuals of the old population with the more fit offsprings
    and get new population for which Σ c_i = 100 ;
} loop;
End ;

Proc Fitness ( parameter x ) ;
Begin ;
    Encrypt P_B under x and retrieve corresponding ciphertext block C_B_X ;
    If ( fitness f(x) = 1 ) then
        Record x ;          /* the key is found */
        exit main ;
    else
        Store f(x) in an array FA[x, f(x)] ;
    end if ;
End ;
```

The GA-based attack for the 5-round variant was similar to the 3-round version except some parameters. For instance, the generation of string-portions which would be used for the shift-bits recombination was revised as;

```
Generate 4096 specific chromosomes sc_i for the shift-bits recombination ;
for ( i = 0 to 4095 ) {
    Proc Fitness ( sc_i ) ;
} next i ;
Select the fittest 16 sc_i and store their left-most 6-bits in an array SA[s_left_i] ;
```

Consequently, the shift-bits recombination operation was also revised in the GA, which was re-implemented as;

```
for ( i = 0 to 15 ) {
    for ( j = 0 to 3 ) {
        Apply shift-bits recombination among SA[s_left_i] and o_j to form o_ij ;
        Proc Fitness ( o_ij ) ;
    } next j ;
} next i ;
```

6.3 Results and Discussion

Several cryptanalytic tests were made within the 3-round and 5-round versions of the symmetric block cipher. For each cipher variant, 20 different random key values were used to generate the 20 different ciphertexts corresponding to a single known plaintext. Then, the brute-force attack with exhaustive key search and GA-based key search attacks were applied to find each of the 20 keys. All the tests, implementations and attacks were carried out on a single *Intel Pentium-MMX 233 Mhz* PC platform with *128* Mbytes of RAM. The execution times for each attack within each key value (denoted in hexadecimal) are given in the tables 6.1 and 6.2. The total time spent for breaking the 20 keys and the corresponding average execution times for both methods and ciphers are also given in the Table 6.3.

It could be seen from all these results that GA-based key search attack outperformed exhaustive key search for both cipher models, when the overall performance and the average key breaking times are considered. In some of the cases, for both cipher models, the exhaustive key search seemed to find the key faster than its GA rival. However, since the key values are random and the search is started from the key value 0 and continued sequentially for the exhaustive search, for some key values this might be possible. But the concluding remark can be stated if only the overall performance for all the 20 keys is compared. This clearly and firmly validates the better cryptanalytic performance of GA-based search over exhaustive key search for both cipher models. It could be analyzed from Table 6.3 that GA-based search found the keys nearly three times faster than the exhaustive search for the 3-round cipher, and two times faster than the exhaustive search for the 5-round cipher. Another important point is that, GA-based attack successfully found the exact key value in all of the 40 tests and did not get stuck at any local optima or did not lack in finding the global optimum.

Another remarkable result is that since the 5-round cipher model was more complex than the 3-round version, the resistivity to both attack types was observed to be significantly increasing. However, the performance degradation for the GA-based attack seemed to be much higher than its rival. This was also an expected result, because the increase in the encryption rounds and in the number of shift-bits do have a much greater negative impact on the performance of the GA-based attack. First, the number of strings and their fitness computation used in the initialization of the shift-bits recombination goes up to 4096 from 64; secondly, for each population, instead of 32, 64 fitness computations are made within the shift-bits recombination; and finally, a great number of fitness computations are required for each generation of the GA and consequently each fitness computation involves encipherment and ciphertext check for

Table 6.1 Results achieved by exhaustive key search vs. key search using GA for the 3-round cipher.

Case:	Attack type:	Key:(Hex)	Broken in:
1	Exhaustive key search	0E37FF08	554.04 sec
2	"	00A258F2	28.78 sec
3	"	08C25921	340.26 sec
4	"	6F1CA520	4335.44 sec
5	"	FF47A408	10099.98 sec
6	"	B93A521D	7318.27 sec
7	"	12E007A9	721.61 sec
8	"	D187C134	8261.45 sec
9	"	0970CE10	368.38 sec
10	"	10056A98	625.99 sec
11	"	FFFF1724	10143.48 sec
12	"	F0000000	9514.69 sec
13	"	0E000001	552.77 sec
14	"	0FFFFFFF	631.42 sec
15	"	1F0F47CB	1226.15 sec
16	"	42780000	2664.43 sec
17	"	33E50018	2083.05 sec
18	"	7811BD22	4815.81 sec
19	"	C8885361	8060.26 sec
20	"	DDD28A44	8901.72 sec
1	key search using GA	0E37FF08	3123.36 sec
2	"	00A258F2	2475.10 sec
3	"	08C25921	2602.27 sec
4	"	6F1CA520	2015.01 sec
5	"	FF47A408	1757.06 sec
6	"	B93A521D	2058.67 sec
7	"	12E007A9	1986.44 sec
8	"	D187C134	1689.03 sec
9	"	0970CE10	2906.12 sec
10	"	10056A98	1050.75 sec
11	"	FFFF1724	843.03 sec
12	"	F0000000	95.59 sec
13	"	0E000001	210.31 sec
14	"	0FFFFFFF	104.16 sec
15	"	1F0F47CB	2573.40 sec
16	"	42780000	698.85 sec
17	"	33E50018	1245.93 sec
18	"	7811BD22	2254.61 sec
19	"	C8885361	2003.07 sec
20	"	DDD28A44	957.82 sec

an increased number of rounds of the cipher. The last issue is also valid for the exhaustive key search, but the encryption and ciphertext check is more straightforward and yet the only process in the attack. This lowers the performance of the exhaustive key search for the 5-round cipher, but the effect on the GA-based search is much higher due to the structure of the genetic process.

As previously mentioned in section 6.3, the fitness function for the GA was not excelled and could be possibly improved and turned into a more efficient operator if a more intensive and extended analysis is made for the cipher models. If a suitable model was established for the key and ciphertext bits' dependence; some relationships among the bits of the key, or ciphertext or interrelationship between both were formulated via *Markov Chains*, or other mechanisms; the correlation between the keys and the

Table 6.2 Results achieved by exhaustive key search vs. key search using GA for the 5-round cipher.

Case:	Attack type:	Key:(Hex)	Broken in:
1	Exhaustive key search	0E37FF08	950.16 sec
2	"	00A258F2	41.97 sec
3	"	08C25921	582.93 sec
4	"	6F1CA520	7507.77 sec
5	"	FF47A408	17246.72 sec
6	"	B93A521D	12525.04 sec
7	"	12E007A9	1280.15 sec
8	"	D187C134	14143.59 sec
9	"	0970CE10	635.60 sec
10	"	10056A98	1078.35 sec
11	"	FFFF1724	17335.36 sec
12	"	F0000000	16199.29 sec
13	"	0E000001	942.85 sec
14	"	0FFFFFFF	1077.53 sec
15	"	1F0F47CB	2108.37 sec
16	"	42780000	4488.56 sec
17	"	33E50018	3512.54 sec
18	"	7811BD22	8119.47 sec
19	"	C8885361	13573.74 sec
20	"	DDD28A44	14987.96 sec
1	key search using GA	0E37FF08	2987.55 sec
2	"	00A258F2	3012.26 sec
3	"	08C25921	5561.73 sec
4	"	6F1CA520	2678.06 sec
5	"	FF47A408	8340.14 sec
6	"	B93A521D	5108.69 sec
7	"	12E007A9	6219.02 sec
8	"	D187C134	2465.41 sec
9	"	0970CE10	3549.50 sec
10	"	10056A98	4006.14 sec
11	"	FFFF1724	2358.61 sec
12	"	F0000000	1034.13 sec
13	"	0E000001	1567.05 sec
14	"	0FFFFFFF	943.16 sec
15	"	1F0F47CB	3678.90 sec
16	"	42780000	1257.22 sec
17	"	33E50018	6431.43 sec
18	"	7811BD22	5219.36 sec
19	"	C8885361	3573.81 sec
20	"	DDD28A44	1904.67 sec

Table 6.3 Total and average execution times for the 20 test keys within both attack types applied to 3 and 5-round cryptosystem variants.

Attacked cryptosystem	**3-round cipher**		**5-round cipher**	
Type of cryptanalysis	**Exhaustive key search**	**key search using GA**	**Exhaustive key search**	**key search using GA**
Total execution time	81247.98 sec.	32650.58 sec.	138337.95 sec.	71896.84 sec.
Average execution time	4062.399 sec.	1632.529 sec.	6916.8975 sec.	3594.842 sec.

ciphertexts was properly measured, then the GA could have been hopefully much more efficient than the one used in this study. All these necessary studies are referred as statistical cryptanalysis methods. For any cipher model, if either of the following criteria holds;

- some set of output bit (ciphertext) coordinates, C_S, are dependent on some set of input bit (plaintext) coordinates, P_S, for a fixed key k,
- some set of output bit (ciphertext) coordinates, C_S, are dependent on some set of key bit coordinates, k_S, for a fixed input, P_S,

then a statistical cryptanalytic attack can be carried out for that cryptosystem.[2] In other words, if a dependence between some subset of the output bit positions and key bit or input bit positions exists for any cryptosystem, then a statistical attack might be possible. The second criteria was observed for the cryptosystem designed in this case study. In fact, both attack types were applied following the basic logic in the second criteria. However, it should be stressed that for any strong symmetric cryptosystems such as DES, the statistical cryptanalysis attack is proven to be infeasible, because no such obvious or direct dependencies were found.[3] Thus, any GA designed so as to enhance the exhaustive key search of DES or to exploit direct statistical interferences would be useless.

On the other hand, for weak symmetric block ciphers or polyalphabetic cryptosystems, any GA-based cryptanalytic attack with a proper fitness function or any genetic operator exploiting the direct dependencies between data and key bits could be considerably successful and efficient. This was proven to be true both for this case study and other studies mentioned in Chapter 5, section 5.1. However, the GA in this study was adapted to benefit from the dependencies partially, and a complex fitness function that has a measure on any correlation or dependencies was not used. This is due to the fact that any detailed statistical analysis or statistical cryptanalysis was beyond the scope of this case study.

As a final statement, I should remark that if any GA is being used to enhance the exhaustive key search or to improve any cryptanalytic attack based on searching the key-space; then the building-block hypothesis corresponds to the dependency between (sub)keys and the related ciphertexts, but in a converse manner. In the building-block hypothesis, it was stated that low order schemata with above average fitness values combine to form optimum whereas in the cryptosystems such as the one in this case study, the dependency (if could be observed) between the key and the ciphertext yields to construct high order schemata with above average fitness. This must be strictly taken into consideration whenever a genetic algorithm is applied to a cryptanalytic attack searching the key(s). I happened to notice this fact in this study and it was also stated in some other researches as well.[4] Henceforth, the success of a genetic algorithmic approach in any type of cryptanalytic attack lies in the proper representation, the achievement of best suitable genetic operators and mechanisms and most crucially, the establishment of a well-adapted and efficient fitness function.

---

[2] A. J. Bagnall, "The Applications of Genetic Algorithms in Cryptanalysis", MS Thesis, University of East Anglia, pp. 92-95, 1996.
[3] *ibid*, p. 95.
[4] *ibid*, pp. 121-122, 140.

Chapter 7

CONCLUSIONS

In this study, it' s shown that genetic algorithms could be successful in the cryptanalysis of simple symmetric block cryptosystems where dependencies between key bits and the data can be observed and their algorithms possess security flaws that enable statistical attacks. On the other hand, for DES and DES-like ciphers, any genetic algorithm would be inefficient for brute-force type cryptanalytic attacks such as searching the key-space due to the lack of dependence between instances of key and plaintext / ciphertext data among such cryptosystems.

It' s also analyzed that genetic algorithms might be successful in the enhancement of the differential or linear cryptanalytic attacks among DES and DES-like cryptosystems. The basic methodology in these cryptanalytic attacks are believed to have a convenient nature to enable the adaptation of genetic algorithmic approaches. (*A. J. Bagnall* had made a similar statement in one of his studies as well.[1]) If, a suitable implementation with the necessary selection, reproduction, fitness function and other mechanisms of the genetic algorithm is established, then the computational complexity and the data requirements in these types of attacks can be decreased significantly to some level. This might result with a much more powerful differential / linear cryptanalytic attack that consequently turns these attacks to an applicable and more feasible in real-life implementations. Thus, the use of genetic algorithms in differential and linear cryptanalysis could outdate the security of DES and DES-like ciphers entirely providing a more realistic, powerful and applicable threat.

The key to the success of genetic algorithmic approaches in differential / linear cryptanalysis is the derivation of best suitable genetic operators and mechanisms and the establishment of a well-adapted and efficient fitness function within the implementation. It should not be forgotten that any genetic algorithm used in differential or linear cryptanalysis do not entirely alter the ordinary methodology in the attack and stand alone as a competitive alternative. Instead, the genetic algorithm should be embedded as an enhancement tool in several phases of such attacks whenever applicable.

[1] A. J. Bagnall, "The Applications of Genetic Algorithms in Cryptanalysis", MS Thesis, University of East Anglia, p. 141, 1996.

SUMMARY

In this study, several researches and analyzes have been achieved involving differential / linear cryptanalysis with genetic algorithms and related hypothetical models are constructed, assumptions and possible outcomes of such models are discussed theoretically. Remarks and recommendations for any successful implementation of genetic algorithmic approaches to differential or linear cryptanalytic attacks are given as a guide for future studies.

Also, a brief perspective and detailed information about the main topics in cryptanalysis, cryptography, symmetric block cryptosystems, differential and linear cryptanalysis, and genetic algorithms are given. Regarding as the basic model of symmetric block ciphers, DES has been studied and discussed in detail. Several observations and experiments related with DES have also been included in this study.

A case study of a simple block cipher' s cryptanalysis and the usage of genetic algorithms in brute-force attacks among such a cryptosystem have been established. How genetic algorithm techniques could be embedded into cryptanalytic attacks or how genetic algorithms might be implemented for the usage of cryptanalysis in such simple ciphers have been discussed.

Implementations, tests, and results providing any precise conclusions with the approval or disapproval of; genetic algorithms' usage in differential / linear cryptanalytic attacks against DES and DES-like ciphers have been beyond the scope of this study due to necessary shortcomings which were mentioned in several chapters of this thesis.

ÖZET

Bu çalışmada, genetik algoritmalarla diferansiyel / lineer kriptanaliz konusuna ilişkin çeşitli araştırmalar ve analizler yapılmış, bağlantılı varsayımsal modeller oluşturulmuş, önergeler ve bu tür modellerden ortaya çıkabilecek olası sonuçlar teorik bir çerçeve içinde tartışılmıştır. İlerideki çalışmalarda, diferansiyel ya da lineer kriptanalitik saldırılara genetik algoritmalar yaklaşımıyla yapılabilecek her türlü başarılı uyarlamaya ışık tutması amacıyla bazı uyarılar ve tavsiyelerde bulunulmuştur.

Ayrıca, kriptanaliz, kriptografi, simetrik blok kriptosistemler, diferansiyel ve lineer kriptanaliz ile genetik algoritmalardaki belli başlı konular ve kavramlara ilişkin ayrıntılı bilgiler verilmiştir. Simetrik blok şifrelerin temel modeli olarak görüldüğünden DES ile ilgili çeşitli gözlemler yapılmış ve DES şifre sistemi ayrıntılı bir şekilde incelenmiştir.

Basit bir şifre sisteminin kriptanalizi ve bu sisteme yapılan brute-force tipi kriptanalitik saldırılarda genetik algoritmaların kullanımına ilişkin örnek bir çalışma yapılmıştır. Bu tür basit şifre sistemlerinin kriptanalizinde genetik algoritmaların nasıl kullanılabileceği ve kriptanalitik saldırılara nasıl uyarlanabileceği konularında çeşitli araştırmalar yapılmış ve gözlemlerde bulunulmuştur.

Daha önceki bölümlerde açıklanan çeşitli nedenler ve teknik yetersizliklerden ötürü, DES ve benzeri şifre sistemlerinin diferansiyel veya lineer kriptanalizinde genetik algoritmaların kullanımına ilişkin uygulamalı çalışmalar teze dahil edilmemiştir. Dolayısıyla, bu konuda olumlu ya da olumsuz kesin çıkarımlara gidilebilecek uyarlamalar, testler ve sonuçlar bulunmamaktadır.

BIBLIOGRAPHY

ADFA Computational Intelligence Group Projects, "Applications of Evolutionary Algorithms in Cryptanalysis", Internet Document, http://www.cs.adfa.oz.au/research/CIG/projects.html, April 1999.

AES, Internet Document, http://csrc.nist.gov/encryption/aes/aes_home.htm, 1999.

ALBA, Enrique, COTTA, Carlos, "Introduction to Nature-Inspired Algorithmic Techniques", Internet Document, http://www.lcc.uma.es/personal/cotta/semEC/cap01/cap_1.html, 1997.

ALCOURT, "Differential Cryptanalysis", Internet Document, http://www.execpc.com/~alcourt/desdoc.html, 1998.

BACK, Thomas, FOGEL, David B., MICHALEWICZ, Zbigniew (editors), *Handbook of Evolutionary Computation*, Institute of Physics Publishing, Oxford University Press, 1997.

BAGNALL, A. J., "The Applications of Genetic Algorithms in Cryptanalysis", MS Thesis, University of East Anglia, 1996.

BAGNALL, A. J., MCKEOWN, G. P., RAYWARD-SMITH, V. J., "The Cryptanalysis of a Three Rotor Machine Using a Genetic Algorithm", ICGA97 The Seventh International Conference on Genetic Algorithms, July 19-23, 1997.

BAKHTIARI, Shahram, "Linear Cryptanalysis of DES Cipher", University of Wollongong, The Report for Master of CS Degree, July 1, 1994.

BAKHTIARI, S., SAFAVI-NAINI, R., "Application of PVM to Linear Cryptanalysis", University of Wollongong, Technical Report, July 25, 1994.

BALDWIN, J. Mark, "A New Factor in Evolution", American Naturalist 30, June 1896, *reprinted in:* http://paradigm.soci.brocku.ca/~lward/Baldwin/BALD_002.html, 1998.

BEASLEY, David, BULL, David R., MARTIN, Ralph R., "An Overview of Genetic Algorithms - Part 1", University Computing, UCISA, 1993.

BEASLEY, David, BULL, David R., MARTIN, Ralph R., "An Overview of Genetic Algorithms - Part 2", University Computing, UCISA, 1993.

BELEW, Richard K., BOOKER, Lashon B. (editors), *Proceedings of the 4th International Conference on Genetic Algorithms and their Applications*, San Mateo, CA, Morgan Kaufmann Publishers, USA, 1991.

BIERWIRTH, Christian, KOPFER, Herbert, MATTFELD, Dirk C., RIXEN, Ivo, "Genetic Algorithm Based Scheduling in a Dynamic Manufacturing Environment", Department of Economics, University of Bremen, Germany, Technical Report.

BIHAM, Eli, "Cryptanalysis of Multiple Modes of Operation", Journal of Cryptology, vol.11 Number 1, 1998.

BIHAM, Eli, "On Matsui' s Linear Cryptanalysis", Technion- Israel Institute of Technology, Technical Report, 1994.

BIHAM, Eli, KNUDSEN, Lars R., "DES, Triple-DES and AES", RSA Laboratories' CryptoBytes, vol.4 Number 1, 1998.

BIHAM, Eli, SHAMIR, Adi, "Differential Cryptanalysis of DES-like Cryptosystems", The Weizmann Institute of Science - Department of Applied Mathematics, Research Paper, 1990.

BIHAM, Eli, SHAMIR, Adi, "Differential Cryptanalysis of the full 16-round DES", Technion- Israel Institute of Technology, Technical Report, 1991.

BIHAM, Eli, SHAMIR, Adi, *Differential Cryptanalysis of the Data Encryption Standard*, Springer-Verlag New York, Inc., USA, 1993.

BLICKLE, Tobias, THIELE, Lothar, "A Comparison of Selection Schemes used in Genetic Algorithms", Swiss Federal Institute of Technology, TIK-Report, No:11, December 1995.

BOLE, Leonard, CYTOWSKI, Jerzy, *Search Methods for Artificial Intelligence*, Academic Press Inc., San Diego, CA, 1992.

BROWN, T.A., *Genetics - A Molecular Approach*, Chapman & Hall, UK, 1992.

BUI, Thang Nguyen, MOON, Byung Ro, "Genetic Algorithm and Graph Partitioning", IEEE Trans. On Computers, vol. 45, No. 7, July 1996.

CERTICOM INC., "An Introduction to Information Security", Certicom Whitepaper, March 1997.

CORDIAN, Eric Michael, "The DES Analytic Crack FAQ", Internet Document, http://www.cyberspace.org/~enoch/crakfaq.html, December 1998.

CRYPTOGRAPHY FAQ, Cryptosystems Journal, Internet Document, http://ourworld.compuserve.com/homepages/crypto/cryfaq05.htm, 1998.

CYPHERPUNKS, "Re: Coding and Nnet's", Internet Document, http://www.inet-one.com/ cypherpunks/dir.95.11.08-95.11.14/msg00209.html, May 1998.

DAMGARD, Ivan B., KNUDSEN, Lars R., "Two-Key Triple Encryption", Journal of Cryptology, vol.11 Number 3, 1998.

DAVIDENKO, Vladimir N., KUREICHIK, Victor M., MIAGKIKH, Victor V., "Genetic Algorithm for Restrictive Channel Routing Problem", ICGA97 The Seventh International Conference on Genetic Algorithms, July 19-23, 1997.

DAVIS, Lawrence (editor), *Genetic Algorithms and Simulated Annealing*, Pitman Publishing, Morgan Kaufmann Pub., Inc., 1987.

DE JONG, Kenneth A., SPEARS, William M., "Using Genetic Algorithms to Solve NP-Complete Problems", ICGA-89 The Third International Conference on Genetic Algorithms, June 4-7, 1989.

EMERGENT TECHNOLOGIES INC., "On the Decryption of Rotor Ciphers Using Genetic Algorithms", Internet Document, http://www.sys.uea.ac.uk/~ajb/rotor.html, September 1997.

FERNANDEZ, Jaime, "The Genetic Programming Tutorial Notebook", Internet Document, http://www.geneticprogramming.com/Tutorial/index.html, 1997.

GENETIC ALGORITHMS FAQ, Internet Document, http://www.cs.cmu.edu/Groups/AI/html/faqs/ai/genetic/part2/faq-doc-6.html, 1997.

GOLDBERG, David E., DEB, Kalyanmoy, KARGUPTA, Hillol, HARIK Georges, "Rapid, Accurate Optimization of Difficult Problems Using Fast Messy Genetic Algorithms", University of Illinois, Urbana-Champaign, IlliGAL Report No. 93004, February 1993.

HART, Thomas E., "Lamarck and His Theory of Evolution", Internet Document, http://www.stg.brown.edu/projects/hypertext/landow/victorian/science/lamarck1.html, 1997.

KALISKI, Burt, "Life After DES", RSA Data Security Inc., Internet Document, http://www.rsa.com, 1998.

KELLER, Bill, LUTZ, Rudi, "A New Crossover Operator for Rapid Function Optimisation Using a Genetic Algorithm", Research Paper, School of Cognitive and Computing Sciences, The University of Sussex.

KELSEY, John, SCHNEIER, Bruce, WAGNER, David, "Key-Schedule Cryptanalysis of IDEA, G-DES, GOST, Safer and Triple-DES", Advances in Cryptology--CRYPTO '96 Proceedings, 1996.

KNUDSEN, Lars R., ROBSHAW, Matt, "Non-linear Approximations in Linear Cryptanalysis", Advances in Cryptology - Proc. EUROCRYPT'96, Springer Verlag, USA, 1996.

LUDOVIC, M.E., "Genetic Algorithms, an Alternative Tool for Security Audit Trails Analysis", Internet Document, http://www.supelec-rennes.fr/rennes/si/equipe/lme/these/oakland95/oakland95.html, 1997.

LUGER, George F., STUBBLEFIELD, William A., *Artificial Intelligence Structures and Strategies for Complex Problem Solving - Second Edition,* The Benjamin / Cummings Publishing Company, Inc., 1993.

MATTHEWS, R. A., "The use of genetic algorithms in cryptanalysis", Cryptologia, v. 17, No. 2, April 1993.

MICHALEWICZ, Zbigniew, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, USA, 1992.

MITCHELL, Melanie, FORREST, Stephanie, HOLLAND, John H., "The Royal Road for Genetic Algorithms: Fitness Landscapes and GA Performance", Proceedings of the First European Conference on Artificial Life, Cambridge, 1991.

NELSON, Sigurd A., "Strain Gage Selection in Loads Equations Using a Genetic Algorithm", NASA Contractor Report 4597, 1994.

PFLEEGER, Charles P., *Security in Computing,* P T R Prentice Hall, Inc., Eaglewood Cliffs, New Jersey, USA, 1989.

PUNCH, W. F., GOODMAN, E. D., PEI, Min, CHIA-SHUN, Lai, HOVLAND P., ENBODY, R., "Further Research on Feature Selection and Classification Using Genetic Algorithms", ICGA93, 1993.

RAWLINS, Gregory J.E. (editor), *Foundations of Genetic Algorithms,* Morgan Kaufmann Pub., San Mateo, Calif., 1991.

RITTER, Terry, "2xIsolated DES: Another Weak Two-Level DES Structure", Ritter Software Engineering White Paper, February 16, 1994.

RITTER, Terry, "The Context of the Fenced DES Design", Ritter Software Engineering White Paper, June 30, 1994.

RIVEST, Ronald L., CONTINI, Scott, ROBSHAW, M. J. B., YIN, Yiqun Lisa, "The Security of the RC6 Block Cipher", RSA Laboratories, Technical Report, August 20, 1998.

ROGAWAY, Phillip, "The Security of DESX", RSA Laboratories' CryptoBytes, vol.2, Number 2, 1996.

ROGERS, James L., MCCULLEY, Collin M., "Integrating a Genetic Algorithm into a Knowledge-Based System for Ordering Complex Design Processes", NASA TM-110247, April 1996.

RSA Data Security Inc, "Answers to FAQ About Today's Cryptography", RSA Laboratories paper, 1996.

RSA Data Security Inc., RSA '99 Press Release - Internet Document, http://www.rsa.com/pressbox/html/990119-1.html, 1999.

RUBIN, Aviel D., HONEYMAN, Peter, "Formal Methods for the Analysis of Authentication Protocols", Technical Report, Center for Information Technology Integration, 1993.

RYAN, George W., "A Genetic Search Technique for Identification of Aircraft Departures", NASA Contractor Report 4688, 1995.

SCHNEIER, Bruce, "Why Cryptography is harder than it looks", CounterPlane Systems, Technical Report, 1997.

SCHNEIER, Bruce, *Applied Cryptography - Second Edition*, John Wiley & Sons, Inc., USA, 1996.

SPILLMAN, R., "Cryptanalysis of knapsack ciphers using genetic algorithms", Cryptologia, v. 17, No. 4, October 1993.

SPILLMAN, R., JANSSEN, M., NELSON, B., KEPNER, M., "Use of a genetic algorithm in the cryptanalysis of simple substitution ciphers", Cryptologia, v. 17, No.1, January 1993.

STALLINGS, William, *Network and Internetwork Security - Principles and Practice*, Prentice Hall International Editions, Eaglewood Cliffs, New Jersey, USA, 1995.

WHITLEY, L. Darrell, "A Genetic Algorithm Tutorial", Technical Report, Computer Science Department, Colorado State University, November 10, 1993.

WHITLEY, L. Darrell (editor), *Foundations of Genetic Algorithms 2*, Morgan Kaufmann Pub., San Mateo, Calif., 1993.

APPENDIX A

A.1 Properties of Modular Arithmetic

If $a \bmod n = b$ then $a = c*n + b$, for some integer c

For any two integers x and y: $x \equiv y$ if and only if $(x \bmod n) = (y \bmod n)$

$x \equiv y$ if and only if $(x - y) = k*n$ for some k

commutativity; $(a + b) \bmod n = (b + a) \bmod n$

$(a*b) \bmod n = (b*a) \bmod n$

associativity; $a + (b + c) \bmod n = (a + b) + c \bmod n$

$a*(b*c) \bmod n = (a*b)*c \bmod n$

distributivity; $a*(b + c) \bmod n = ((a*b) + (a*c)) \bmod n$

identity; $a + 0 \bmod n = a$

$a*1 \bmod n = a$

inverse; $a + (-a) \bmod n = 0$

$a*(a^{-1}) \bmod n = 1$ if $a \neq 0$

reducibility; $(a + b) \bmod n = ((a \bmod n) + (b \bmod n)) \bmod n$

$(a*b) \bmod n = ((a \bmod n)*(b \bmod n)) \bmod n$

$(a*(b + c)) \bmod n = (((a*b) \bmod n) + ((a*c) \bmod n)) \bmod n$

It should also be noted that;

$$(a - b) \bmod n = a + (-b) \bmod n$$

$$(a \div b) \bmod n = a*(b^{-1}) \bmod n$$

$$(a \bmod n) \bmod n = a \bmod n$$

A.2 XOR Operations

XOR is the exclusive-or operation. In mathematical notation, it' s denoted by the symbol $\oplus$. It' s a standard function which operates on bits and it is commonly used in logic gates, in various software and hardware implementations, and in cryptographic applications. XOR is simply shown as below, where X and Y are two single-bit entries;

X	Y	X ⊕ Y
0	0	0
0	1	1
1	0	1
1	1	0

Also, for any *x, y, z, k* values as single-bit entries or bit-wise blocks, the following XOR properties are valid:

$x \oplus y = xy' + x'y$, where, ′ stands for the complement operation

$(x \oplus y)' = x \oplus y'$
$(x \oplus y)' = x' \oplus y$

commutativity; $x \oplus y = y \oplus x$

associativity; $(x \oplus y) \oplus z = x \oplus (y \oplus z)$

identity; $x \oplus x' = 1$
$x \oplus 1 = x'$
$x \oplus 0 = x$
$x \oplus x = 0$

transitivity; if $z = x \oplus y$, then $x = z \oplus y$
$y = x \oplus z$

$(x \oplus y) \oplus (z \oplus k) = (x \oplus z) \oplus (y \oplus k)$

reducibility; $x \oplus y = (x \oplus z) \oplus (y \oplus z)$

APPENDIX B

B.1 Avalanche Effect Analysis for 56-bit keyed DES in ECB Mode

In this research, some tests are made and some results are obtained using 56-bit standard DES algorithm in ECB mode and some other necessary programs all written in C language in order to observe the Avalanche Effect and diffusion performance of DES encryption algorithm. Related data and the results are as follows;

Test 1:

Sample Plaintext:

```
☺This is a trial for DES;to see the AVALANCHE effect.☻
```

It should be noted that all the data and outputs throughout all the tests are edited in MS-DOS platform, where some ASCII characters are not MS-WINDOWS standards; also some characters cannot be displayed properly, for instance in the above sample plaintext data, the last character is displayed as null since its original ASCII value was 0.

Encrypted outputs and related keys of the sample plaintext using 56-bit key standard DES algorithm - where the initial keys' sizes were 64 bits as the input to the encryption program - are all given as follows;

Key 1 (Shown in Binary and Decimal formats):

```
0100000001000000010000000100000001000000010000000100000000000000
64       64       64       64       64       64       64       0
```

Encrypted Sample Plaintext using Key 1:

```
¶[maJieL▼òL¥▒ùŞŞ→Æ▓ïıúTEmõ@`cÿÚã-AÍ■┌▓~u|◙uôfdØûÖ╣-▲¶▼
```

Key 2 (Shown in Binary and Decimal formats):

```
0100000001000000010000000100000001000000010000000100000000000001
64       64       64       64       64       64       64       1
```

Encrypted Sample Plaintext using Key 2:

```
☻◄òøÁ←ø|╝6©→►337?ilM×w{_╠IlÛObh¯ØjhÚwQsqÅåïıäŞöÜ133←┐↳Y☻
```

As it can be seen from the data above, only changing 1 bit of the encryption key (the eighth byte of the Key 1 is changed from value 0 to 1, where only a single bit, the first bit from the right is differentiated) and using this new key as Key 2, made very significant changes in the ciphertexts. These changes can be seen both from the output data characters of both ciphertexts as shown above, or from the Table B.2. Thus, changing a single bit in the key made great diffusions all around the ciphertext data where the plaintext input was exactly the same. There were similar results with other sample data which showed that DES had a good avalanche effect property. For both

tables, each row corresponds to the binary and decimal ASCII values of each character (one byte) in the plaintext starting from the 1st byte from the left.

Table B.1 ASCII character values for the Sample Plaintext.

Binary	Decimal
00000001	1
01010100	84
01101000	104
01101001	105
01110011	115
00100000	32
01101001	105
01110011	115
00100000	32
01100001	97
00100000	32
01110100	116
01110010	114
01101001	105
01100001	97
01101100	108
00100000	32
01100110	102
01101111	111
01110010	114
00100000	32
01000100	68
01000101	69
01010011	83
00111011	59
01110100	116
01101111	111
00100000	32
01110011	115
01100101	101
01100101	101
00100000	32
01110100	116
01101000	104
01100101	101
00100000	32
01000001	65
01010110	86
01000001	65
01001100	76
01000001	65
01001110	78
01000011	67
01001000	72
01000101	69
00100000	32
01100101	101
01100110	102
01100110	102
01100101	101
01100011	99
01110100	116
00101110	46
00000010	2
11111111	255
00000000	0

Table B.2 ASCII values for the characters of the Ciphertexts encrypted with Key 1 and Key 2.

Ciphertext 1 (encrypted with Key 1)		*Ciphertext 2 (encrypted with Key 2)*	
Binary	Decimal	Binary	Decimal
11110100	244	00000010	2
01011011	91	10101100	172
01101101	109	10010101	149
01100001	97	10011011	155
01001010	74	10110101	181
11101100	236	00011011	27
01100101	101	10011011	155
01001100	76	10110011	179
00011111	31	10111100	188
10010101	149	00110110	54
00011100	28	10111000	184
10111110	190	00010110	22
10110000	176	00010000	16
10010111	151	00110011	51
10011110	158	00110011	51
10011110	158	00110111	55
00011010	26	11100111	231
10010010	146	01101001	105
10010010	146	01101100	108
10110010	178	01001101	77
00010111	23	11101000	232
10001011	139	01110111	119
10001101	141	01111011	123
10100011	163	01011111	95
11000010	194	11001100	204
01000101	69	01001001	73
01101101	109	01101100	108
11100100	228	11101010	234
01000000	64	01001111	79
01100000	96	01100010	98
01100011	99	01101000	104
11101101	237	11101110	238
11101001	233	01001111	79
11000110	198	01101010	106
11000100	196	01101000	104
01000001	65	11101001	233
11010110	214	01110111	119
11111110	254	01010001	81
11011010	218	01110011	115
11011011	219	01110001	113
01111110	126	10001111	143
01110101	117	10000110	134
01111100	124	10001011	139
01111111	127	10001101	141
01110101	117	10000100	132
11100010	226	00010101	21
01100110	102	10010100	148
01100100	100	10011010	154
10011101	157	00110001	49
10010110	150	00110011	51
10011001	153	00110011	51
10111001	185	00011011	27
00010110	22	10111111	191
00000110	6	10101100	172
11110100	244	01011001	89
00000011	3	10101001	169

Test 2:

In this test, some similar trials are also made for the analysis of avalanche effect with DES; but this time the same key is used for encrypting both of the plaintext data, while slight changes are being made in the plaintext pairs. In the previous test, while making some slight changes in the key, the plaintext was kept unchanged and the differences in the ciphertext pairs were observed. The changes and diffusion effects are analyzed among the ciphertext pairs.

Sample Plaintext 1:

```
►Divide et impera,it's no secret
```

Sample Plaintext 2:

```
ÉDivide et impera-it's no secret
```

Only two characters, the first and 18th from the left are different in the plaintexts above.

Key (Shown in Binary and Decimal formats):

```
0000000100011011011100000001000011111111100000000100010000010001
1       27      112     16      255     128     68      17
```

Encrypted outputs of the sample plaintexts using 56-bit key standard DES algorithm - where the initial key size was again 64 bits as the input to the encryption program, and it was used for encrypting both sample plaintexts 1 & 2 - are all given as follows;

Encrypted Sample Plaintext 1:

```
(åô╣ ûşù↕gH´`eOkN╚B╚ÕHÕB╗kÔLobKfF
```

Encrypted Sample Plaintext 2:

```
jçş╣ ûæ£►gH´`eOkNò▲òÞ→│←ØkÔLobKfF
```

In this study, the difference between Plaintext 1 and 2 is achieved only by the change of two bits, the 1st bit from the left in the 1st byte and the 1st bit from the right in the 18th byte respectively, which also made slight differences in the 1st and 18th characters of the plaintext data due to the change in their ASCII values. However, after encrypting these two plaintexts with exactly the same key, the first and third 8-byte data blocks in the ciphertexts were composed of almost completely different characters, thus a 2-bit difference in the plaintext pairs affected approximately 14 bytes of data in the ciphertext pairs. These can also be analyzed from the Table B.3. Similar trials are also made resulting with acceptable avalanche effect rates, however the rates and diffusion effects achieved in Test 1 seemed to be much higher.

It must be stressed that; since ECB operation mode of DES encryption algorithm is used during these studies, the differences in the ciphertext pairs occur only amongst the 8-byte blocks in which the bit changes exist. In order to affect the whole ciphertext output and to widen the diffusion effect throughout all data blocks, at least 1

or 2 bits must be changed in each and every 8-byte blocks in the plaintext input. This is due to the structure of ECB mode. On the other hand, if any of the other operation modes of DES had been used rather than ECB, the diffusion rate and the amount of avalanche effect would have been much higher with the same amount of bit changes in the plaintexts.

Table B.3 Plaintext and corresponding Ciphertext pairs where 2 bits differ in the plaintext inputs and encrypted with the same key.*

Plaintext 1		*Plaintext 2*		*Ciphertext 1*		*Ciphertext 2*	
Binary	Dec.	Binary	Dec.	Binary	Dec.	Binary	Dec.
00010000	**16**	**1**0010000	**144**	00101**0**00	**40**	01101**0**10	**106**
01000100	68	01000100	68	1000**1**111	**143**	1000**0**111	**135**
01101001	105	01101001	105	100**1**00**1**1	**147**	100**1**11**1**0	**158**
01110110	118	01110110	118	10111001	185	10111001	185
01101001	105	01101001	105	10010110	150	10010110	150
01100100	100	01100100	100	1001**1**111	**159**	1001**0**00**1**	**145**
01100101	101	01100101	101	1001**0**111	**151**	1001**1**100	**156**
00100000	32	00100000	32	00010**0**10	**18**	00010**0**00	**16**
01100101	101	01100101	101	01100111	103	01100111	103
01110100	116	01110100	116	01001000	72	01001000	72
00100000	32	00100000	32	11101111	239	11101111	239
01101001	105	01101001	105	01100000	96	01100000	96
01101101	109	01101101	109	01100101	101	01100101	101
01110000	112	01110000	112	01001111	79	01001111	79
01100101	101	01100101	101	01101011	107	01101011	107
01110010	114	01110010	114	01001110	78	01001110	78
01100001	97	01100001	97	**1**1001000	**200**	**1**0010101	**149**
00101**1**0**0**	**44**	00101**1**0**1**	**45**	**0**1000**1**0	**66**	00011**1**10	**30**
01101001	105	01101001	105	**1**1001000	**200**	**1**0010101	**149**
01110100	116	01110100	116	**1**1**1**00**1**01	**229**	**1**0111000	**184**
00100111	39	00100111	39	01001000	72	00011010	26
01110011	115	01110011	115	**1**1**1**00**1**01	**229**	**1**0110011	**179**
00100000	32	00100000	32	**0**1000**1**0	**66**	00011011	**27**
01101110	110	01101110	110	**1**1001011	**203**	**1**0011**1**01	**157**
01101111	111	01101111	111	01101011	107	01101011	107
00100000	32	00100000	32	11100010	226	11100010	226
01110011	115	01110011	115	01001100	76	01001100	76
01100101	101	01100101	101	01101111	111	01101111	111
01100011	99	01100011	99	01100010	98	01100010	98
01110010	114	01110010	114	01001011	75	01001011	75
01100101	101	01100101	101	01100110	102	01100110	102
01110100	116	01110100	116	01000110	70	01000110	70

** The changes in bits and corresponding ASCII decimal values are denoted with bold character printing.*