Investigation of Symmetric Block Cipher Algorithms

http://amberj.devio.us/projects/crypto/

Dissertation Submitted in Partial fulfillment of the Requirement for the Award of the Degree of

Master of Computer Application

Semester VIII

Session Jan – May, 2011

Under the guidance of
Mrs. Yasmin Shaikh

Submitted By
Amber Jain
IC-2K7-05

International Institute of Professional Studies
Devi Ahilya Vishwavidyalaya, Indore, M.P.
2011

International Institute of Professional Studies
Devi Ahilya Vishwavidyalaya, Indore, M.P.

DECLARATION

I hereby declare that the project entitled "**Investigation of Symmetric Block Cipher Algorithms**" which is submitted by me for the partial fulfillment of the requirement for the award of Master of Computer Application (6 Years) VIII Semester to International Institute of Professional Studies, Devi Ahilya Vishwavidyalaya, Indore, comprises my own work and due acknowledgement has been made in text to all other material used.

Signature of Student:

Date:

Place:

International Institute of Professional Studies
Devi Ahilya Vishwavidyalaya, Indore, M.P.

CERTIFICATE

It is to certify that we have examined the dissertation on "**Investigation of Symmetric Block Cipher Algorithms**", submitted by **Mr. Amber Jain** to the International Institute of Professional Studies, DAVV, Indore and hereby accord our approval of it as a study carried out and presented in a manner required for its acceptance in partial fulfillment for the award of the degree of "Master of Computer Application (6 Years) VIII Semester".

Internal Examiner

Signature**:** ________________

Name **:** ________________

Date **:** ________________

External Examiner

Signature**:** ________________

Name **:** ________________

Date **:** ________________

Course Coordinator

Signature**:** ________________

Name **:** ________________

Date **:** ________________

Project Guide

Signature**:** ________________

Name **:** ________________

Date **:** ________________

ACKNOWLEDGEMENTS

This work is the result of inspiration, support, guidance, cooperation and facilities that were extended to me by persons at all levels and I am indebted to all of them.

I cannot fully express my gratitude towards **Mrs. Yasmin Shaikh** who believed in me from the start. It is my proud privilege to have worked under her guidance, who generously shared her wisdom and expertise with me and has provided me with excellent guidance.

I'm also indebted to all my teachers who have guided me towards the right direction. They provided me the support, necessary guidance, valuable suggestions and helped me in this work.

I would also like to thank **Mr. B.K. Tripathi** (Director, IIPS) and **Mr. Ramesh Thakur** (Course coordinator, MCA, IIPS) for their valuable guidance and support.

I also extend thanks to my parents for their unending support and belief that has helped me enormously to complete this project.

Amber Jain

ABSTRACT

Cryptography is the art and science of keeping messages secure. Cryptography ensures information security which is protecting information and information systems from unauthorized access, use, disclosure, disruption, modification, perusal, inspection, recording or destruction. Cryptography performs 4 key functions: confidentiality, authentication, integrity and non-repudiation.

Symmetric key algorithms are better suited for bulk data encryption (i.e. confidentiality) than public key cryptography. A block cipher is a symmetric key cipher operating on fixed-length groups of bits, called blocks, with an unvarying transformation in contrast to stream cipher where plaintext bits are combined with a pseudorandom cipher bit stream.

Though one can propose a new symmetric cipher algorithm using standard constructs (e.g. feistel cipher, substitution permutation network etc.), designing a secure cipher is very difficult. Therefore, naïve or amateur cryptographers almost always design flawed cryptosystems. Other than peer review and years of cryptanalysis, there are many other decisions, guidelines and criteria that increase security of the cryptographic algorithm (and ultimately increase the confidence in the algorithm).

This project titled "Investigation of Symmetric Block Cipher Algorithms" (http://amberj.devio.us/projects/crypto/) involves the study of concepts of cryptography, design constructs (and techniques) and symmetric block cipher algorithms. This work also collects and documents a comprehensive list of design decisions (or guidelines) of symmetric block cipher algorithms cryptographic algorithms whose subset can be used to design secure algorithms.

TABLE OF CONTENTS

LIST OF FIGURES AND TABLES

INTRODUCTION

CHAPTER 1
INTRODUCTION

1.1 Motivation:

Cryptography is the practice and study of hiding information. Modern cryptography follows a strongly scientific approach. Quoting Bruce Schneier [1]:

> "Design work is the mainstay of the science of cryptography, and it is very specialized. Cryptography blends several areas of mathematics: number theory, complexity theory, information theory, probability theory, abstract algebra, and formal analysis, among others. Few can do the science properly, and a little knowledge is a dangerous thing: inexperienced cryptographers almost always design flawed systems. Good cryptographers know that nothing substitutes for extensive peer review and years of analysis. Quality systems use published and well-understood algorithms and protocols; using unpublished or unproven elements in a design is risky at best. Cryptographic system design is also an art. A designer must strike a balance between security and accessibility, anonymity and accountability, privacy and availability. Science alone cannot prove security; only experience, and the intuition born of experience, can help the cryptographer design secure systems and find flaws in existing designs. "

1.2 Vision:

"…to explore cryptographic concepts, constructs, design guidelines and algorithms".

1.3 Aim:

To infer desirable design decisions (or guidelines) that must be taken into consideration when designing 'secure' cryptosystems.

1.4 Objectives:

- To learn about cryptographic concepts, constructs and algorithms.
- To explore existing symmetric block cipher algorithms.
- To infer desirable design decisions that must be taken into consideration when designing 'secure' cryptosystems.

1.5 Organization of Report:

Chapter 2 contains brief information about the literature reviewed from different sources. Chapter 3 contains analysis of various cryptographic constructs and structures which can be used to design cryptographic algorithms. Chapter 4 contains details about research methodology including the symmetric block cipher algorithms which were investigated.

Chapter 5 contains details about the scope of work. Chapter 6 lists comparisons between symmetric block cipher algorithms that were investigated and inferred design guidelines. Chapter 7 lists conclusions drawn based on obtained results.

LITERATURE REVIEW

CHAPTER 2
LITERATURE REVIEW

2.1 Terminology:

A message is *plaintext*. The process of disguising a message in such a way as to hide its substance (as well as meaning) is *encryption* (also called *enciphering*). An encrypted message is *ciphertext*. The process of turning ciphertext back into plaintext is *decryption* (also termed as *deciphering*). A *key* is a piece of information (a parameter) that determines the functional output of a cryptographic algorithm or *cipher*. In encryption, a key specifies the particular transformation of plaintext into ciphertext, or vice versa during decryption. The range of possible values of the key is called the *keyspace*. A *cryptosystem* is an algorithm, plus all possible plaintexts, ciphertexts, and keys.

Encryption and decryption functions (see Section 2.3) are represented as:

$$E_K(M) = C$$

$$D_K(C) = M$$

These functions have the property that:

$$D_K(E_K(M)) = M$$

Some algorithms use a different encryption key and decryption key (see Section 2.3). That is, the encryption key, K1, is different from the corresponding decryption key, K2. In this case:

$$E_{K1}(M) = C$$

$$D_{K2}(C) = M$$

$$D_{K2}(E_{K1}(M)) = M$$

The art and science of keeping messages secure is *cryptography*, and it is practiced by *cryptographers*. *Cryptanalysis* is the art and science of recovering the plaintext of a message (or breaking the ciphertext) without access to the key. Successful cryptanalysis may recover the plaintext or the key. *Cryptanalysts* are practitioners of cryptanalysis; that is, seeing through the disguise. The branch of mathematics encompassing both cryptography and cryptanalysis is *cryptology* and its practitioners are *cryptologists*.

In contrast, a *code* is a method used to transform a message into an obscured form, preventing those who do not possess special information, or key, required to apply the transform from understanding what is actually transmitted. The usual method is to use a *codebook* with a list of common phrases or words matched with a *codeword*. Encoded messages are sometimes termed *codetext*, while the original message is referred to as plaintext.

2.2 Functions of Cryptography:

The main function of cryptography is to ensure *information security* which is protecting information and information systems from unauthorized access, use, disclosure, disruption, modification, perusal, inspection, recording or destruction. Cryptography includes techniques such as encryption/decryption, hashing, digital signatures, timestamps etc. to ensure information security. Broadly speaking, cryptography performs following 4 key functions:

- **Confidentiality**: Protection against the disclosure of information to parties other than the intended recipient (for example: encryption and decryption).
- **Authentication**: It should be possible for the receiver of a message to ascertain its origin; an intruder should not be able to masquerade as someone else (for e.g. public key cryptography and digital signatures).
- **Integrity**: It should be possible for the receiver of a message to verify that it has not been modified in transit; an intruder should not be able to substitute a false message for a legitimate one (for e.g. hashing).
- **Non-repudiation**: A sender should not be able to falsely deny later that he sent a message (for e.g. digital signatures with timestamps).

2.3 Types of Key-based Algorithms:

There are two general types of key-based algorithms:

- **Symmetric Algorithms**: These are also referred to as secret-key algorithms, single-key algorithms, conventional algorithms, shared algorithms, private-key algorithms or one-key algorithms. These are the algorithms where the encryption key can be calculated from the decryption key and vice versa. In most symmetric algorithms, the encryption key and the decryption key are trivially related, often identical. Encryption and decryption with a symmetric algorithm are denoted by:

  $E_K(M) = C$

  $D_K(C) = M$

  Symmetric algorithms can be represented diagrammatically as:

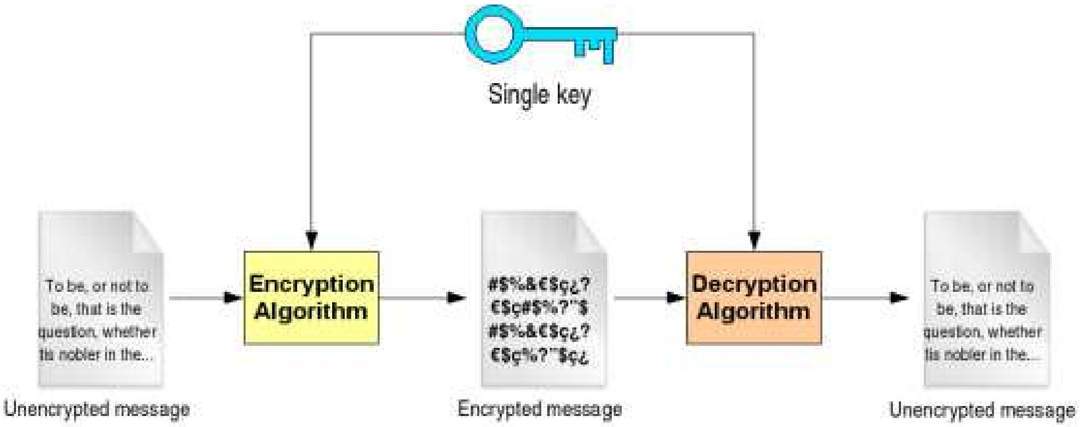
Single key
To be, or not to be, that is the question, whether tis nobler in the...
Unencrypted message
Encryption Algorithm
#$%&€$ç¿?
€$ç#$%?"$
#$%&€$ç¿?
€$ç%?"$ç¿
Encrypted message
Decryption Algorithm
To be, or not to be, that is the question, whether tis nobler in the...
Unencrypted message

Figure 2.1: Symmetric-key algorithm (Source: [4])

These algorithms require that the sender and receiver agree on a key before they can communicate securely. The security of a symmetric algorithm rests in the key; divulging the key means that anyone could encrypt and decrypt messages. As long as the communication needs to remain secret, the key must remain secret. Some examples of popular and well-respected symmetric algorithms include Twofish, Serpent, AES (Rijndael), Blowfish, CAST5, RC4, 3DES, and IDEA.

Symmetric algorithms can be divided into two categories: stream ciphers and block ciphers.

- A *block cipher* is a symmetric key cipher operating on fixed-length groups of bits, called blocks, with an unvarying transformation. A block cipher encryption algorithm might take (for example) a 128-bit block of plaintext as input, and output a corresponding 128-bit block of ciphertext. The exact transformation is controlled using a second input — the secret key. Decryption is similar: the decryption algorithm takes, in this example, a 128-bit block of ciphertext together with the secret key, and yields the original 128-bit block of plaintext.
- A *stream cipher* is a symmetric key cipher where plaintext bits are combined with a pseudorandom cipher bit stream (keystream), typically by an exclusive-or (xor) operation. In a stream cipher the plaintext digits are encrypted one at a time, and the transformation of successive digits varies during the encryption. An alternative name

is a state cipher, as the encryption of each digit is dependent on the current state. In practice, the digits are typically single bits or bytes.

- **Public-key Algorithms**: This cryptographic approach involves the use of *asymmetric key algorithms* — that is, the non-message information (the *public key* or the encryption key) needed to transform the message to a secure form is different from the information needed to reverse the process (the *private key* or the decryption key). The person who anticipates receiving messages first creates both a public key and an associated private key, and publishes the public key. When someone wants to send a secure message to the creator of these keys, the sender encrypts it (transforms it to secure form) using the intended recipient's public key; to decrypt the message, the recipient uses the private key.

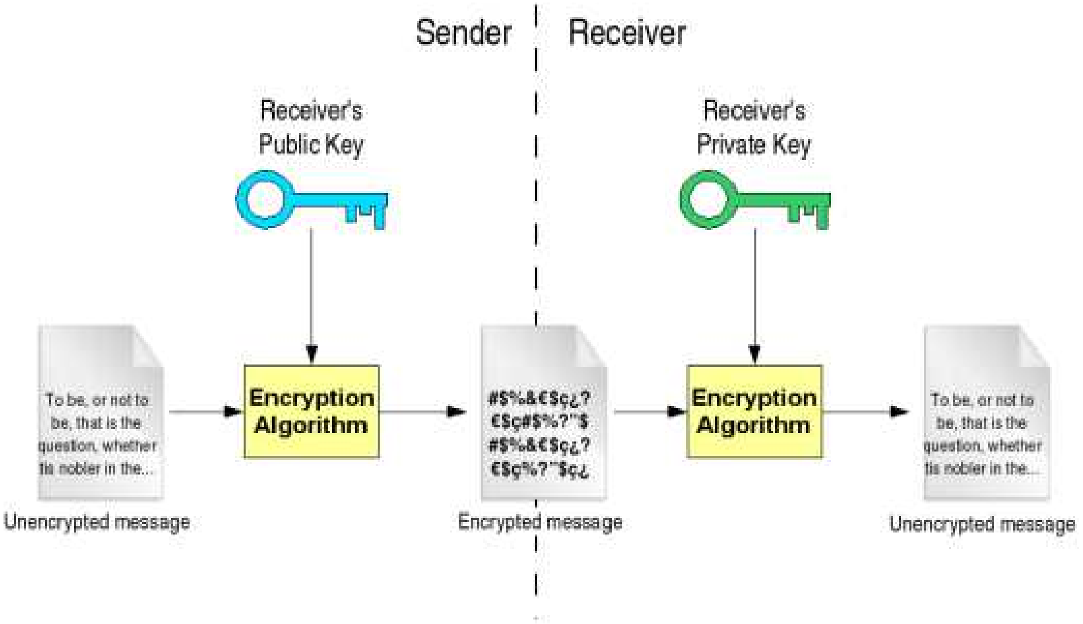
Sender
Receiver
Receiver's Public Key
Receiver's Private Key
To be, or not to be, that is the question, whether tis nobler in the...
Encryption Algorithm
#$%&€$ç¿? €$ç#$%?"$ #$%&€$ç¿? €$ç%?"$ç¿
Encryption Algorithm
To be, or not to be, that is the question, whether tis nobler in the...
Unencrypted message
Encrypted message
Unencrypted message

Figure 2.2: Public-key algorithm (Source: [5])

Encryption using public key K is denoted by:

$E_K(M) = C$

Even though the public key and private key are different, decryption with the corresponding private key is denoted by:

$D_K(C) = M$

Unlike symmetric key algorithms, a public key algorithm does not require a secure initial exchange of one or more secret keys between the sender and receiver. The particular algorithm used for encrypting and decrypting was designed in such a way that, while it is easy for the intended recipient to generate the public and private keys and to decrypt the message using the private key, and while it is easy for the sender to encrypt the message using the public key, it is extremely difficult for anyone to figure out the private key based on their knowledge of the public key.

The use of these keys also allows protection of the authenticity of a message by creating a *digital signature* of a message using the private key, which can be verified using the public key.

Examples of well-regarded asymmetric key techniques for varied purposes include Diffie–Hellman key exchange protocol, ElGamal, various elliptic curve techniques, Paillier cryptosystem and RSA encryption algorithm.

2.4 Security of Cryptosystem:

Cryptanalysis is the science of recovering the plaintext of a message without access to the key. Successful cryptanalysis may recover the plaintext or the key. It also may find weaknesses in a cryptosystem that eventually lead to the previous results. (The loss of a key through noncryptanalytic means is called a *compromise*.). An attempted cryptanalysis is called an *attack*.

2.4.1 Kerckhoffs' Principle:

This is a fundamental assumption in cryptanalysis. Kerckhoffs’ principle (also called Kerckhoffs' assumption, axiom or law) was stated by Auguste Kerckhoffs. It states that [3]: "The method must not need to be kept secret, and having it fall into the enemy's hands should not cause problems."

Claude Shannon reformulated this principle (called Shannon's maxim) in his own words as "The enemy knows the system." If algorithm is secure even with the knowledge of inner workings of cipher, then it certainly is secure without the detailed specification of the cipher. The most secure cryptographic algorithms are the ones that have been put into public domain, have been thoroughly cryptanalysed by world’s best cryptanalysts for years, and are still unbreakable.

2.4.2 Types of Cryptanalytic Attacks:

There are four general types of cryptanalytic attacks [2] (with the assumption that Kerckhoffs' principle applies):

- **Ciphertext-only attack**. The cryptanalyst has the cipher text of several messages, all of which have been encrypted using the same encryption algorithm. The cryptanalyst's job is to recover the plaintext of as many messages as possible, or better yet to deduce the key (or keys) used to encrypt the messages, in order to decrypt other messages encrypted with the same keys.

  Given: $C_1 = E_k(P_1)$, $C_2 = E_k(P_2)$,...$C_i = E_k(P_i)$

  Deduce: Either P_1, P_2,...P_i; k; or an algorithm to infer P_{i+1} from $C_{i+1} = E_k(P_{i+1})$
- **Known-plaintext attack**. The cryptanalyst has access not only to the ciphertext of several messages, but also to the plaintext of those messages. His job is to deduce the key (or keys) used to encrypt the messages or an algorithm to decrypt any new messages encrypted with the same key (or keys).

  Given: P_1, $C_1 = E_k(P_1)$, P_2, $C_2 = E_k(P_2)$,...P_i, $C_i = E_k(P_i)$

  Deduce: Either k, or an algorithm to infer P_{i+1} from $C_{i+1} = E_k(P_{i+1})$
- **Chosen-plaintext attack**. The cryptanalyst not only has access to the ciphertext and associated plaintext for several messages, but he also chooses the plaintext that gets encrypted. This is more powerful than a known-plaintext attack, because the cryptanalyst can choose specific plaintext blocks to encrypt, ones that might yield more information about the key. His job is to deduce the key (or keys) used to encrypt the messages or an algorithm to decrypt any new messages encrypted with the same key (or keys).

  Given: P_1, $C_1 = E_k(P_1)$, P_2, $C_2 = E_k(P_2)$,...P_i, $C_i = E_k(P_i)$, where the cryptanalyst gets to choose P_1, P_2,...P_i

  Deduce: Either k, or an algorithm to infer P_{i+1} from $C_{i+1} = E_k(P_{i+1})$
- **Adaptive-chosen-plaintext attack**. This is a special case of a chosen-plaintext attack. Not only can the cryptanalyst choose the plaintext that is encrypted, but he can also modify his choice based on the results of previous

encryption. In a chosen-plaintext attack, a cryptanalyst might just be able to choose one large block of plaintext to be encrypted; in an adaptive-chosen-plaintext attack he can choose a smaller block of plaintext and then choose another based on the results of the first, and so forth.

There are at least three other types of cryptanalytic attacks:

- **Chosen-ciphertext attack**. The cryptanalyst can choose different ciphertexts to be decrypted and has access to the decrypted plaintext. For example, the cryptanalyst has access to a tamperproof box that does automatic decryption. His job is to deduce the key.

  Given: C_1, $P_1 = D_k(C_1)$, C_2, $P_2 = D_k(C_2)$,...C_i, $P_i = D_k(C_i)$

  Deduce: k

  This attack is primarily applicable to public-key algorithms. A chosen-ciphertext attack is sometimes effective against a symmetric algorithm as well. (Sometimes a chosen-plaintext attack and a chosen-ciphertext attack are together known as a *chosen-text attack*.)
- **Chosen-key attack**. This attack doesn't mean that the cryptanalyst can choose the key; it means that he has some knowledge about the relationship between different keys. It's strange and obscure, not very practical.
- **Rubber-hose cryptanalysis**. The cryptanalyst threatens, blackmails, or tortures someone until they give him the key. Bribery is sometimes referred to as a *purchase-key attack*. These are all very powerful attacks and often the best way to break an algorithm.

2.4.3 Information Theory:

Claude Shannon published his pioneering work on modern introduction theory [6, 7]. *Information theory* is a branch of applied mathematics involving the quantification of information. Information theory is mainly used in cryptography to prove lower bounds on the size of the secret key required to achieve a certain level of security in secrecy and authentication systems.

- **Amount of Information**: In [2], amount of information in a message is defined as the minimum number of bits needed to encode all possible meanings of that message, assuming all messages are equally likely.

- **Entropy**: Formally, the amount of information in a message M is measured by the entropy of a message, denoted by H(M) (as defined in [2]). The entropy of a message also measures its *uncertainty* which is the number of plaintext bits needed to be recovered when the message is scrambled in ciphertext in order to learn the plaintext.
- **Rate of the Language**: For a given language, the rate of the language is defined in [2] as:

  $r = H(M)/N$

  in which N is the length of the message. The rate of normal English takes various values between 1.0 bits/letter and 1.5 bits/letter, for large values of N. The *absolute rate of a language* is the maximum number of bits that can be coded in each character, assuming each character sequence is equally likely. If there are L characters in a language, the absolute rate is:

  $R = \log_2 L$

  This is the maximum entropy of the individual characters. For example, English, with 26 letters, has the absolute rate of $\log_2 26$, or about 4.7 bits/letter.
- **Redundancy of Language**: The redundancy of a language, denoted by D, is defined by [2]:

  $D = R - r$

  For example, given that the rate of English is 1.3, the redundancy is 3.4 bits/letter. This means that each English character carries 3.4 bits of redundant information.
- For a message of length n, the number of different keys that will decipher a ciphertext message to some intelligible plaintext in the same language as the original plaintext (such as an English text string) is given by the following formula (as given in [2]):

  $2^{H(K)- nD} - 1$
- **Unicity Distance**: In [2], Unicity distance (denoted by U), (also called the unicity point) is an approximation of the amount of ciphertext such that the sum of the real information (entropy) in the corresponding plaintext plus the entropy of the encryption key equals the number of ciphertext bits used [2].

For most symmetric cryptosystems, the unicity distance is defined as the entropy of the cryptosystem divided by the redundancy of the language [2]:

$U = H(K)/D$

2.4.4 Security of Cryptosystem Based on Information Theory:

- A mathematical model of a secure cryptosystems was defined by Shannon. The aim of cryptanalyst is to determine either the key or plaintext or both. But he may as well be satisfied with some probabilistic information about the plaintext. Cryptanalysts use the natural redundancy of language to reduce the number of possible plaintexts. The more redundant the language, the easier it is to cryptanalyze. For example, a cryptanalyst might have information like the language of plaintext or standard endings and/or beginnings as in source code both of which have certain redundancy associated with it (i.e. ciphertext unavoidably yields some information about the corresponding plaintext). A good cryptographic algorithm keeps this information to a minimum; a good cryptanalyst exploits this information to determine the plaintext. The purpose of cryptanalysis is to modify the probabilities associated with each possible plaintext. And, then there is a finite probability that one plaintext will result from all possible plaintexts as certain.
- The entropy of a cryptosystem is a measure of the size of the keyspace, K. It is approximated by the base two logarithm of the number of keys:

  $H(K) = \log_2 K$

  In general, the greater the entropy, the harder it is to break a cryptosystem.
- Shannon showed that ciphertexts longer than unicity distance are reasonably certain to have only one meaningful decryption. Shorter ciphertexts are very likely to have many different meaningful decryptions and thus increasing the difficulty on behalf of attacker to choose the correct one. One important consideration is that unicity distance gives probabilistic results. Unicity distance estimates the minimum amount of ciphertext for which it is likely that there is only a single intelligible plaintext decryption when a brute-force attack is attempted. Also, unicity distance guarantees insecurity if it’s too small but does not guarantee security if it’s high.

- **Ideal Secrecy**: Shannon defined a cryptosystem whose unicity distance is infinite as one that has *ideal secrecy*. Note that an ideal cryptosystem is not necessarily a perfect cryptosystem, although a perfect cryptosystem would necessarily be an ideal cryptosystem. If a cryptosystem has ideal secrecy, even successful cryptanalysis will leave some uncertainty about whether the recovered plaintext is the real plaintext.
- **Perfect Secrecy**: A perfect cryptosystem is the one in which ciphertext yields no possible information about the plaintext (except possibly the length of plaintext). Shannon theorized that it is only possible if the number of possible keys is at least as large as the number of possible messages. In other words, the key must be at least as long as the message itself, and no key can be reused. This means that one-time pad is the only cryptosystem which achieves perfect secrecy.
- **One-time Pad (OTP)**: According to [8], OTP is a type of encryption, which is impossible to crack if used correctly. Each bit (or character) from the plaintext is encrypted by a modular addition with a bit (or character) from a secret random key (or pad) of the same length as the plaintext, resulting in a ciphertext. If the key is truly random, as large as or greater than the plaintext, never reused in whole or part, and kept secret, the ciphertext will be impossible to decrypt or break without knowing the key. A given ciphertext message is equally likely to correspond to any possible plaintext message of equal size. And thus, it is impossible for the cryptanalyst to determine which plaintext message is the correct one.

2.4.5 Confusion and Diffusion:

Confusion and diffusion are two desirable properties of the operation of a secure cipher which were identified by Claude Shannon in [7].

- **Confusion**: According to [9], confusion refers to making the relationship between the key and the ciphertext as complex and involved as possible. One aim of confusion is to make it very hard to find the key even if one has a large number of plaintext-ciphertext pairs produced with the same key. Therefore, each bit of the ciphertext should depend on the entire key, and in different ways on different bits of the key. In particular, changing one bit of the key should change the ciphertext completely.

- **Diffusion**: According to [9], diffusion refers to the property that the redundancy in the statistics of the plaintext is "dissipated" in the statistics of the ciphertext. Diffusion means that the output bits should depend on the input bits in a very complex way. In a cipher with good diffusion, if one bit of the plaintext is changed, then the ciphertext should change completely, in an unpredictable or pseudorandom manner. More generally, one may require that flipping a fixed set of bits should change each output bit with probability one half.

The simplest way to achieve both diffusion and confusion is a substitution-permutation network (see Section 3.1.4). In these systems, the plaintext and the key often have a very similar role in producing the output, hence it is the same mechanism that ensures both diffusion and confusion [9].

2.4.6 Avalanche Effect:

- According to [10], avalanche effect refers to a desirable property of cryptographic algorithms, typically block ciphers and cryptographic hash functions. The avalanche effect is evident if, when an input is changed slightly (for example, flipping a single bit) the output changes significantly (e.g., half the output bits flip). In the case of quality block ciphers, such a small change in either the key or the plaintext should cause a drastic change in the ciphertext. The actual term was first used by Horst Feistel, although the concept dates back to at least Shannon's diffusion (see Section 2.4.5).

  If a block cipher or cryptographic hash function does not exhibit the avalanche effect to a significant degree, then it has poor randomization, and thus a cryptanalyst can make predictions about the input, being given only the output. This may be sufficient to partially or completely break the algorithm. Thus, the avalanche effect is a desirable condition from the point of view of the designer of the cryptographic algorithm or device.

- **Strict Avalanche Criterion**: Strict avalanche criterion (SAC), a generalization of the avalanche effect, is satisfied if each of the output bits changes with a 50% probability, whenever a single input bit is complemented.
- **Bit Independence Criterion**: According to [10], bit independence criterion (BIC) states that output bits j and k should change independently when any single input bit i is inverted, for all i, j and k

2.4.7 Complexity Theory:

Complexity theory provides a methodology for analyzing the computational complexity of different cryptographic techniques and algorithms.

An algorithm's complexity is determined by the computational power needed to execute it. The computational complexity of an algorithm is often measured by two variables: T (for time complexity) and S (for space complexity, or memory requirement). Both T and S are commonly expressed as functions of n, where n is the size of the input.

Generally, the computational complexity of an algorithm is expressed in terms of order of growth of functions. A common notation to represent complexity is "big O" notation (which gives asymptotic upper bound to algorithm's running time). It's just the term of the complexity function which grows the fastest as *n* gets larger; all lower-order terms are ignored.

Table 2.1 lists some algorithms with their complexities (Partly adopted from [2]):

Class	Complexity
Constant	O(1)
Polynomial	$O(n^m)$ where m=constant
Linear (Polynomial)	O(n)
Quadratic (Polynomial)	$O(n^2)$
Cubic (Polynomial)	$O(n^3)$
Exponential	$O(t^{f(n)})$ where t is a constant greater than 1 and f(n) is some polynomial function of n
Superpolynomial	$O(c^{f(n)})$

	where c is a constant and f(n) is more than constant but less than linear

Table 2.1: Complexities of different classes of algorithms

Information theory tells us that all cryptographic algorithms (except one-time pads) can be broken. Complexity theory tells us whether an attack on them is feasible in polynomial time (and thus making algorithm insecure). Modern cryptography follows a strongly scientific approach, and designs cryptographic algorithms around computational hardness assumptions, making such algorithms hard to break by an adversary. Such systems (also termed as *computationally secure systems* in contrast to *unconditionally secure systems* like one-time pad) are not unbreakable in theory but it is infeasible to do so by any practical means.

According to [2], ideally a cryptographer would like to be able to say that the best algorithm to break this encryption algorithm is of exponential-time complexity. In practice, given the current state of the art of computational complexity theory, all known cracking algorithms for cryptosystems are of superpolynomial-time complexity.

ANALYSIS

CHAPTER 3

ANALYSIS

3.1 Cryptographic Constructs And Structures:

Over years, many cryptographic algorithms have been proposed and many do use combination of different standard constructs at different levels. A comprehensive list includes:

3.1.1 Substitution Box:

S-Box (Substitution-box) is a basic construct of symmetric key algorithms which performs substitution. In block ciphers, they are typically used to obscure the relationship between the key and the ciphertext i.e. Shannon's property of confusion (see Section 2.4.5). In many algorithms, the S-Boxes are carefully chosen to resist cryptanalysis.

An S-Box takes some number of input bits, m, and transforms them into some number of output bits, n: an m×n S-Box can be implemented as a lookup table with 2^m words of n bits each. Fixed as well as dynamically generated tables might be used.

3.1.2 Permutation Box:

A permutation box (or P-box) is a method of bit-shuffling used to permute or transpose bits across S-boxes inputs, retaining (Shannon’s) diffusion (see Section 2.4.5) while transposing. A P-box is a permutation of all the bits: it takes the outputs of all the S-boxes of one round, permutes the bits, and feeds them into the S-boxes of the next round. A good P-box has the property that the output bits of any S-box are distributed to as many S-box inputs as possible.

3.1.3 Feistel Cipher:

A Feistel cipher (also called Feistel network) is a symmetric structure used in the construction of block ciphers named after cryptographic pioneer Horst Feistel. The Feistel structure has the advantage that encryption and decryption operations are very similar, even identical in some cases, requiring only a reversal of the key schedule. A Feistel network is an iterated cipher with an internal function called a round function. Figure 3.1 shows both encryption and decryption using feistel construction:

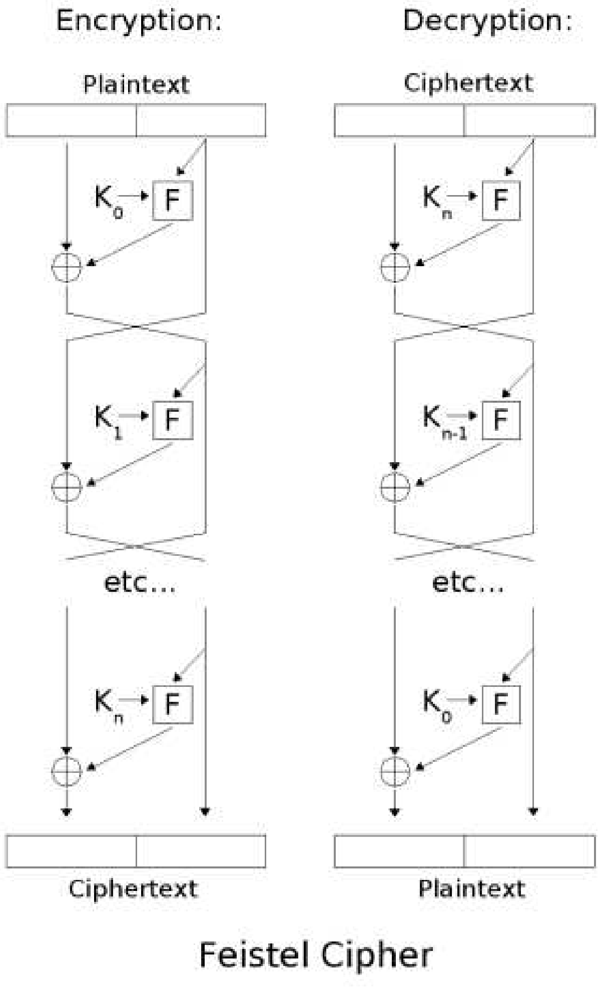

Figure 3.1: Encryption and decryption in feistel cipher (Source: [12])

The construction details are as follows [11]:

Let F be the round function and let K_0, K_1, K_2,…..,K_n be the sub-keys for the rounds 0,1,2,....,n respectively.

Then the basic operation is as follows:

- Split the plaintext block into two equal pieces, (L_0, R_0)
- For each round i = 0, 1, 2…n; compute

$L_{i+1} = R_i$

$R_{i+1} = L_i \oplus F(R_i, K_i)$ (where $\oplus$ is XOR operation)

Then the ciphertext is (R_{n+1}, L_{n+1})

- Decryption of a ciphertext (R_{n+1}, L_{n+1}) by computing for i=n, n-1,…, 0

  $R_i = L_{i+1}$

  $L_i = R_{i+1} \oplus F(L_{i+1}, K_i)$ (where $\oplus$ is XOR operation)

  Then plaintext is (L_0, R_0).

One advantage of the Feistel model compared to a substitution-permutation network is that the round function F does not have to be invertible.

3.1.4 Substitution-permutation Network:

Substitution-permutation network (also referred to as SPN or SP-network), is a series of linked mathematical operations used in block cipher algorithms.

According to [13], SPN network takes a block of the plaintext and the key as inputs, and applies several alternating "rounds" or "layers" of substitution boxes (S-boxes) and permutation boxes (P-boxes) to produce the ciphertext block (see Sections 3.1.1, 3.1.2) . The S-boxes and P-boxes transform (sub-)blocks of input bits into output bits. It is common for these transformations to be operations that are efficient to perform in hardware, such as exclusive or (XOR) and bitwise rotation. The key is introduced in each round, usually in the form of "round keys" derived from it.

Decryption is done by simply reversing the process (using the inverses of the S-boxes and P-boxes and applying the round keys in reversed order).

3.1.5 Simple XOR:

XOR is exclusive-or operation: '^' in C or $\oplus$ in mathematics.

$0 \oplus 0$	0
$0 \oplus 1$	1
$1 \oplus 0$	1
$1 \oplus 1$	0

Table 3.1: XOR operation

2 notable properties of XOR operation are:

$a \oplus a = 0$

$a \oplus b \oplus b = a$

RESEARCH METHODOLOGY

CHAPTER 4
RESEARCH METHODOLOGY

4.1 Approach:

The activity started with careful study of specifications, variations, known cryptanalysis, workings etc. of 5 symmetric block cipher algorithms (Blowfish, Camellia, CAST-128, DES and IDEA). During course of this investigation, notable design guidelines were collected. This section presents brief details of aforementioned algorithms. For a comprehensive list of inferred design decisions, see Section 6.2.

4.2 Investigation of Symmetric Block Cipher Algorithms:

4.2.1 Blowfish:

- Blowfish is a keyed, symmetric block cipher, designed in 1993 by Bruce Schneier [14]. Blowfish has a 64-bit block size and a variable key length from 8 up to 448 bits. It is a 16-round Feistel cipher and uses large key-dependent S-boxes. It is similar in structure to CAST-128, which uses fixed S-boxes.
- Figure 4.1 shows the action of Blowfish. Each line represents 32 bits. The algorithm keeps two subkey arrays: the 18-entry P-array and four 256-entry S-boxes. The S-boxes accept 8-bit input and produce 32-bit output. One entry of the P-array is used every round, and after the final round, each half of the data block is XORed with one of the two remaining unused P-entries.
- Figure 4.2 shows Blowfish's F-function. The function splits the 32-bit input into four eight-bit quarters, and uses the quarters as input to the S-boxes. The outputs are added modulo 232 and XORed to produce the final 32-bit output.
- Decryption is exactly the same as encryption, except that P1, P2,..., P18 are used in the reverse order.
- Blowfish's key schedule starts by initializing the P-array and S-boxes with values derived from the hexadecimal digits of pi, which contain no obvious pattern. The secret key is then, byte by byte, cycling the key if necessary, XORed with all the P-entries in order. A 64-bit all-zero block is then encrypted with the algorithm as it stands. The resultant ciphertext replaces P1 and P2. The same ciphertext is then encrypted again with the new subkeys, and P3 and P4 are replaced by the new ciphertext. This continues, replacing

the entire P-array and all the S-box entries. In all, the Blowfish encryption algorithm will run 521 times to generate all the subkeys - about 4KB of data is processed.

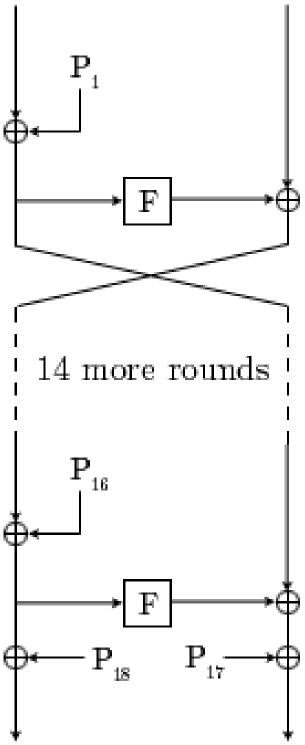

Figure 4.1: Feistel Structure of Blowfish (Source: [15])

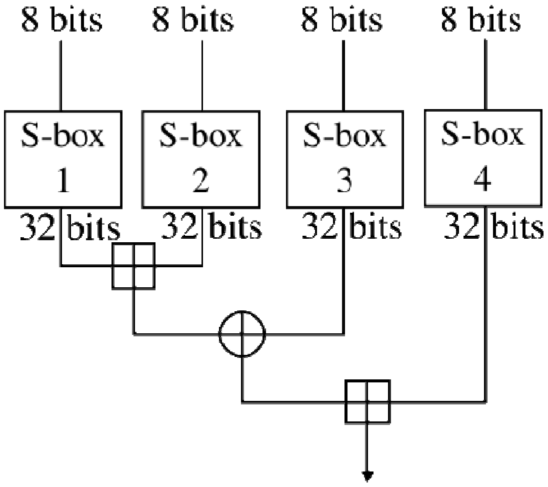

Figure 4.2: The round function (Feistel function) of Blowfish (Source: [16])

4.2.2 Camellia:

- Camellia [17] was jointly developed by Nippon Telegraph and Telephone Corporation and Mitsubishi Electric Corporation in 2000. Camellia specifies the 128-bit block size and 128-, 192-, and 256-bit key sizes. It is a feistel network cipher with 18 or 24 rounds.
- Main structure is similar to DES-like ciphers, not AES i.e. 18-round Feistel structure for 128-bit key and 24-round Feistel structure for 192- and 256-bit keys. The FL/FL-1 function layers are inserted every 6 rounds.
- **Main Components**: Every six rounds, a logical transformation layer is applied: the so-called "FL-function" or its inverse. Camellia uses four 8 x 8-bit S-boxes with input and output affine transformations and logical operations. The cipher also uses input and output key whitening. The diffusion layer uses a linear transformation based on an MDS matrix with a branch number of 5. Briefly stating:

  Round function (F-function): Byte-oriented SPN structure.

  FL/FL-1 function layers: Combination of AND, OR, Rotation, and XOR

  Whitening: XOR

  Subkey generation: Intermediate keys are generated from secret key using 2-rounds.

  Feistel structure: Subkeys are created from secret key and intermediate keys using Rotation & Choice technique
- Camellia can be divided into key scheduling part and data randomizing part:

  Key Scheduling Part: In the key schedule part of Camellia, the 128-bit variables of KL and KR are defined as follows. For 128-bit keys, the 128-bit key K is used as KL and KR is 0. For 192-bit keys, the leftmost 128-bits of key K are used as KL and the concatenation of the rightmost 64-bits of K and the complement of the rightmost 64-bits of K are used as KR. For 256-bit keys, the leftmost 128-bits of key K are used as KL and the rightmost 128-bits of K are used as KR.

  The 128-bit variables KA and KB are generated from KL and KR which are then used to generate final 64-bit subkeys.

Data Randomizing Part:

Encryption: Encryption is performed using an 18-round (for 128-bit keys) and 24 round (192- and 256-bit keys) Feistel structure with FL- and FLINV-functions inserted every 6 rounds. This step generates 128 bit ciphertext from 128 bit input plaintext.

Decryption: The decryption procedure of Camellia can be done in the same way as the encryption procedure by reversing the order of the subkeys.

- **Components of Camellia**:

F-function: F-function takes two parameters. One is 64-bit input data F_IN. The other is 64-bit subkey KE. F-function returns 64-bit data F_OUT. This step includes camellia's S-boxes.

FL- and FLINV-functions: FL-function takes two parameters. One is 64-bit input data FL_IN. The other is 64-bit subkey KE. FL-function returns 64-bit data FL_OUT. FLINV-function is the inverse function of the FL-function.

4.2.3 CAST-128:

- CAST-128 (alternatively CAST5) is a symmetric block cipher. It was created in 1996 by Carlisle Adams and Stafford Tavares using the CAST design procedure; another member of the CAST family of ciphers, CAST-256 was later derived from CAST-128. It is a DES-like Substitution-Permutation Network (SPN) cryptosystem which appears to have good resistance to differential cryptanalysis, linear cryptanalysis, and related-key cryptanalysis. This cipher also possesses a number of other desirable cryptographic properties, including avalanche, Strict Avalanche Criterion (SAC), and an absence of weak and semi-weak keys.
- CAST-128 is a 12- or 16-round Feistel network with a 64-bit block size and a key size of between 40 to 128 bits (but only in 8-bit increments). The full 16 rounds are used when the key size is longer than 80 bits. Components include large 8×32-bit S-boxes, key-dependent rotations, modular addition and subtraction, and XOR operations. There are three alternating types of round function, but they are similar in structure and differ only in the choice of the exact operation (addition, subtraction or XOR) at various points.

- **Description**: The full encryption algorithm is given in the following four steps [19]:

  INPUT: plaintext m1...m64; key K = k1...k128.

  OUTPUT: ciphertext c1...c64.

  - (key schedule) Compute 16 pairs of subkeys {Kmi, Kri} from K
  - (L0,R0) <-- (m1...m64). (Split the plaintext into left and right 32-bit halves L0 = m1...m32 and R0 = m33...m64.)
  - (16 rounds) for i from 1 to 16, compute Li and Ri as follows:

    Li = Ri-1;

    Ri = Li-1 ^ f(Ri-1,Kmi,Kri)

    (f is of Type 1, Type 2, or Type 3, depending on i).
  - c1...c64 <-- (R16,L16). And exchange final blocks L16, R16 and concatenate to form the ciphertext.

  Decryption is identical to the encryption algorithm given above, except that the rounds (and therefore the subkey pairs) are used in reverse order to compute (L0,R0) from (R16,L16).

4.2.4 Data Encryption Standard (DES):

The Data Encryption Standard (DES) [20] is a block cipher that was selected by the National Bureau of Standards as an official Federal Information Processing Standard (FIPS) for the US. It is based on a symmetric-key algorithm that uses a 56-bit key.

- DES is the archetypal block cipher — an algorithm that takes a fixed-length string of plaintext bits and transforms it through a series of complicated operations into another ciphertext bitstring of the same length. In the case of DES, the block size is 64 bits. DES also uses a key to customize the transformation, so that decryption can supposedly only be performed by those who know the particular key used to encrypt. The key consists of 64 bits; however, only 56 of these are actually used by the algorithm. Eight bits are used solely for checking parity, and are thereafter discarded. Hence the effective key length is 56 bits.

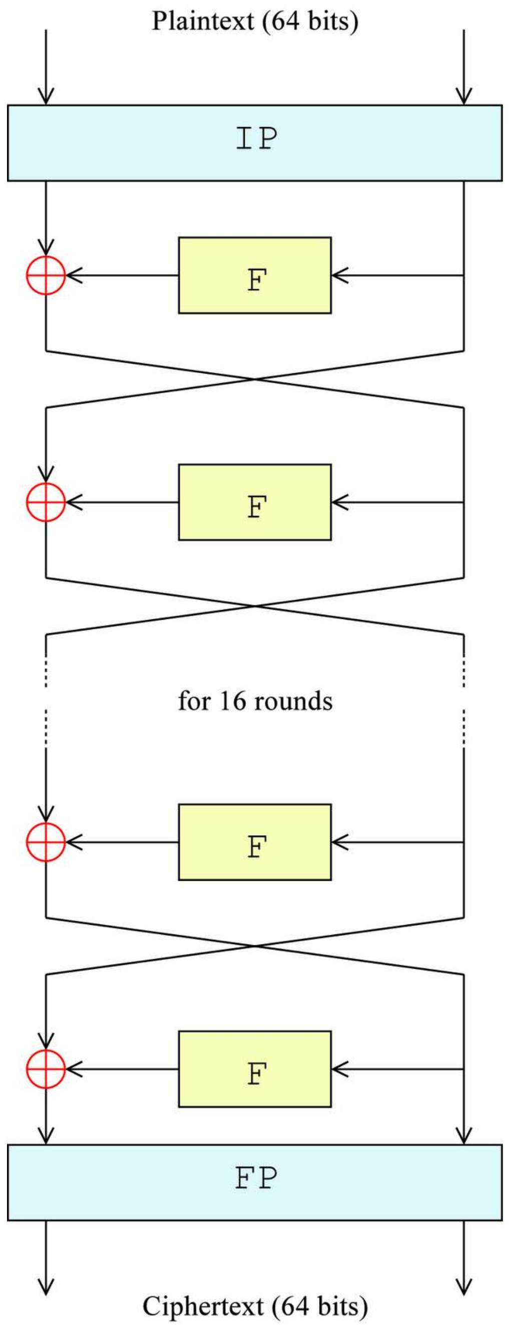

Figure 4.3: The overall Feistel structure of DES

- The algorithm's overall structure is shown in Figure 4.3. There are 16 identical stages of processing, termed rounds. There is also an initial and final permutation, termed IP and FP, which are inverses (IP "undoes" the action of FP, and vice versa). IP and FP have almost no cryptographic significance.

Before the main rounds, the block is divided into two 32-bit halves and processed alternately; known as the Feistel scheme.

The ⊕ symbol denotes the exclusive-OR (XOR) operation. The F-function scrambles half a block together with some of the key. The output from the F-function is then combined with the other half of the block, and the halves are swapped before the next round. After the final round, the halves are not swapped; this is a feature of the Feistel structure which makes encryption and decryption similar processes.

4.2.5 International Data Encryption Algorithm (IDEA):

- International Data Encryption Algorithm (IDEA) is a symmetric block cipher designed by James Massey and Xuejia Lai. IDEA was originally called IPES (Improved Proposed Encryption Standard).
- **Description**: According to [21], IDEA operates on 64-bit blocks using a 128-bit key, and consists of a series of eight identical transformations (a round, see the illustration) and an output transformation (the half-round).

  The processes for encryption and decryption are similar. IDEA derives much of its security by interleaving operations from different groups — modular addition and multiplication, and bitwise eXclusive OR (XOR) — which are algebraically "incompatible" in some sense. In more detail, these operators, which all deal with 16-bit quantities, are:

  - Bitwise eXclusive OR.
  - Addition modulo 216.
  - Multiplication modulo 216+1, where the all-zero word (0x0000) is interpreted as 216.
- **Key Schedule**: Each round uses six 16-bit sub-keys, while the half-round uses four, a total of 52 for 8.5 rounds. The first eight sub-keys are extracted directly from the key, with K1 from the first round being the lower sixteen bits; further groups of eight keys are created by rotating the main key left 25 bits between each group of eight. This means that it is rotated less than once per round, on average, for a total of six rotations. The very simple key

schedule makes IDEA subject to a class of weak keys; some keys containing a large number of 0 bits produce weak encryption.

SCOPE OF THE WORK

CHAPTER 5

SCOPE OF THE WORK

5.1 Scope

This report covers investigation of cryptography theory, description of 5 symmetric block cipher algorithms and lists design decisions/guidelines (desirable for cryptographic algorithms) inferred from this investigation. This work may be useful in several contexts:

- Designing ‘secure’ symmetric block cipher algorithms.
- This can serve as a reference text for cryptographic concepts and constructs, symmetric ciphers and design decisions (which act as guidelines in designing secure algorithms).
- Designing symmetric cipher algorithms that suit different applications (like bulk encryption, random bit generation, hashing etc.) by choosing proper subset of design decisions and guidelines (as listed in this text).
- Designing symmetric cipher algorithms that are implementable on different platforms (like microcontrollers, microprocessors, VLSI hardware etc.) by choosing proper subset of design decisions and guidelines (as listed in this text).

RESULTS AND DISCUSSIONS

CHAPTER 6

RESULTS AND DISCUSSIONS

6.1 Comparison of Symmetric Block Cipher Algorithms:

A comparison of 5 symmetric block cipher algorithms that were investigated in Section 4.2:

Property	Blowfish	Camellia	CAST-128	DES	IDEA
Key length	8–448 bits in steps of 8 bits	128, 192 or 256 bits	40 to 128 bits	56 bit	128 bits
Variable key length	Yes	Yes	Yes	No	No
Type	Feistel	Feistel	Feistel	Balanced feistel	SPN
Block Size	64 bits	128 bit	64 bits	64 bit	64 bits
Avalanche	Yes	Yes	Yes	Yes	Yes
Coding effort needed	Comparatively easy	Comparatively easy	Comparatively easy	Comparatively difficult	Comparatively easy
Weak keys	Yes	Probably yes	Probably yes	Yes	Yes
S box	Yes	Yes	Yes	Yes	Yes
Precomputable subkeys	Yes	Yes	Yes	Yes	Yes
Rounds	16 (feistel)	18 or 24 (feistel)	12 or 16 (feistel)	16 (feistel)	8.5
Current state	Secure	Secure	Insecure	Highly insecure	Secure

6.2 Design Decisions Inferred:

The investigation carried out under this project led me to some design decisions for cryptographic algorithms. A comprehensive list of design decisions (most of these are not new as almost all block ciphers include a subset of these) that were inferred from the study of symmetric block cipher algorithms (see Section 4.2) are as follows:

6.2.1 Based on Area of Application [14]:

A standard encryption algorithm must be suitable for many different applications:

- Bulk encryption. The algorithm should be efficient in encrypting data files or a continuous data stream.
- Random bit generation. The algorithm should be efficient in producing single random bits.
- Packet encryption. The algorithm should be efficient in encrypting packet-sized data. (An ATM packet has a 48- byte data field.) It should implementable in an application where successive packets may be encrypted or decrypted with different keys.
- Hashing. The algorithm should be efficient in being converted to a one-way hash function.

6.2.2 Based on Platforms [14]:

A standard encryption algorithm must be implementable on a variety of different platforms, each with their own requirements. These include:

- Special hardware. The algorithm should be efficiently implementable in custom VLSI hardware.
- Large processors. While dedicated hardware will always be used for the fastest applications, software implementations are more common. The algorithm should be efficient on 32-bit microprocessors with 4 kbyte program and data caches.
- Medium-size processors. The algorithm should run on microcontrollers and other medium-size processors, such as the 68HC11.
- Small processors. It should be possible to implement the algorithm on smart cards, even inefficiently.

6.2.3 Additional Design Guidelines [14, 2]:

- The algorithm should be simple to code. Experiences with DES (see Section 4.2.4 for brief description of DES) show that programmers will often make implementation mistakes if the algorithm is complicated. If possible, the algorithm should be robust against these mistakes.
- The algorithm should have a flat keyspace, allowing any random bit string of the required length to be a possible key. There should be no weak keys.
- The algorithm should facilitate easy key-management for software implementations. Software implementations of DES generally use poor key management techniques. In particular, the password that the user types in becomes the key. This means that although DES has a theoretical keyspace of 256, the actual keyspace is limited to keys constructed with the 95 characters of printable ASCII. Additionally, keys corresponding to words and near words are much more likely.
- The algorithm should be easily modifiable for different levels of security, both minimum and maximum requirements.
- All operations should manipulate data in byte-sized blocks. Where possible, operations should manipulate data in 32-bit blocks.
- Use simple operations that are efficient on microprocessors: e.g., exclusive-or, addition, table lookup, modular- multiplication.
- Employ precomputable subkeys. On large-memory systems, these subkeys can be precomputed for faster operation. Not precomputing the subkeys will result in slower operation, but it should still be possible to encrypt data without any precomputations. Use subkeys that are a one-way hash of the key. This would allow the use of long passphrases for the key without compromising security.
- Consist of a variable number of iterations. For applications with a small key size, the trade-off between the complexity of a brute-force attack and a differential attack make a large number of iterations superfluous. Hence, it should be possible to reduce the number of iterations with no loss of security (beyond that of the reduced key size).

- Consist of a variable number of iterations.
- Algorithm should have a variable length, scalable key, from 32 bits to atleast 256 bits.
- Use a design that is simple to understand. This will facilitate analysis and increase the confidence in the algorithm. In practice, this means that the algorithm will be a Feistel iterated block cipher (see Section 3.1.3).
- If possible, large and key-dependant large S-boxes should be used. Larger S-boxes are more resistant to differential cryptanalysis. Also, while fixed S-boxes must be designed to be resistant to differential and linear cryptanalysis, key-dependent S-boxes are much more resistant to these attacks.
- Permutations should be key dependant.
- Combine operations from different algebraic groups. The IDEA cipher (see Section 4.2.5 for a brief description of IDEA cipher algorithm) introduced this concept, combining XOR mod 216, addition mod 216, and multiplication mod 216+1 [7]. The MMB cipher uses a 32-bit word, and combines XOR mod 232 with multiplication mod 232-1.
- The number of operations required to determine the key from a sample of plaintext and ciphertext should be statistically equal to the product of the operations in an encryption times the number of possible keys. (This means that no plaintext attack should be better than brute force.)
- Knowledge of the algorithm should not defeat the strength of the cipher. All the security should rest in the key. This is basically Kerckhoffs’ principle (see Section 2.4.1).
- A one-bit change of the key should produce a radical change in the ciphertext using the same plaintext, and a 1-bit change of the plaintext should produce a radical change in the ciphertext using the same key. This is the avalanche effect or Shannon’s diffusion. Strictly speaking, strict avalanche criterion and bit independence criteria are desirable (see Sections 2.4.6 and 2.4.5).
- Shannon’s diffusion is desirable from the cipher algorithm.
- The algorithm should contain a noncommutative combination of substitution and permutation.

- The length of the ciphertext should be the same length as the plaintext.
- Redundant bit groups in the plaintext should be totally obscured in the ciphertext. Also, there should be no simple relationships between any possible keys and ciphertext effects.
- Unlike DES, the S-box design criteria should be public.

6.2.4 More Guidelines:

Design criteria specified by Federal Register (NBS) for proposal of a standard cryptographic algorithm which eventually lead to development of DES:

- The algorithm must provide a high level of security.
- The algorithm must be completely specified and easy to understand.
- The security of the algorithm must reside in the key; the security should not depend on the secrecy of the algorithm.
- The algorithm must be available to all users.
- The algorithm must be adaptable for use in diverse applications.
- The algorithm must be economically implementable in electronic devices.
- The algorithm must be efficient to use.
- The algorithm must be able to be validated.

The algorithm must be exportable.

CONCLUSIONS

CHAPTER 7

CONCLUSIONS

The design decisions inferred (see Section 6.2) under this project can be used to design 'secure' cryptographic algorithms.

Also, the inferred design guidelines allow one to vary different factors affecting algorithm's:

- portability to different platforms.
- suitability for different applications.
- modifiability for different levels of security, both minimum and maximum requirements.

All the files related to this project are available at http://amberj.devio.us/projects/crypto/ .

REFERENCES

&

BIBLIOGRAPHY

REFERENCES

1. Bruce Schneier, “Why Cryptography Is Harder Than It Looks”, http://www.schneier.com/essay-037.html
2. Bruce Schneier, Applied Cryptography (2e), John Wiley & Sons, 1996
3. http://en.citizendium.org/wiki?title=Kerckhoffs%27_Principle&oldid=100677206
4. http://gdp.globus.org/gt4-tutorial/multiplehtml/images/security_concepts_symmetric.png
5. http://gdp.globus.org/gt4-tutorial/multiplehtml/images/security_concepts_asymmetric.png
6. C.E. Shannon, "A Mathematical Theory of Communication", Bell System Technical Journal, vol. 27, pp. 379–423, 623-656, July, October, 1948
7. Shannon, Claude. "Communication Theory of Secrecy Systems", Bell System Technical Journal, vol. 28(4), page 656–715, 1949.
8. http://en.wikipedia.org/w/index.php?title=One-time_pad&oldid=425759980
9. http://en.wikipedia.org/w/index.php?title=Confusion_and_diffusion&oldid=389143741
10. http://en.wikipedia.org/w/index.php?title=Avalanche_effect&oldid=422317482
11. http://en.wikipedia.org/w/index.php?title=Feistel_cipher&oldid=417952543
12. http://upload.wikimedia.org/wikipedia/commons/f/f8/Feistel_cipher_diagram.png
13. http://en.wikipedia.org/w/index.php?title=Substitution-permutation_network&oldid=414960026
14. http://www.schneier.com/paper-blowfish-fse.html
15. http://upload.wikimedia.org/wikipedia/en/3/34/BlowfishDiagram.png
16. http://upload.wikimedia.org/wikipedia/en/2/22/BlowfishFFunction.svg
17. http://info.isl.ntt.co.jp/crypt/eng/camellia/index.html
18. http://www.ietf.org/rfc/rfc3713.txt
19. http://www.thc.org/root/docs/cryptography/rfc2144.txt.html
20. http://en.wikipedia.org/wiki/Data_Encryption_Standard
21. http://en.wikipedia.org/wiki/International_Data_Encryption_Algorithm

BIBLIOGRAPHY

1. http://en.wikipedia.org/wiki/Cryptography
2. http://en.wikipedia.org/wiki/Public-key_cryptography
3. http://en.wikipedia.org/wiki/Symmetric_key_algorithms
4. http://en.wikipedia.org/wiki/Code_%28cryptography%29
5. http://en.wikipedia.org/wiki/Information_security
6. http://en.wikipedia.org/wiki/Ciphertext
7. http://en.wikipedia.org/wiki/Kerckhoffs%27s_Principle
8. http://en.wikipedia.org/wiki/Public-key_cryptography
9. http://petitcolas.net/fabien/kerckhoffs/, electronic version and English translation of journal article by Auguste Kerckhoffs in “La cryptographie militaire"
10. http://en.wikipedia.org/wiki/Information_theory
11. http://en.wikipedia.org/wiki/S-box
12. http://en.wikipedia.org/wiki/Transposition_cipher
13. http://en.wikipedia.org/wiki/Substitution_cipher
14. http://en.wikipedia.org/wiki/Permutation_box
15. http://en.wikipedia.org/wiki/Exclusive_or
16. http://en.wikipedia.org/wiki/Block_cipher
17. http://en.wikipedia.org/wiki/Stream_cipher
18. http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf