Adaptive Encryption Techniques In Wireless Communication Channels With Tradeoffs Between Communication Reliability And Security

Walid Al Zibideh

University of Mississippi

Adaptive Encryption Techniques in Wireless Communication Channels with Tradeoffs between Communication Reliability and Security

A DISSERTATION
SUBMITTED TO THE FACULTY OF
THE UNIVERSITY OF MISSISSIPPI
IN PARTIAL FULFILLMENT OF THE
REQUIRMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
IN ELECTRICAL ENGINEERING

Walid Yousef Al Zibideh

May 2013

Abstract

Encryption is a vital process to ensure the confidentiality of the information transmitted over an insecure wireless channel. However, the nature of the wireless channel tends to deteriorate because of noise, interference and fading. Therefore, a symmetrically encrypted transmitted signal will be received with some amount of error. Consequently, due to the strict avalanche criterion (SAC), this error propagates during the decryption process, resulting in half the bits (on average) after decryption to be in error. In order to alleviate this amount of error, smart coding techniques and/or new encryption algorithms that take into account the nature of wireless channels are required. The solution for this problem could involve increasing the block and key lengths which might degrade the throughput of the channel. Moreover, these solutions might significantly increase the complexity of the encryption algorithms and hence to increase the cost of its implementation and use.

Two main approaches have been followed to solve this problem, the first approach is based on developing an effective coding schemes and mechanisms, in order to minimize and correct the errors introduced by the channel. The second approach is more focused on inventing and implementing new encryption algorithms that encounter less error propagation, by alleviating the SAC effect. Most of the research done using these two approaches lacked the comprehensiveness in their designs. Some of these works focused on improving the error performance and/or enhancing the security on the cost of complexity and throughput.

In this work, we focus on solving the problem of encryption in wireless channels in a comprehensive way, that considers all of the factors in its structure (error performance, security and complexity). New encryption algorithms are proposed, which are modifications to the Standardized Encryption Algorithms and are shown to outperform the use of the these algorithms in wireless channels in terms of security and error performance with a slight addition in the complexity. We introduce new modifications that improves the error performance for a certain required security level while achieving the highest possible throughput. We show how our proposed algorithm outperforms the use of other encryption algorithms in terms of the error performance, throughput, complexity and is secure against all known encryption attacks. In addition, we study the effect of each round and S-Box in symmetric encryption algorithms on the overall probability of correct reception at the receiver after encryption and the effect on the security is analyzed as well. Moreover, we perform a complete security, complexity and energy consumption analysis to evaluate the new developed encryption techniques and procedures. We use both analytical computations and computer simulations to evaluate the effectiveness of every modification we introduce in our proposed designs.

Acknowledgements

I would like to thank everybody who helped me to finish this journey, my family, my friends, my professors and my advisor Dr. Matalgah for his great help and support he provided me with during my masters and Ph.D studies. I would like to thank my wife Esraa for her patience and support during hard times, all hard problems seemed to be easy when she was beside me. I would like also to thank my little baby Salma, for the joy and happiness she introduced to my life during this long journey. It was impossible to finish this journey without the excessive support I got from my mother, brothers and my sister.

University, Mississippi Walid Yousef Al Zibideh
May 2013

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivations and Contributions

Encryption is an essential process to assure confidentiality because wireless channels are open media to intruders in which they can intercept and alter the content of any transmitted information. Well known standardized encryption algorithms such as the data encryption standard (DES) were designed to achieve security against intruders. To accomplish this, DES (as well as other symmetric encryption algorithms) was designed in such a way to satisfy what is known as the strict avalanche criterion (SAC). The SAC of an algorithm requires that encrypting two slightly different plaintexts (i.e., the data before encryption) with the same key results in two different ciphertexts (i.e., the data after encryption), on average half the bits will be different even though the plaintexts are different by a single bit only. This criterion also requires that decrypting two slightly different ciphertexts with the same key results in two different plaintexts, on average half the bits will be different even though the ciphertexts are different by a single bit only. Any algorithm that satisfy this property does not exhibit any statistical correlation between input and output that an adversary might use in attacks. Thus the algorithm will multiply the bit errors multifolds; i.e., if there is one error in the received ciphertext, there will be many errors in the decrypted plaintext.

The SAC is a desirable property in encryption algorithms, because any plaintexts with slight difference will result in different ciphertexts, hence make the algorithm secure against

Chosen-Plaintext (CP) attackers. The S-Box in DES was designed in such a way to meet the SAC [1], [2]. However, this criterion becomes catastrophic in wireless channels because wireless channel tends to deteriorate because of noise, interference and fading. Therefore, it is clearly noticed that any algorithm that satisfies the avalanche criterion is very sensitive to bit errors. If one bit of the ciphertext is received in error, each bit of the plaintext will have a probability of error equal to 0.5.

This problem attracted a lot of research due to its great impact on wireless applications. Two main approaches were followed to solve this problem. The first approach was based on developing effective coding schemes and mechanisms in the physical layer, in order to correct and minimize the errors caused by the channel [3]-[5]. The second approach is more focused on inventing and implementing new encryption algorithms that encounter less error propagation, and hence alleviating the SAC [6]-[8]. In [6] a new mode of operation is used with the use of DES to transmit the encrypted data over the wireless channel. While in [7], a new encryption algorithm is introduced, which is a modification to the Data Encryption Standard (DES) and is shown to outperform the standard DES when implemented in wireless channels, moreover, the proposed algorithm is shown to outperform the used mode of operation in [6]. In [8], the authors developed an optimized framework for encryption over wireless channels using the Rijndael algorithm. The proposed mechanism in [8] is mainly based on using longer block and key sizes whenever the channel conditions are better.

1.2 Organization of the Dissertation

In the beginning of this dissertation and after presenting the background of the dissertation subject and some related work from the literature in chapter 2, in chapter 3 we evaluate the security enchantment of the new round proposed in M-DES in a more comprehensive way, where we numerically evaluate the probability of a successful attack on the proposed algorithm as well as calculating the time required for such an attack. In addition, we also evaluate the effect of number of rounds and number of S-Boxes on the probability of correct reception assuming different channel conditions (i.e. different signal-to-noise ratio (SNR) values). Moreover, we evaluate the effect of number of rounds and number of S-Boxes on the security level of the encrypted data in DES as well as in the new round proposed in [7]. The remaining part of this dissertation can be classified in two parts. In the first part, we propose two new encryption algorithms that improves the error performance and throughput in the wireless channel. While in the second part, we evaluate the performance of the proposed algorithms in regard to other well-known standardized encryption algorithms, in term of their complexity and their energy consumption.

In chapter 4 we use a similar mechanism as in [8] and apply it to an adaptive variable block size version of the modified DES introduced in [7]. In particular, we propose a variable key and cipher sizes rather than the fixed 136-bit key and 128-bit cipher used in [7]. We use analytical results to show that our proposed optimized algorithm outperforms the use of Rijndael with variable length as well as the 256-bit AES algorithm, in terms of throughput for a fixed required security level. We also study the security of the proposed algorithm by evaluating its strength against applicable attacks. In chapter 5, we propose a new encryption algorithm where we modify the S-Box design in addition to applying a key coded permutation box at the output of DES, where we use an 80-bit key to permutate the encrypted sub-frames. We use hamming(7,4) codes due to their simplicity and effectiveness in error detection and correction. In addition, four bits are required to code four bits of information, which perfectly fits our application. In another design we add another 80-bit key to permutate the coded encrypted blocks to the proposed algorithm output, which increase the confusion of the encrypted information and hence enhance the security of the algorithm. Using simulation and numerical results, we show that by using KBCP-M-DES we experience more reliable data transmission compared to AES and DES. In chapter 6, we introduce a comprehensive analytical complexity analysis for the proposed algorithms as well as for well-known standardized encryption algorithms. The complexity analysis of this class of encryption algorithms (Symmetric Encryption Algorithms) had attracted many researchers, who developed and used different methodologies to analyze and compute the complexity [9]-[11]. However, those methodologies lacked the comprehensiveness in their analysis, due to the ignorance of the memory access time, which greatly affects the performance. In our analysis, we use a comprehensive methodology where we consider the memory access time required by the algorithm unlike other methodologies. We show that our proposed algorithms outperforms AES and 3-DES in terms of the complexity while it adds a slight addition to the complexity of DES which is considered insecure. In chapter 7, we use a combination of two different approaches to compute the energy of the encryption and decryption process on a certain microprocessor, and to use the bit-error-rate results computed at the receiver to measure the energy savings from one algorithm to the other. In this chapter we first analyze the energy consumption of the selected encryption algorithms based on their hardware complexities. In other words, we calculate the total number of required instructions (i.e, arithmetic, logic and memory access instructions) to encrypt one or more blocks of data. Based on the hardware used for encryption, we can calculate the energy consumed by each instruction using that particular hardware. Hence, we can calculate the total energy consumed by the encryption algorithm on any particular hardware. Next, we analyze the energy savings of each encryption algorithm in regards to the no encryption case, based on the SNR savings for a certain bit error rate (BER) requirement. This is an excellent indicator of the energy saving in a wireless channel environment as shown in [12]-[18].

As a result, in this dissertation we propose new encryption algorithms that can be used in wireless application to improve the error performance and enhance the security. In addition, we introduce comprehensive evaluation methods to show that our proposed algorithms outperform other secure well-known encryption algorithms in terms of their error performance, security, throughput, complexity and energy consumption.

Chapter 2

Background and Related Work

2.1 Chapter Overview

In this chapter we introduce the main background elements of cryptography, but we focus on those elements that are more applicable in wireless communication. Moreover, we present related work from the literature and point out to the trade-offs that these works encounters.

2.2 Classification of Cryptosystems

Cryptosystems are classified based on their encryption mechanism, which basically depends on the number of keys used for encryption, the secrecy of those keys, and how they are handled. In the following we state the two main classes of cryptosystems and their properties. However, in the rest of this chapter as well as most of this dissertation, we will be interested in symmetric cryptosystems.

2.2.1 Symmetric Cryptosystems

Symmetric Cryptosystems are a class of encryption algorithms that use one key for both encryption and decryption. This class is also known as private cryptosystems, because the key used in encryption and decryption is kept secret between the communication parties [19]. The key is exchanged and managed using known algorithms that deal with the key initiation and exchange process, such as diffie-helman, SHA-1, etc. [19], [20]. Fig. 2.1 shows the general design of symmetric cryptosystems, by which Alice uses the key shared between her and Bob to encrypt a message and send it to him. Upon receiving the encrypted message, Bob decrypts it using the same key Alice used for encryption.

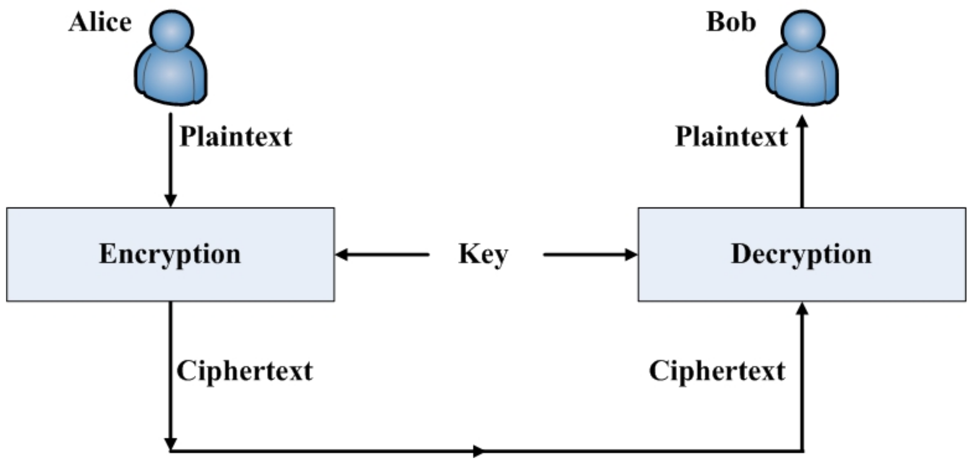


Figure 2.1. Symmetric (Private) key cryptography

2.2.2 Asymmetric Cryptosystems

The asymmetric cryptosystems are called asymmetric because they use two distinct keys for encryption and decryption, unlike symmetric cryptosystems. The encryption is done with the public key of the intended receiver of the message, while the decryption is performed by the private key of the receiver [19], [20]. Assuming Alice wants to send an encrypted message to Bob over the network, Alice needs to encrypt the message with the public key she obtained through Bob's broadcasting. When Bob receives the encrypted message, he can decrypt it using his private key that he kept as a secret and didn't share it on the network. Therefore, each user on the network should generate two keys, one key being called the public key which he broadcasts over the network, so that any user that wants to send him a message will use this key to do so. The other generated key will be called the private key, which the user keeps as a secret for himself, so that he will be able to decrypt any message that was originally encrypted with his public key. Fig. 2.2 explains the general idea of asymmetric encryption. This kind of encryption is also known as a public cryptosystem because each user shares one of his keys with the rest of the network users.

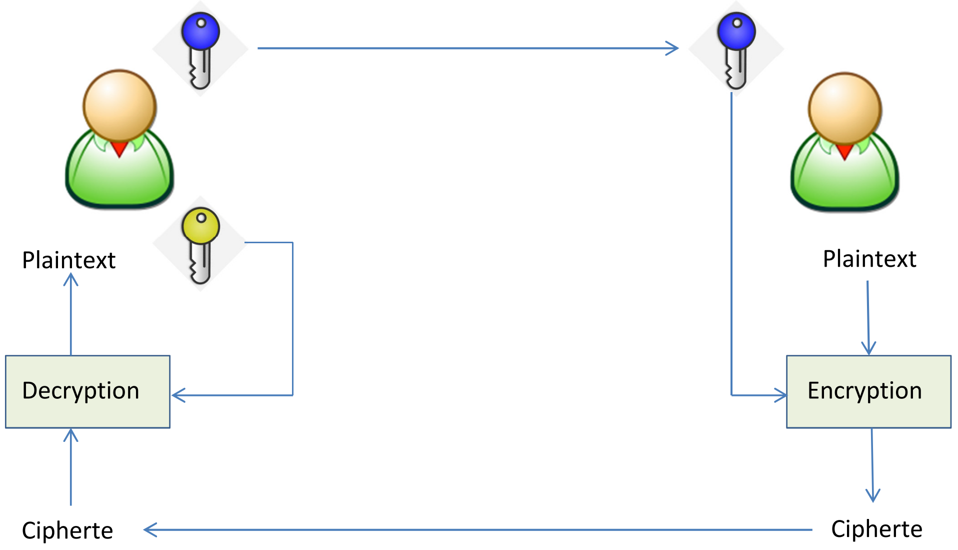


Figure 2.2. Asymmetric (Public) key cryptography

2.3 Block Ciphers

Block ciphers are ciphers that are symmetric (same key for encryption and decryption) and work on a fixed length block of bits. Those ciphers take an input of a certain bit length and produce a ciphertext of the same, longer, or shorter length. These ciphers use a secret key of a fixed bit length. The same key is used for encryption and decryption. Lucifier is the first known block cipher and was designed by IBM in 1970 [19].

2.3.1 Block Cipher Design

Substitution and Permutation

Substitution is an encryption method by which units of the plaintext are replaced with ciphertext according to a regular system or a table. The unit may be a single bit, letter, pairs of bits or letters, or more. The inverse substitution is performed at the receiver side to obtain the original text. There are different types of substitution methods such as the simple substitution cipher by which the algorithm operates on a single bit or letter. Another type is called the polygraphic substitution, by which the algorithm operates on a large group of letters or bits. A substitution cipher can be also classified in terms of it being fixed or not. For example, the monoalphabetic cipher uses fixed substitution all over the algorithm. On the other hand, the polyalphabetic cipher uses a number of substitutions throughout the encryption process, therefore the same input may be mapped to different outputs at different stages or phases of the encryption process.

Permutation is an encryption method by which units of the plaintext are rearranged in some new order. The number of ways that n distinct letters can be reordered is $n!$. Some cipher permutations have the same size as the plaintexts while others are longer or shorter in size. A Longer cipher implies that some bits in the plaintext were copied, while a shorter cipher implies that some bits of the plaintext are dropped. Substitution and Permutation are both important in the design of a block cipher to insure confusion and diffusion, where confusion is the property of making the relationship between the key and the resulting cipher as complex and involved as possible. Diffusion means that the ciphertext bits should depend on the plaintext bits in a very complicated way. Both terms were originally defined by Claude E. Shannon [20].

2.3.2 Block Cipher Properties

Diffusion

Diffusion is the property of making the relationship between the plaintext and the ciphertext as complex and involved as possible. In other words, having two plaintexts that are different by one bit only, should result in two completely different ciphertexts. Therefore, each bit of the ciphertext depends on all the bits in the plaintext.

Confusion

Confusion is the property of making the relationship between the key and the ciphertext as complex and involved as possible. The objective of confusion is to make it very hard to find the key even if a large number of plaintext-ciphertext pairs that were encrypted using the same key are available. To achieve this goal, each bit of the ciphertext should depend on all the bits of the key, In other words, changing one bit of the key should change the entire ciphertext, i.e., by changing one bit of the input, each bit of the output have a probability of 0.5 to be changed. The avalanche effect is a great sensor for good and effective randomization inside the algorithm.

Avalanche Effect

The avalanche effect is a desirable property of block cipher encryption algorithms to ensure security. This property guarantees that changing one or more bits of plaintext changes the output significantly. Similarly, by changing one or more bits of the key, the output changes significantly as well. However, the strict avalanche criterion assures that changing one or more bits of the input will change half the bits of the output on average [8].

2.3.3 Block Cipher Modes

In this section we introduce the various modes of operation used for encryption.

Electronic Codebook (ECB) Mode

In this mode each block is encrypted and transmitted separately as shown in Fig. 2.3. Using this mode, encrypting each block separately makes this mode simple and less complex. It also serves as a clean architecture boundary for other modes of encryption [19], [20].

Security

This mode of operation does not add any security to the encrypted data, the security is based totally on the security of the used algorithm only. Because encrypting a plaintext many times with the same key will result in the same ciphertext. The National Institute of Standards and Technology approves the use of the ECB mode with a FIPS-approved encryption algorithm such as AES and 3-DES.

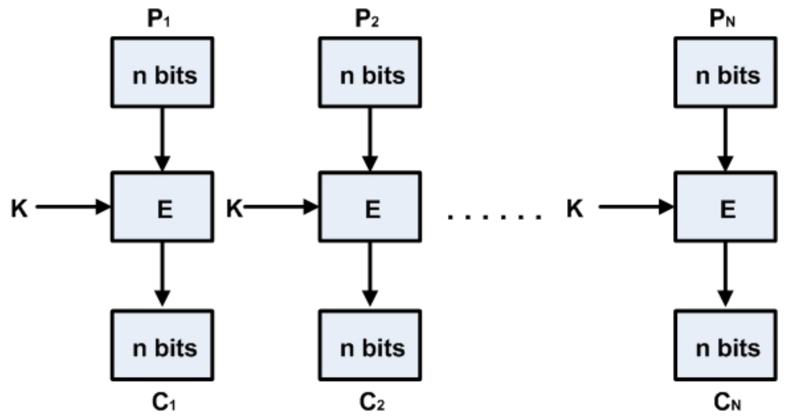


Figure 2.3. Electronic Code Block (ECB)

Complexity

This mode of operation adds no complexity to the encryption process, therefore the number of added operations or clock cycles is zero using the ECB mode.

Error Performance

This mode of operation does not have any effect on the bit error rate. When the encrypted data is transmitted over the channel, the only source for the error is in fact caused by the channel conditions only. The ECB mode does not affect the error performance in any way. This mode is considered the best in terms of bit error rate compared to other modes of operations.

Cipher Block Chaining (CBC) Mode

In this mode the first data block is XORed with a random number called the initial vector (IV) before encryption, the initial vector has the same length as the data block. The first encrypted block is then XORed with the next plaintext block before being encrypted with the same key [19], [20]. The same process is repeated for all blocks as shown in Fig. 2.4.

Security

This mode does not increase the security of the used algorithm significantly. However, this mode can alleviate the need to change the used key occasionally, due to the fact that encrypting a plaintext many times will result in different ciphertexts each time. Because

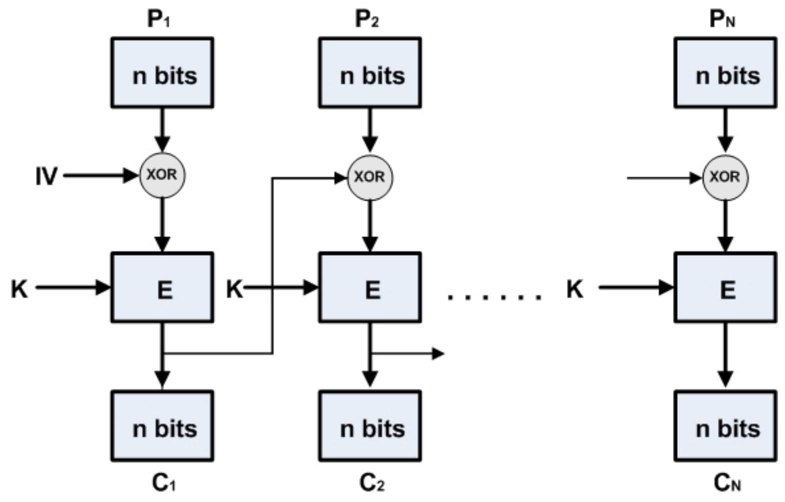


Figure 2.4. Cipher Block Chaining (CBC)

the encryption procedure depends on the previous ciphertext in addition to the key and the random IV. However, this mode does not make the encryption algorithm immune against Chosen-Plaintext Attacks such as the differential cryptanalysis attack, due to the fact that the IV is of the same length as the algorithm's block. Therefore, if the underlying algorithm is insecure against differential cryptanalysis and linear cryptanalysis, the addition of the CBC mode will not make it immune to this attack. On the other hand, if the underlying algorithm is impossible to break using those attacks (i.e., AES is immune to those attacks), the addition of the CBC mode does not degrade nor improve the performance of the algorithm against those attacks. **Complexity**

On the other hand, this mode adds some complexity due to the extra XOR operation before encryption. Assuming that the ciphertext is N bits long, then $N/8$ *byte-wise* additional XOR operations are required. Hence $3 \times \frac{N}{8}$ additonal clock cycles are required to encrypt each block. Assuming DES is used for encryption, 8 additional clock cycles are required, which means that 24 additional clock cycles are required to encrypt each block of information. Although this addition seems to be insignificant, it becomes noticeable when a large amount of data is encrypted.

Error Performance

The problem of this mode is the error propagation. That is, if an error that is imposed by the channel affects one ciphertext, after decryption not only the associated plaintext with the affected ciphertext will be in error, but the next plaintext will be decrypted in error as well. Moreover, due to the avalanche criterion half the bits of both plaintexts will be received in error. Therefore, this mode increases the bit error rate and hence the error performance using this mode of operation will be degraded.

Cipher Feedback (CFB) Mode

In this mode, the initial vector is first encrypted, then the encrypted vector is divided into two parts. Assuming that the block cipher is n bits long and the plaintext is r bits long, the first r bits of the encrypted vector are XORed with the first plaintext block. The second block is encrypted similarly [19], [20], but the initial vector is shifted by r-bits (i.e. the previous ciphertext is shiffted into the current plaintext), successive blocks are encrypted similarly as shown in Fig. 2.5.

Security

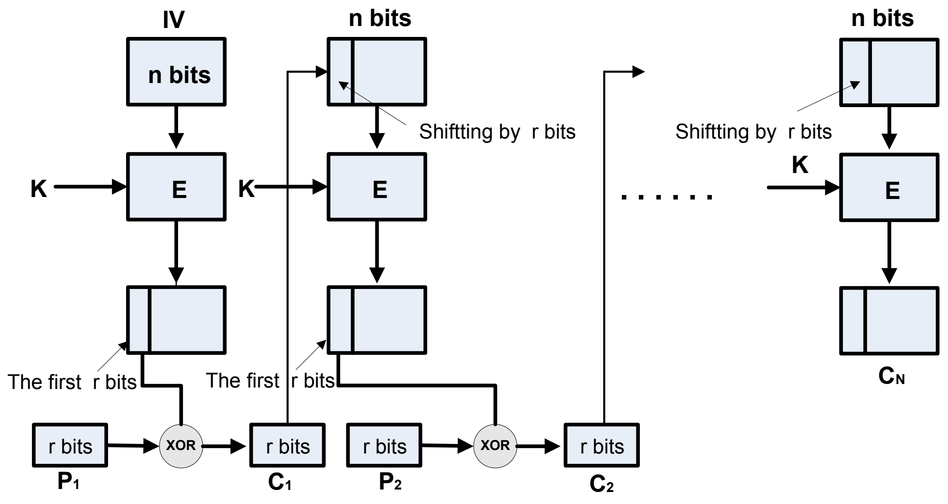


Figure 2.5. Cipher Feedback (CFB)

The security addition of this mode is exactly the same as that of the CBC mode. That is, this mode reduces the need to change the key frequently, due to the fact that encrypting a plaintext with the same key will result in a different ciphertext each time encryption is performed. Moreover, this is a result of having the previous ciphertext as an input to encrypt the next plaintext. However, this mode does not increase the security against chosen-plaintext attacks such as differential cryptanalysis and linear cryptanalysis.

Complexity

This mode of encryption includes shifting the IV by the previous ciphertext before encrypting the IV, then the parts of the encrypted shifted IV are XORed with the current plaintext. Assuming that the plaintext consists of r bits, therefore, $r/8$ *byte-wise* shift operations and $r/8$ XOR operations are required. Hence, $80 \times \frac{r}{8} + 3 \times \frac{r}{8} = 10 \times r + 3 \times \frac{r}{8}$ additional clock cycles are required to encrypt one block of information.

Error Propagation

The error propagation of this mode follows the exact same behavior of the CBC mode.

Output Feedback (OFB) Mode

This mode is exactly the same as the previous mode except that the second initial vector is shifted by the first r bits of the previous encrypted IV. Therefore, the IV for each block is the previous block's IV shifted by r bits each time. Figure 2.6 shows the concept of output feedback [19], [20].

Security

This mode of operation follows the same security characteristics of the CBC and CFB modes.

Complexity

The complexity addition of this mode is similar to that of the CFB mode.

Error Propagation

The error propagation of this mode follows the exact same behavior of the CBC and CFB modes.

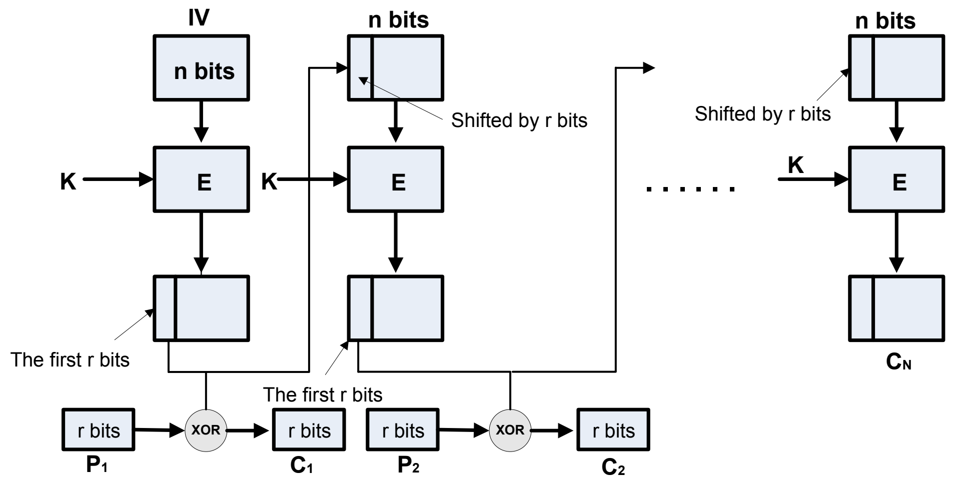


Figure 2.6. Output Feedback (OFB)

Counter (CTR) Mode

Each cipher block is independent of the other blocks in this mode. Each data block has its unique IV which is an increment of the previous IV. The IV is encrypted and XORed with the data block. The IV should be of the same size of the data block. The counter mode is shown in Fig. 2.7.

Security

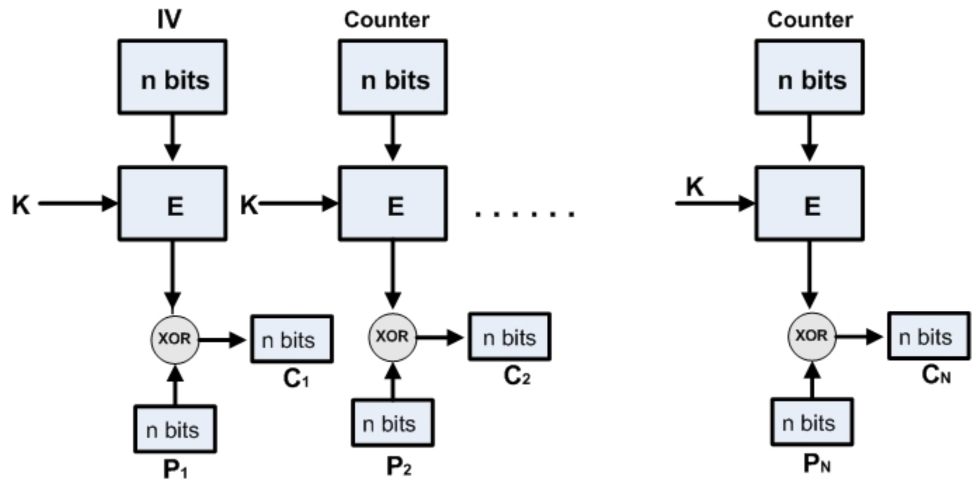


Figure 2.7. Counter (CTR)

This mode adds the same security characteristics that are added by CBC, CBF and OFB modes. That is due to the fact that each plaintext will result in a different ciphertext if it is encrypted more than one time, this is because each time the counter is encrypted with the same key before being XORed with the plaintext. However, the counter is always changing with the time, hence, the value that is XORed with the plaintext is changed at each point of time. Therefore, there is no need to change the key frequently. On the other hand, as the previous modes, this mode does not add any immunity against chosen-plaintext attacks.

Complexity

This mode adds some complexity that is equal to that added by the CBC modes and less than that added by CFB and OFB, because the only addition in this mode is the XOR operation. Hence, $n/8$ *byte-wise* XOR operations are required, which means that $3 \times \frac{n}{8}$ clock cycles are required in this mode of operation.

Error Propagation

The main advantage of this mode is that it does not exhibit any error propagation as that of CBC, CFB and OFB. Although, this mode achieves the same security critiria of the CBC, CFB and the OFB modes, it does not have any effect on the error performance of the encryption algorithm. It also adds less complexity than that added by the CFB and OFB.

Table 1.1 summarizes the difference between different modes of operation and their effect on the security, complexity and error propagation.

Table 2.1. Properties of various modes of operation

Mode	Security	Complexity	Error Propagation
ECB	None	No Addition	No
CBC	Key Security	Slight Addition	Yes
CFB	Key Security	High Addition	Yes
OFB	Key Security	High Addition	Yes
CTR	Key Security	Slight Addition	No

2.4 Cryptanalysis of Block Ciphers

2.4.1 Brute Force Attack

The brute force attack is the basic and most straight forward attack. The attack idea is based on trying all possible keys to decrypt some ciphertext. Each time the attacker will check the obtained plaintext and if it matches his expectation, then that certain key used to obtain this plaintext is the correct key that was used in the encryption process. The following assumptions must be made in order for this attack to work:

1. It is assumed that the attacker knows the algorithm that is used for encryption.
2. It is assumed that the attacker has access to some ciphertext.
3. It is assumed that the attacker knows the type of the data that was encrypted (example: bank accounts information, employees information)

For example, assuming that some encryption algorithm has a key of 4 bits. Then, if an attacker wants to use the brute force attack, it is assumed that he has obtained some ciphertext and knows which algorithm was used. The attack mechanism is based on trying all possible key combinations starting from 0000 up to 1111, which corresponds to $2^4 = 16$ keys. When the plaintext makes sense to the attacker or meets his expectation, then the key that was used to obtain that plaintext is the correct key, and hence the attacker has cracked the algorithm. Therefore, in order for an encryption algorithm to be secure against this type of attacks it should have the following characteristics:

- The primary factor in keeping the algorithm secure against brute force attack is to have a long key such that the intruder can not check all of its possible combination in a feasible time.
- The key should be changed in a certain amount of time that is less than the time it takes an attacker to guess it. For example if the intruder can guess a 10-bit key in 2 seconds,

the key should be changed every 1 second. However, this solution will have a major drawback, which is the key management. Managing a key that is changing every 1 second and managing its transfer between the two or more parties of the communication channel can be exhausting and needs more resources and time. However using a 128-bit key for example is considered secure and there is no need to change it for the current used technology.

- The algorithm itself can be secure, so the attacker will not know how long is the key nor how to do the decryption process. For example, some companies develop and use their own encryption algorithm without revealing it to the outer world. This solution may be reasonable, but it will limit the use of such encryption algorithms to a very narrow scope. For example, the encryption algorithm can not be used for the companies outer communication because nobody other than the company itself knows the algorithm, and hence nobody can decrypt their ciphertexts.

2.4.2 Differential Cryptanalysis

History

Differential Cryptanalysis is a general technique of cryptanalysis applicable primarily to block ciphers. It is based on studying the relationship between the ciphertexts of closely related plaintexts, trying to discover where the attacked algorithm does not explore random behavior. It was introduced as a successful method of attack by Eli Biham and Adi Shamir in the late 1980s, where they published a paper including a theoretical attack that can attack the Data Encryption Standard (DES) and obtain its key, given the fact that 2^{47} pairs of plaintext and ciphertext are available to the attacker and all pairs were encrypted using the same key [21]. Although the attack might be hard to be applicable, one of it shows that there is a possible flow in the design of DES. However, Don Coppersmith, the IBM DES original team members stated later that his team was aware of this attack as was the

National Security Foundation (NSA), but it was kept secret to keep the United States in front of other countries in the science of cryptography, and it was called the T-attack or the Tickle attack within the IBM team. DES was designed with this attack in mind while previous ciphers where not, until Shamir and Biham showed after 13 years that it can be attacked using this attack.

Attack idea

Differential cryptanalysis is usually a chosen plaintext attack. A chosen plaintext attack is an attack based on the ability of the attacker to chose any plaintext and be able to obtain its corresponding ciphertext. The attacker is assumed to be ignorant about the key, so it is the final objective of the attacker to obtain the key using a sufficient number of chosen plaintexts.

The basic method of the differential cryptanalysis attack is based on calculating the difference between pairs of plaintexts δ_x, and the difference between their corresponding ciphertext is analyzed to detect any statistical patterns in their distribution. The difference is usually measured in terms of the XOR operation.

The statistical properties depend primarily on the nature of the used S-Box. Therefore, the attacker uses pairs of differences called differentials (δ_x, δ_y), where δx is the difference between two inputs and δ_y is defined as $\delta_y = S(X \oplus \delta_x) \oplus S(X)$. Therefore, δ_y is the difference between the result of entering X into a specific S-Box, XOR-ed with the result of entering the difference between two inputs (one of them is X) to the same S-Box. Hence, one particular ciphertext difference is expected to be more frequent, so the cipher can be distinguished from the random outputs [22].

Attack Mechanism

The attack basically consists of the following steps:

- The difference between two ciphertexts is observed as a function of the difference between the corresponding plaintexts.

- The highest probability differential input is found.
- The keys are assigned certain probabilities and the most probable key is located.

Each S-Box in DES has a 6-bit input and 4-bit output. Assuming that we have two inputs for one S-box (x, x^*), each of x and x^* have 64 possible values, and hence the tuple (x, x^*) will have $64^2 = 4096$ possible values. As the six bit variables x, x^* and $x' = x \oplus x^*$ vary over their 64 possible values, the four bit variables $y = S(x), y^* = S(X^*)$ and $y' = y \oplus y^*$ vary over their possible 16 values. Therefore, we have 64 values for x' and 16 values for y'. Hence, some of the values for y' are going to be the same.

The distribution of y' can be found by counting the number of times each of the possible values of y' are repeated over varying the values of x'. This process is repeated for each of the eight S-Boxes. Table 1.5 shows the distribution of y' for the first S-Box in DES. x' is a 6-bit values, hence it takes values from 00 to 3F, while y' is a 4-bit value, hence it takes values from 0 to F. The summation of each row is 64, because each input difference x' occurs

Table 2.2. Differentials Distribution table of DES S-Box 1

x'	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	64	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
01	0	0	0	6	0	2	4	4	0	10	12	4	10	6	2	4
02	0	0	0	8	0	4	4	4	0	6	8	6	12	6	4	2
03	14	4	2	2	10	6	4	2	6	4	4	0	2	2	2	0
$\vdots$																
0C	0	0	0	8	0	6	6	0	0	6	6	4	6	6	14	2
$\vdots$																
34	0	8	16	6	2	0	0	12	6	0	0	0	0	8	0	6
$\vdots$																
3E	4	8	2	2	2	4	4	14	4	2	0	2	0	8	4	4
3F	4	8	2	2	2	4	4	14	4	2	0	2	0	8	4	4

for 64 out of the 4096 total cases of (x, x^*). For the first row, $x' = 0$, when $x' = x \oplus x^* = 0$.

Therefore, $y' = 0$, for all the 64 cases where $x = x^*$ ($x' = 0$). Similarly, for the next row for example, there are 64 cases for (x, x^*) where $x' = x \oplus x^* = 01$, but the difference y' varies between 0 to F for all 64 input cases.

- When $x' = 01$, y' will never be 0,1,2, 4 or 8. Therefore the probability of having $y' = 0, 1, 2, 4$ or 8 given that $x' = 01$ is equal to zero.
- The probability of having $y' = 5$ or E, given that $x' = 01$ is $2/64 = 1/32$
- The probability of having $y' = 6, 7, B$ or F, given that $x' = 01$ is $4/64 = 1/16$
- The probability of having $y' = D$, given that $x' = 01$ is $6/64$
- The probability of having $y' = 9$ or C, given that $x' = 01$ is $10/64$
- The probability of having $y' = A$, given that $x' = 01$ is $12/64$

The same process is repeated for all S-Boxes, so that each of the eight S-boxes will have a different differentials distributions table. It is clear that the tables are non uniformly distributed.

Considering the case were x' is 34 in the mapping table of the first S-Box of DES, the possible outputs y' are 1, 2, 3, 4, 7, 8, D and F. The number of occurrences for each of the possible outputs are 8, 16, 6, 2, 12, 6, 8 and 6, respectively.

Determining the key

Assuming that we know that the inputs for the S-Box1 are 01 and 35, the XOR of the two inputs is 34. Both inputs will be XORed with the key before getting to the S-Box. The XOR for the two inputs after the key XOR stage will remain the same. That tells us that the input XOR is independent of the key. The following equation shows the proof.

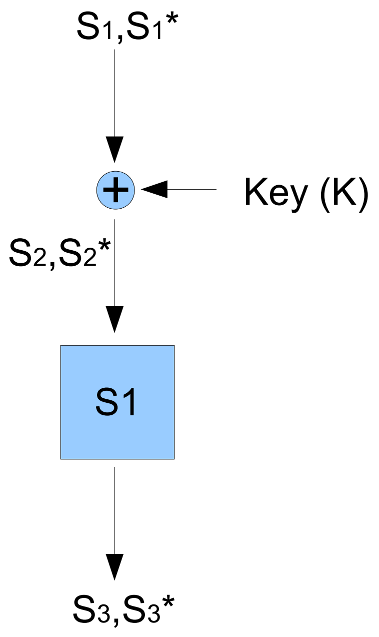


Figure 2.8. XOR operation before and after S-Box1

$$\begin{aligned} S_2' &= S_2 \oplus S_2^* \\ &= (S_1 \oplus K) \oplus (S_1^* \oplus K) \\ &= S_1 \oplus S_1^* = S_1' \end{aligned}$$

where S_1 and S_1^* are the two inputs, while S_1' is the XOR of the two inputs. S_2 and S_2^* are the two inputs after XORing with the key, while S_2' is the XOR of the two inputs after XORing each one of them with the key. S_3 and S_3^* are the two inputs after the S-Box phase, while S_3' is the XOR of the two inputs after each one of them passes by the S-Box. Figure 2.8 shows those stages in detail.

Since $S_1' = S_2'$, then both should be equal to 34. We know that S_1 and S_1^* where 01 and 35 respectively. However, all the possible cases for S_2 and S_2^* are 06,10,16,1C,22,2428 and 32. Given equation 2.4.1, we can notice the following relationship

$$S_2 = S_1 \oplus K$$

Therefore

$$K = S_1 \oplus S_2$$

Hence, we can try to guess the key given this information. Moreover, the key will be one of the XOR results of all possible values of S_1 and S_2.

$$\begin{array}{rr} 06 \oplus 01 = 07 & 06 \oplus 35 = 33 \\ 10 \oplus 01 = 11 & 10 \oplus 35 = 25 \\ 16 \oplus 01 = 17 & 16 \oplus 35 = 23 \\ 1C \oplus 01 = 1D & 1C \oplus 35 = 29 \\ 22 \oplus 01 = 23 & 22 \oplus 35 = 17 \\ 24 \oplus 01 = 25 & 24 \oplus 35 = 11 \\ 28 \oplus 01 = 29 & 28 \oplus 35 = 1D \\ 32 \oplus 01 = 33 & 32 \oplus 35 = 07 \end{array}$$

So the possible key is one of the following 07, 11, 17, 1D, 23, 25, 29, 33. To narrow this set, another pair (which XOR is 34) is taken and the same procedure is repeated. For example, $S_1 = 21$ and $S_1^* = 15$, moreover S_1' and S_2' are both equal to 34. Hence all possible values for S_2 and S_2^* are 01, 02, 15, 21, 35 and 36. By repeating the same process of XORing all possible values of S_1 with all possible values of S_2 to find all possible keys

$$\begin{array}{rr} 01 \oplus 21 = 20 & 01 \oplus 15 = 14 \\ 02 \oplus 21 = 23 & 02 \oplus 15 = 17 \\ 15 \oplus 21 = 34 & 15 \oplus 15 = 00 \\ 21 \oplus 21 = 00 & 21 \oplus 15 = 34 \\ 35 \oplus 21 = 14 & 35 \oplus 15 = 29 \\ 36 \oplus 21 = 17 & 36 \oplus 15 = 23 \end{array}$$

So the possible key is one of the following {20, 23, 34, 00, 14, 17, 29}. However, the correct key must appear in both sets. Therefore, by intersecting {07, 11, 17, 1D, 23, 25, 29, 33} with {20, 23, 34, 00, 14, 17, 29}, the key must be either 17 or 23. To determine which one is the correct key, more pairs of our selection must be available.

2.4.3 Linear Cryptanalysis

History

The linear Cryptanalysis attack was created by Mitsuru Matsui in 1992. The attack was first applied to the fast data Encipherment Algorithm (FEAL) and then to DES. The attack is shown to break DES using 2^{43} known plaintext [23], [24]. The attack is based on finding affine approximations to the action of a cipher.

Attack idea

A known plaintext attack is the ability of the attacker to have a samples of plaintext and their corresponding ciphertexts. The shortcoming of this attack is that the attacker is not able to chose a specific plaintext to obtain its ciphertext. However, if the attacker is lucky, the needed pairs might be included in the sample that he already has. The attack assumes that the attacker has the ability to hack into someone's database to get those pairs or get them by any other way.

The linear cryptanalysis attack is based on trying to construct linear equations to relate the ciphertext with the plaintext and the key. Any linear equation relating the plaintext, ciphertext, and the key should hold with a probability of 1/2 in an ideal cipher. The way the inputs and outputs may be related is using the XOR operation. For example,

$$P_1 \oplus P_5 \oplus C_2 = K_3$$

When the linear approximation equations are obtained, using the given known plaintexts, the key bits can be extracted based on those equations.

2.5 Standardized Well-Known Block Ciphers

2.5.1 Data Encryption Standard (DES)

The Data Encryption Standard DES is one example of symmetric “private” key block ciphers. DES was selected by the National Institute of Standards and Technology to be used in encrypting all governmental documents in 1977. It was standardized as an official Federal information standard (FIPS 46). DES was designed by IBM and the National Security Agency (NSA), and was considered secure until 1999 when it was broken in 22 hours and 15 minutes, due to its small size key [19], [20], [25]. The algorithm was originally designed with a 64-bit input (Plaintext) and a 64-bit output (ciphertext) with a 56-bit key, and consisted of sixteen identical rounds. The general structure of DES encryption is shown in Fig. 2.9.

DES encryption consists of the following phases:

1. Initial Permutation.

2. Rounds 1 through 16.

3. Final Permutation.

Each round needs a different 48-bit key that is generated by the round key generator which takes the 56-bit key as its input. In the following, we discuss each phase in detail, but first the DES round key generator will be introduced.

DES Round Key Generator

The round key generator is an important phase of DES, because it is responsible for generating sixteen unique keys, one key for each round. The input of the DES round key generator is the general 56-bit key and the output is the sixteen 48-bit round keys.

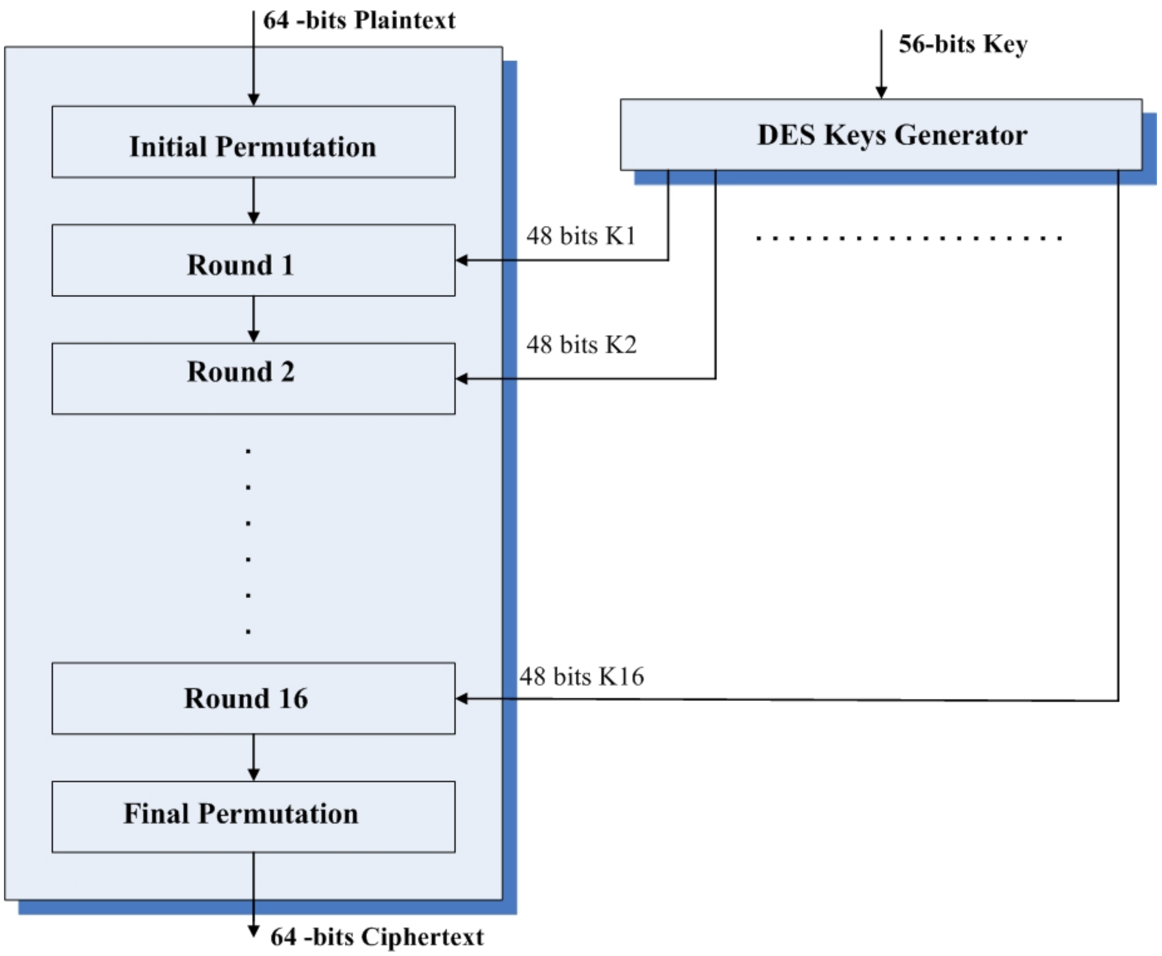
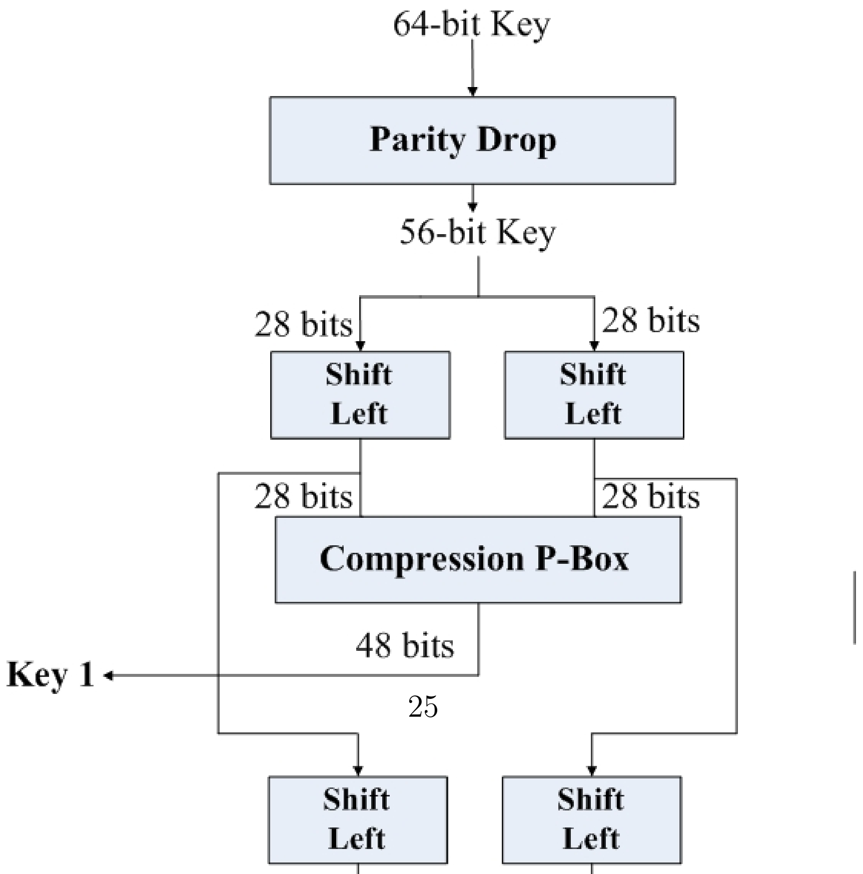


Figure 2.9. DES Encryption



A figure of the DES round key generator is shown in Fig. 2.10, the general 64-bit key is minimized to 56 bits by removing the parity bit of each byte (i.e., bits 8, 16, 24,...64), then a 1-1 permutation is done. The permutation at this phase is changing the bit locations only. To generate the first round key, the 56-bit permutated input will be divided into two parts, each of 28 bits. Each of these 28 bits are shifted to the left by one bit, then both part enter to a compression permutation box (i.e., some bits are dropped after the permutation). The 48-bit output of the compression P-Box is the first round key. The second key is generated by taking the previous input of the compression P-Box and repeating the same process. The same process is repeated to generate all of the sixteen keys.

Permutations

The initial and final permutation phases take a block input of 64 bits and generate an output block of the same size. The permutation is a process of changing the locations of the bits without changing the values of the bits, this process is 1-1 bit mapping as shown in Fig. 2.11. The permutation is done based on a permutation table.

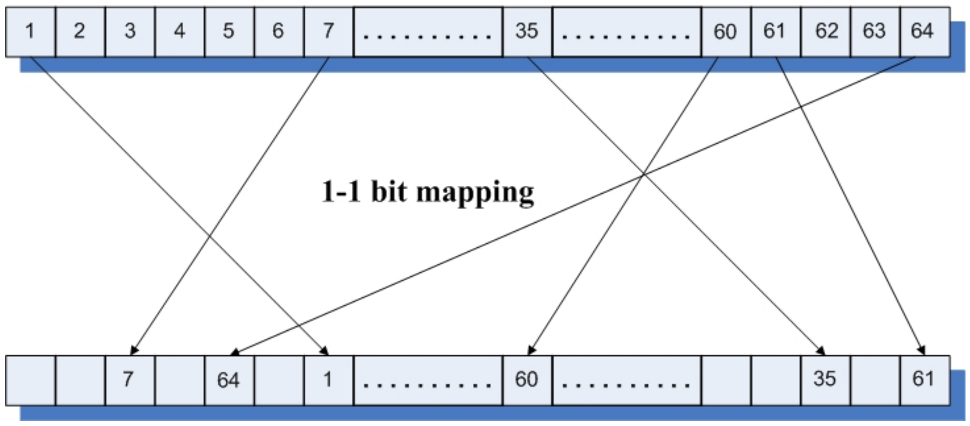


Figure 2.11. Initial and Final Permutation

Rounds 1 through 16

Each round takes a block input of 64 bits that is cascaded from the output of the previous round. Thus the input of the first round is the output of the initial permutation phase, while the output of round 16 is the input for the final permutation phase. Each round requires a unique 48-bit key that is generated by the DES round key generator, which we described earlier. All the 16 rounds have the same function. A block diagram of the DES round structure is shown in Fig. 2.12.

As shown in the figure, at the beginning of each round the 64-bit block is divided into two parts, left and right each, of 32 bits. The right part is taken exactly as the left 32-bit output of the same round, while the left part of the output is XOR-ed with the 32-bit output of the F-Box. The F-Box takes the right 32 bits input of the round and the 48-bit round key as its inputs.

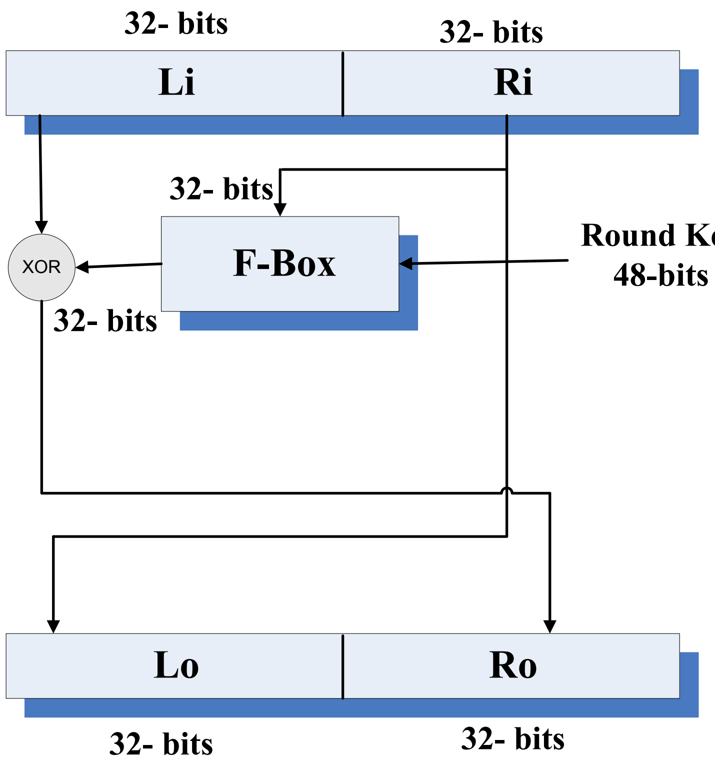


Figure 2.12. DES Round Structure

F-Box

We used the notation "F-Box" to describe the DES function, which is a function of the 48-bit round key and the right 32-bit half of the round input $f(R_i, K_i)$. A block diagram of the

F-Box structure is shown in Fig. 2.13. As shown in the figure the round key is XOR-ed with the output of the expansion box (P-Box), which has the right 32 bits round input as its input and expands it to 48 bits output. The 48-bit result of the XOR operation is the input for the S-Box. The 32-bit output of the S-Box phase is then permutated through 1-1 mapping (the same way as the initial and final permutation phases but with a new permutation table).

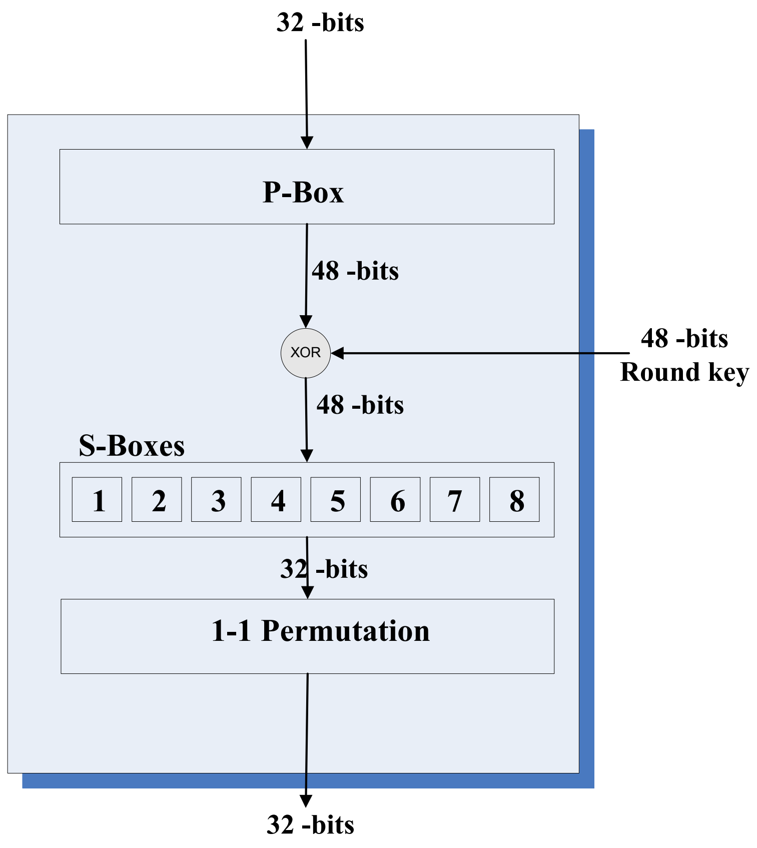


Figure 2.13. F-Box Structure

Expansion (P-Box)

The expansion box is a 32-48 permutation box, in which 1 bit is mapped to one or two bits. The 32-bit input of the P-Box is mapped to a 48-bit output. The 32-bit input frame is divided into eight sub-frames each of four bits, and each sub-frame is mapped into a new sub-frame of six bits, which makes the output frame equal to 48 bits. The P-Box mapping is shown in Fig. 2.14, where the four bits of the input sub-frame are mapped into the four middle bits of the output sub-frame, while the last bit of the previous input sub-frame is mapped to the first bit of the current output sub-frame, and the first bit of the next sub-frame is mapped to the last bit of the current output sub-frame.

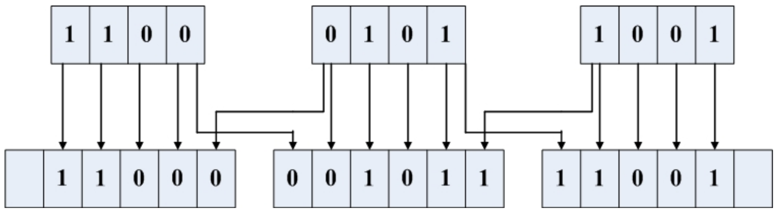
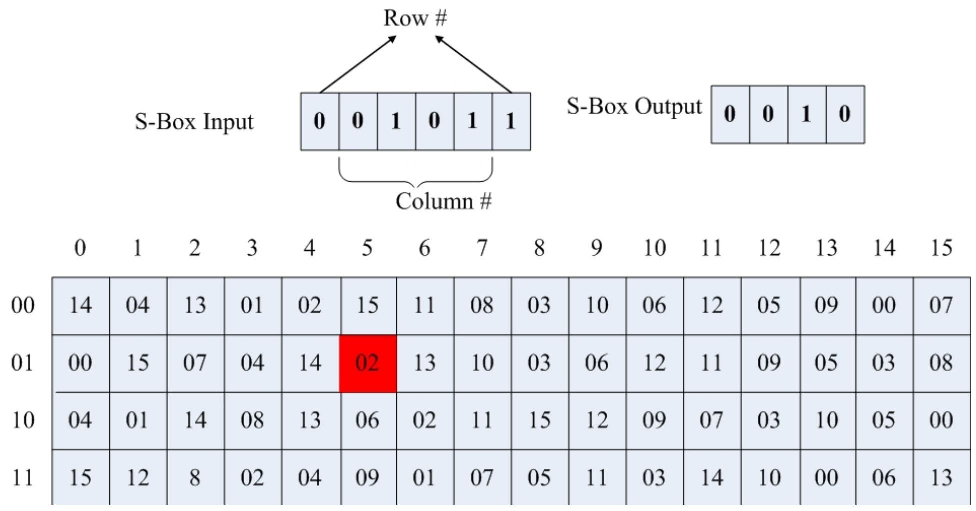


Figure 2.14. P-Box mapping

S-Box

The S-Box phase is the most responsible part for the bit mixing in DES. The 48-bit input frame of the S-Box phase is divided into eight sub-frames each of six bits. Each of the sub-frames is an input to one of the eight S-Boxes. Each S-Box takes an input of six bits and generates an output of four bits, and uses a 4×16 table for the mapping. The tables are efficiently filled, the effect of which is presented in the following chapters. The four middle bits of the six input bits are used as an index for the column in the mapping table and the two bits on the edges are used as an index for the row. Therefore, when any S-Box has an input, the input is mapped to an entry in the table and the value in the table is the output of the S-Box. An example for one of the eight S-Box mappings is shown in Fig. 2.15. In the figure, the input for S-Box number one is 001011. The middle bits are 0101, which refers to the fifth column, while the edge bits are 01 which refers to the second row. So, the output is the entry at the intersection of the fifth column with the second row as shown in the figure.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
00	14	04	13	01	02	15	11	08	03	10	06	12	05	09	00	07
01	00	15	07	04	14	02	13	10	03	06	12	11	09	05	03	08
10	04	01	14	08	13	06	02	11	15	12	09	07	03	10	05	00
11	15	12	8	02	04	09	01	07	05	11	03	14	10	00	06	13

Figure 2.15. S-Box #1 mapping

Table 2.3. Weak Keys

Keys before parities drop (64 bits)	Actual keys (56 bits)
0101 0101 0101 0101	0000000 0000000
1F1F 1F1F 0E0E 0E0E	0000000 FFFFFFF
E0E0 E0E0 F1F1 F1F1	FFFFFFF 0000000
FEFE FEFE FEFE FEFE	FFFFFFF FFFFFFF

Table 2.4. Semi-Weak Keys

Keys before parities drop (64 bits)	Actual keys (56 bits)
01FE 01FE 01FE 01FE	FE01 FE01 FE01 FE01
1FE0 1FE0 0EF1 0EF1	E01F E01F F10E F10E
01E0 01E0 01F0 01F0	E001 E001 F101 F101
1FFE 1FFE 0EFE 0EFE	FE1F FE1F FE0E FE0E
011F 011F 010E 010E	1F01 1F01 0E01 0E01
E0FE E0FE F1FE F1FE	FEE0 FEE0 FEF1 FEF1

Security of DES

The secrecy of any algorithm is measured by the strength of the algorithm against attacks. This means that in order for an encryption algorithm to be secure, the attacker should not have the ability to know the secret encrypted data. Therefore, the attacker should not be able to benefit from having the ciphertext in knowing the original plaintext or guessing the key. The main types of attacks are discussed in the following subsections.

Brute force attack

The brute force attack is an example of a ciphertext only attack in which the attacker tries all possible keys to decrypt the data. At the early stages of DES design, the designers and researchers thought that the DES algorithm would be secure against this attack since the key size is 56 bits; it was thought that trying all the possible keys, which are 2^{56} different keys, would be impossible in a reasonable amount of time given the computer capabilities at that time. Today a regular computer with one processor can check up to a million keys in one second, which means over two thousand years would be required to check all the possible keys. Although a computer with million chips (parallel processing) today can test all the keys in much less than 20 hours, a computer with these capabilities in 1977 would have needed many millions of hours, and as a result the algorithm was considered secure until it was cracked in 1999 in only 22 hours [19], [26].

Weak keys

There are also a number of keys called weak keys because they generate sixteen similar round keys. These keys are shown in Table 1.3. Some keys, called the semi-weak keys, generate only two distinct round keys rather than sixteen, where each of these two keys is repeated eight times. The keys are listed in Table 1.4. Another type of weak key is called a "possible weak key". There are 48 keys that create four distinct round keys rather than sixteen, where each of these keys is repeated four times [19], [27].

Differential Cryptanalysis

Differential Cryptanalysis attack was introduced by Eli Biham and Adi Shamir as a type of known plaintext attack. The attack idea supposes that the attacker gains access to one of the communication party's computers, and gets a number of pairs of plaintexts and their corresponding ciphertexts from the victim's database (Known Plaintext Attack).

In this type of attack, the attacker analyzes the relationship between the different plaintexts and analyzes their propagation within the ciphertexts. In 1991, this attack was successful in breaking the full 16 rounds of DES using 2^{47} pairs with a probability of $2^{-47.2}$ and a complexity of 2^{37} encryption and decryption operation [21], [28].

2.5.2 Triple DES (3-DES)

General Design

To solve the problem of security in DES, the 3-DES algorithm was proposed. The new algorithm decrypted the original ciphertext of DES using a new 56-bit key and again encrypted the result with a new 56-bit key. Therefore, three stages with three different keys are introduced. The total key size is hence 168 bits long, which makes it secure against brute force attack. Fig. 2.16 shows the general design of the 3-DES algorithm.

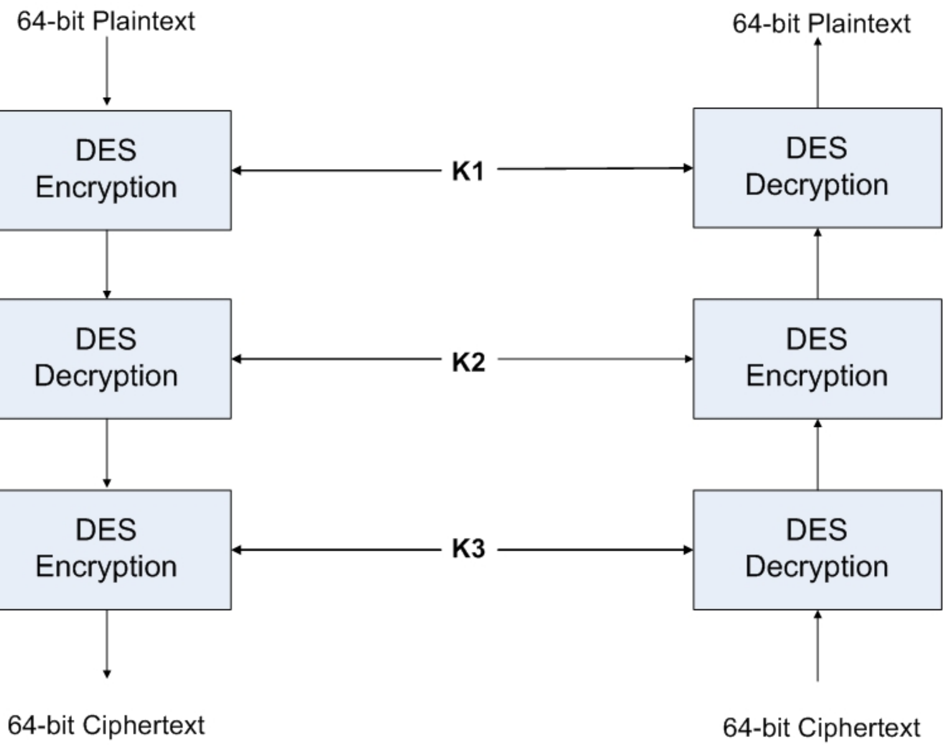


Figure 2.16. 3-DES General Design

In some cases where a lower security level is required, the first and third key can be the same, therefore the total key length is 112 bits in this case.

Security of 3-DES

As we stated earlier, 3-DES is secure against brute force attack due to the fact that it uses a key length of either 112 or 168 bits. Which makes it impossible to guess the key through an exhaustive search on any of today's fastest supercomputers. However, the best differential cryptanalysis attack applied to 3-DES was shown to need 2^{32} pairs. Although the complexity of this attack is 2^{112} which makes the procedure impractical to break, it is still less than the complexity of an exhaustive search which has a complexity of 2^{168} [19], [29].

2.5.3 Advanced Encryption Standard (AES)

The Advanced Encryption Standard (AES) was announced in December 2001 by the National Institute of Standards and Technology (NIST) as its new symmetric key block cipher. After DES was broken, NIST sought for a replacement specifying that the new algorithm should have a block size of 128 bits and different key sizes of the set 128, 192, and 256 bits. In October 2000 NIST announced that the Rijndael algorithm was selected out of the 21 proposed algorithm to be the AES, and AES was finally standardized later in December 2001 and published as FIPS 197 [30]. The general architecture of AES is shown in Fig. 2.17.

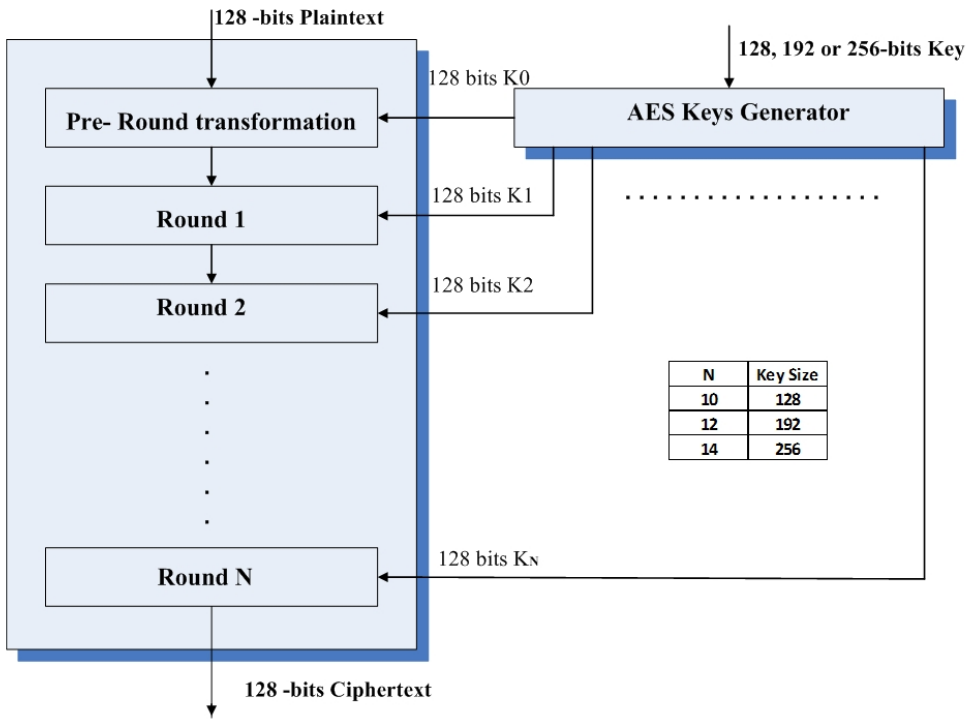


Figure 2.17. AES general architecture

The plaintext and ciphertext are both 128-bit blocks. AES works in three modes, and each mode has a different key length and different number of rounds. The first mode has a key of 128 bits and 10 rounds, the second mode has a key of 192 bits and 12 rounds, while the third mode has a key of 256 bits and 14 rounds.

AES round structure

Each round consists of four different types of bit transformations except for the last round, which has only three of the transformations. The transformations are sub-bytes, shift rows, mix columns, and add round key transformations. The last round misses the mix columns transformation, while the pre-round transformation only uses the add round key transformation. Figure 2.18 shows a block diagram for the general structure of AES round.

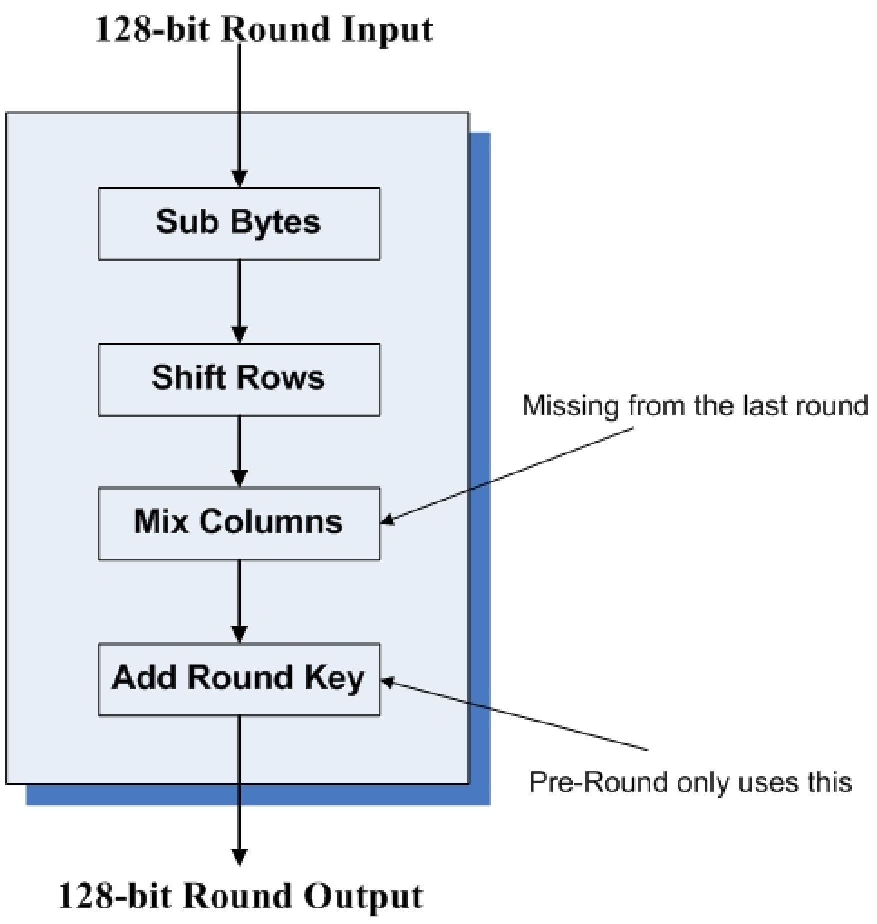
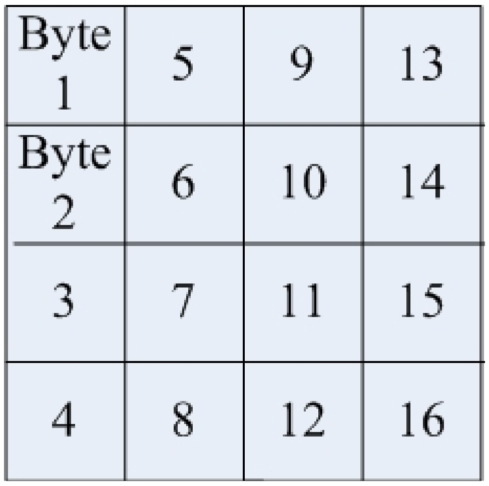


Figure 2.18. AES round structure

The input and output for each transformation is called a state, and each state is a 128-bit block represented by a row matrix of sixteen bytes as shown in the Fig. 2.19.

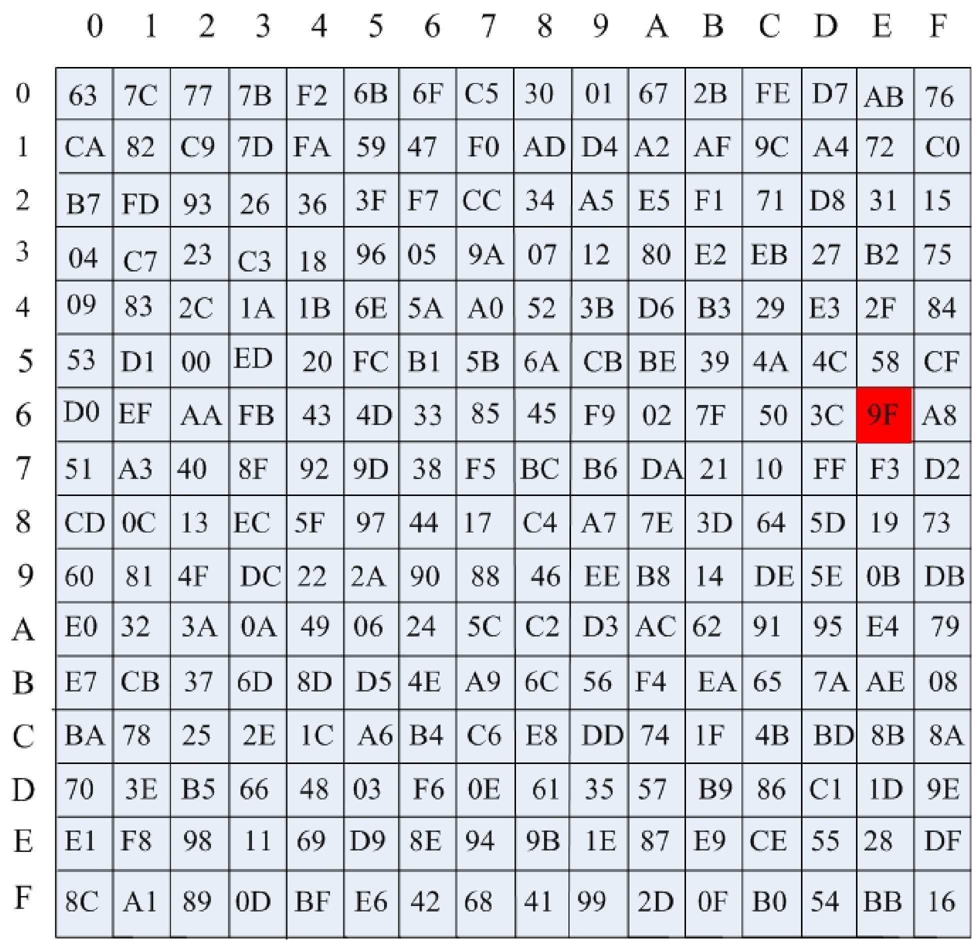


Byte 1	5	9	13
Byte 2	6	10	14
3	7	11	15
4	8	12	16

Figure 2.19. AES state

Sub-byte transformation

The sub-byte transformation is the first step in any round. The state is transformed to a new state, byte by byte. Each byte is transformed into another byte using the transformation table; the byte is taken as two hexadecimal digits, where the left digit defines the row number and the right digit defines the column number, and the table is the same as the S-Box transformation table except that it is 16×16 and the inputs of the table are from $00 - FF$. The mapping is done based on the table shown in Fig. 2.20. Assume that the first byte of the input block (state) has the hexadecimal value of $6E$. Then we look into the value at the intersection of row 6 with column E. The output is $9F$. Thus, the first byte of the output state is replaced with $9F$.



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	CB	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Figure 2.20. AES S-Box

Shift row transformation

The input state for the shift row transformation stage is the output state of the sub-byte transformation stage. In this stage, the byte rows of each state are shifted by a specific number; the first row is not shifted, the second row is shifted by one byte, the third row is shifted by two bytes, and the fourth row is shifted by three bytes.

Mix Column transformation

While the shift row stage works on the row level of the states, the mix column stage works on the column level. It converts each column of the current state to a new column by matrix multiplication of the column by a constant column matrix. The multiplication of the bytes is done in GF(2^8) with modulus (10001101).

Add round key transformation

This stage as well transforms each column of the current state to a new column by XORing each of the columns with its corresponding key word to produce the new column.

AES key generation

As introduced in the previous section, each round needs a round key that is used in the bit transformation. Therefore, the number of keys needed are more than the number of rounds by one, because each round in addition to the pre-round needs a round key. For example, if we take the 128-bit key size, the 128 bits are divided into four words (a word is 32 bits), and these words are numbered from 0 to 3. These words are used in producing the 44 words needed for all rounds (4 words $\times$ 11 rounds). The pre-round stage key consists of the words 0-3, the first round key consists of the words 4-7, and so on until the last round key consists of the words 40-43. The mechanism by which the 44 words are generated is shown in Fig. 2.21.

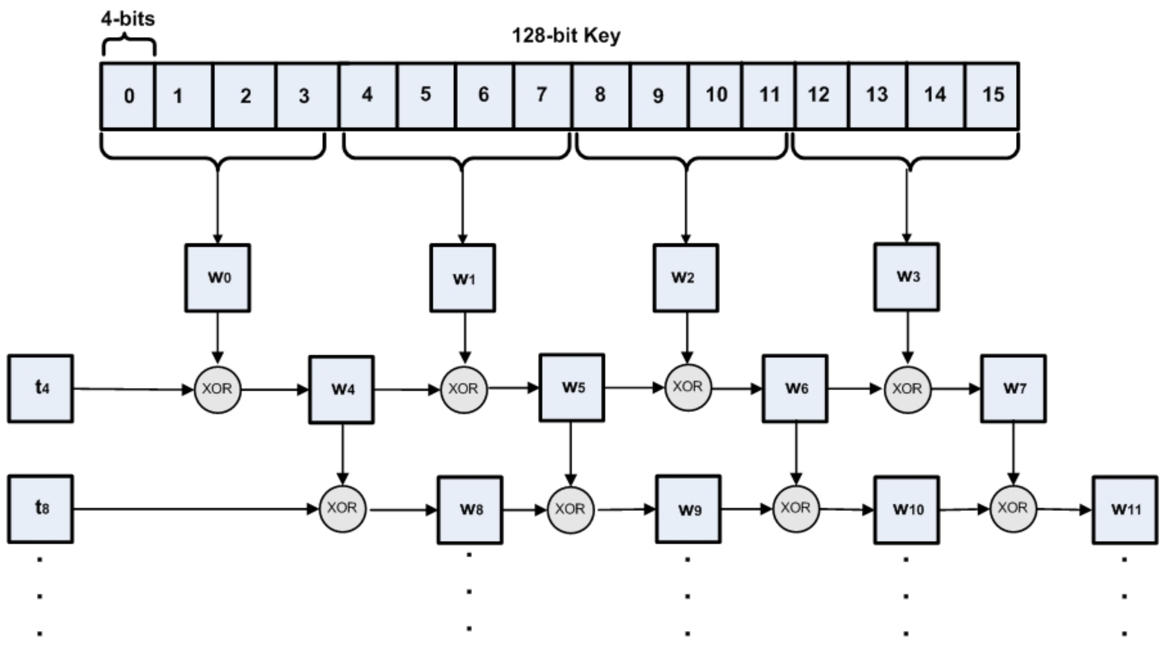


Figure 2.21. AES Key Generation

The words $t_4, t_8, ...$ are called temporary words; they are generated from some bit manipulations starting with constant words for each round.

AES security

Brute Force Attack

It was stated earlier that DES was vulnerable to brute force attack, because 2^{56} key trials can be done on today's computer within a reasonable amount of time. On the other hand, if AES is to broken using this attack, the process of breaking it needs 2^{128} key trials, and this is impossible on any of today's computers. If the time needed to break DES is t seconds, the time needed to break AES is $t \times 2^{128-56} = t \times 2^{72}$ seconds.

Differential Cryptanalysis

AES was designed after DES was attacked, and due to the fact that DES was known to be insecure to this attack, AES was designed in such a way that is prone to differential cryptanalysis. Given that the attacker has access to any number of plaintext-ciphertext pairs no one was able to crack the algorithm using this attack yet. However, some versions of AES with less number of rounds were attacked using differential cryptanalysis. Table 1.5 shows the number of pairs required to break simpler versions of AES along with the complexity of each attack. Although these attacks takes less time than an exhaustive search, those attacks are still impractical due to the difficulty of obtaining all the pairs, and a time complexity that is more than 2^{70} is considered unfeasible. We can also notice that for the 9 rounds version, a lower number of pairs are needed, but they are supposed to be generated by related keys rather than random keys [31], [32].

2.6 Previous Work on Joint Error Correction and Encryption

The property of avalanche effect in modern block ciphers has a catastrophic effect in wireless communication channels. Due to the noise, interference, and fading introduced by wireless channels, the encrypted data is received with some amount of error, and because of the strict avalanche effect, the decryption process results in half the original data to be decrypted in error. This problem attracted a lot of researchers due to its great impact and effect on the performance of encryption over wireless channels and hence various wireless applications. One approach that is used to solve the problem is to implement new, smart, and efficient coding techniques to eliminate the effect of the error caused by the channel at the receiver side, while another approach is to focus on the encryption algorithm itself by redesigning its components, in such a way to reduce the avalanche effect and compensate for the security using new entities and designs.

2.6.1 High Diffusion Cipher

This work was introduced in 2006 [33], and is a 10 rounds cipher based on picking some steps from the AES standard such as the S-Box and key mixing stages. However, the authors added a new stage to the first 9 rounds called the HD-encode stage. The HD-encode stage is based on a column multiplication of the previous state by a given 4×4 matrix for rounds 1 to 7 and a 7×4 matrix for rounds eight and nine. Those matrices were found to have less error propagation than the column multiplication stage of AES. The authors results outperforms AES in terms of error performance. However, the security of their design is less secure than AES (although their design is considered secure because, although it takes time less than an

Table 2.5. Differential Cryptanalysis on Weaker versions of AES

AES version	key Size	Required Number of Pairs	Time Complexity
AES (6 Rounds)	any	2^{32}	2^{72}
AES (6 Rounds)	any	$6 \cdot 2^{32}$	2^{44}
AES (7 Rounds)	192	$19 \cdot 2^{32}$	2^{155}
AES (7 Rounds)	256	$21 \cdot 2^{32}$	2^{172}
AES (8 Rounds)	192	2^{128}	2^{188}
AES (8 Rounds)	256	2^{128}	2^{204}
AES (9 Rounds)	256	2^{85} RK	2^{224}

exhaustive search, it can't be broken in a feasible amount of time). On the other hand, the algorithm is less efficient in terms of energy consumption than AES.

2.6.2 The Pyramid Cipher

This cipher encrypts a 128-bit plaintext into a 192 bit ciphertext using a 129-bit key [34]. It consists of five successive rounds, with each round consisting of three different stages. The key mixing and the substitution stages are identical to those of AES. However, the third stage consists of a column multiplication with a fixed 16×16 array for the first three rounds, and a 16×24 array for the remaining two rounds. The algorithm is shown to have an improved performance over AES when convolutional coding is concatenated to it. However, without the use of coding the pyramid algorithm performs similar to AES in terms of error performance. On the other hand, the time required for encryption and decryption is much greater than that required by AES. The authors shows that their algorithm is resistant to differential cryptanalysis attack by proving that it takes more than an exhaustive search to attack their algorithm.

2.6.3 Expanded Cipher Feedback Mode of Operation (ECFB)

In this work the author introduced a new mode of operation to be used with symmetric-key encryption algorithms [6]. The new mode is called expanded-cipher feedback mode. The author compared the error performance of using the new mode of operation with using traditional modes of operations such as the output feedback mode (OFB), cipher feedback mode (CFB), and the cipher chaining mode (CBC). The use of the proposed mode of operation improves the performance of the received information. The security of the new mode is well investigated and the new mode is shown to be secure against applicable types of attacks.

2.6.4 Qusai-Cyclic Low-Density Parity Check (QC-LDPC) Code based Encryption

In this work the authors introduced a new joint encryption and channel coding scheme, based on the use of quasi-cyclic low-density parity-check (QC-LDPC) codes with a symmetric-key secure encryption algorithm [3]. The proposed scheme utilizes a specific form of error vectors (perturbation vectors) that can be calculated from a random syndrome by the sender and receiver. The algorithm was proposed for the computation of the error vectors based on the right inverse of the parity-check matrix of the code. The proposed system provides an efficient error performance, an acceptable level of security and a low-complexity, practical implementation.

2.6.5 Turbo Codes Based Encryption

This work proposes a joint error correction and encryption scheme based on turbo codes [4]. Two inter leavers are used where the original inter leaver of turbo coding is used with some improvements to make it controlled by a secret key. Another inter leaver that is controlled by another key is added at the first output of the turbo encoder. The authors showed that the proposed algorithm slightly improves the error performance in a small range of signal-to-ratio (SNR), compared to the use of traditional turbo codes.

2.6.6 McEliece Public Key Cryptosystem

McEliece Cryptosystem is the first public encryption algorithm, it was developed in 1978 by Robert McEliece [35]. It was the first algorithm to use randomization in the encryption process, the algorithm is based on the hardness of decoding a general linear code (NP-hard problem). This algorithm has many advantages over the RSA algorithm. The encryption and decryption are faster, and the security grows rapidly with the increase of the key size. The algorithm in its original format was not popular and never gained much acceptance in the cryptography community. The algorithm uses variable size random permutation and substitution matrices. Some modifications to this algorithm were shown to have an improved error performance. However, most of those systems suffered from many security problems [36], [37], [38].

2.6.7 Kaks D-sequences

The author proposed an approach for joint encryption and error correction based on decimal expansion of fractions known as D-sequences [39]. In his work, he shows that using the proposed encoding operation is equivalent to using RSA encryption algorithm in terms of security. However, the complexity of the algorithm was high.

2.6.8 Godoy and Pereira Scheme

The purpose of this work was to alleviate the effect of the error in the communication channel without actually modifying or changing the encryption algorithm [40]. The algorithm derives new generator matrices from the original ones by row permutation. However, the number of generators were countable and finite, so the system was not immune against brute-force attacks.

2.6.9 Hwang and Rao Scheme

The authors proposed a new private-key encryption algorithm that is error correcting. The algorithm was based on non-linear channel coding [41]. However, the algorithm needed large parameters to reduce the error which increases the complexity of such an algorithm. The algorithm was shown to be vulnerable to some types of attacks.

2.6.10 Cryptocoding

This technique is based on quasigroup string transformation. In this technique, 16 orders of quasigroup are chosen over 2^{480} possibilities when the encoding functions are generated. This system achieves both error correction and security, at the cost of complexity, because the decoding process it is extremely complicated [42].

2.7 Modified Data Encryption Standard (M-DES)

M-DES was introduced as a modification to the data encryption standard [7], by which the error performance is improved and the security is enhanced compared to DES. In general, the design of M-DES is very similar to the design of the data encryption standard, which is a symmetric encryption algorithm except in the S-Box, and by introducing a new round we call Round 17, which has its own key that is different from the original DES 56-bit key.

2.7.1 Structure and Design

M-DES has two main modifications to the standard DES. First the number of distinct mapping tables is reduced from eight to two only. In addition, a new round 17 with a new 80-bit key is added to the original 16 rounds. The general design of the new algorithm is shown in Figure 2.22.

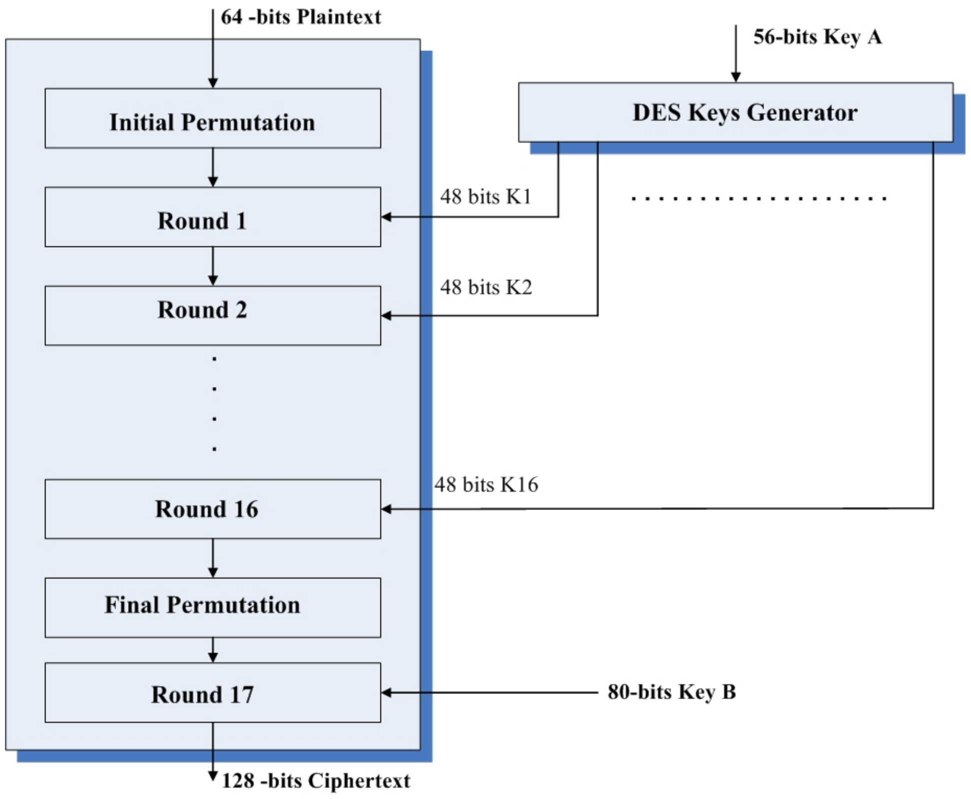


Figure 2.22. Design of the modified DES algorithm (M-DES)

S-Boxes

The S-Box is mainly responsible for the shuffling of the bits. A careful design of the S-Box will result in an avalanche effect of 0.5, which produces a secure cipher [2], [1]. However, to reduce the error in M-DES, the first four S-Boxes uses a similar mapping tables which are the same as the first mapping table of the standard DES, while the fifth through the eighth S-Boxes use similar mapping tables that are the same as the fourth S-Box mapping table in the standard DES.

Round 17

The motive for the second modification is in fact coming from the work in [21], where the authors showed that DES can be cracked using the differential cryptanalysis attack if the attacker has 2^{47} pairs of plaintext and ciphertext. The authors also showed that each distinct S-Box mapping table requires around $2^{6.5}$ pairs to be cracked. Therefore, in our proposed M-DES, while using only two distinct mapping tables (rather than the eight distinct mapping tables in the standard DES), the number of pairs needed to crack the algorithm reduces to 2^{13} pairs. To overcome this security reduction, a new round is introduced in M-DES. By introducing round 17, the algorithm becomes in fact secure to both brute force and differential cryptanalysis attacks as will be shown later in this section. Round 17 has two inputs and one output, the two inputs are the 64-bit output of the final permutation stage and an 80-bit key; the output is the 128-bit cipher. The 80-bit key is used to map the 64-bit input of Round 17 to a 128-bit output. This mapping procedure is shown in Fig. 2.23 where the 64-bit input of round 17 is divided into sixteen sub-frames of four bits each and the 80-bit key is divided into 16 sub-keys, each of five bits, while the 128-bit output consists of 32 sub-frames of four bits each. Each five bits of the 80-bit key is used to map one of the 4-bit input sub-frames to one 4-bit output sub-frame. So the input sub-frames will be scrambled randomly in 16 out of the 32 output sub-frames according to the key. The remaining 16 sub-frames of the output are randomly filled with zeros and ones. For example, suppose the first 5 bits of the key are 00101. This means that the first 4-bit sub-frame of the input will be mapped to the fifth 4-bit sub-frame of the output. At the destination, the receiver is assumed to have the 80-bit key, so it will be able to recover the useful 64-bit ciphertext out of the total 128-bit received ciphertext.

In summary, the proposed algorithm has a plaintext of 64 bits, an overall ciphertext of 128 bits, and an overall key of 136 bits.

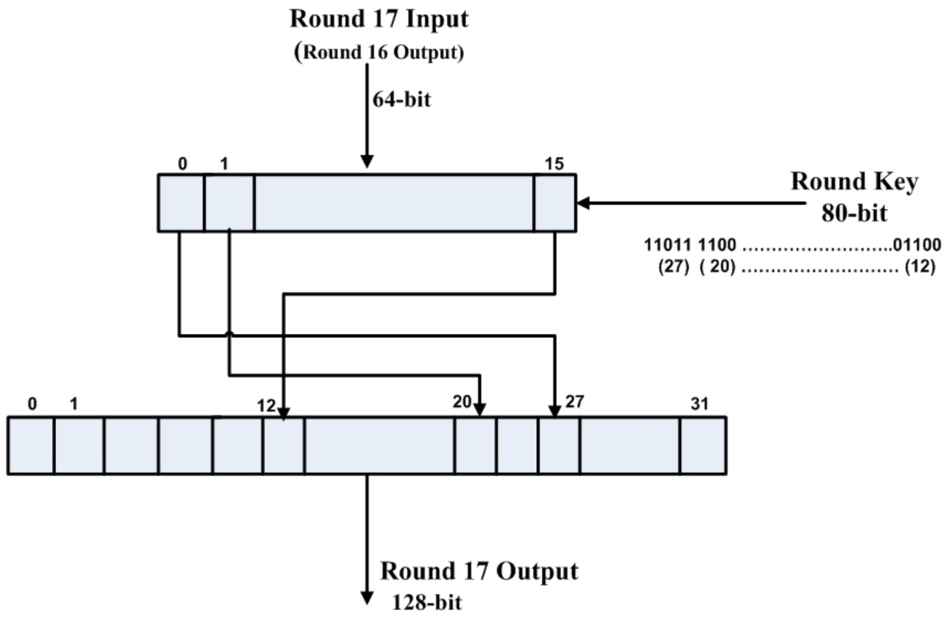


Figure 2.23. Round 17 Design, mapping the 16 input sub-frames into the 32 output sub-frames using the 80-bit key

2.7.2 Security Analysis

The security of M-DES encryption algorithm is measured by the attacker's ability to crack it. There are five general types of attacks that may be applied to any encryption algorithm: ciphertext only, known plaintext, chosen plaintext, chosen ciphertext, and chosen text attacks. However, in symmetric block cipher algorithms the first and second types are the most applicable attacks, so we analyze the effect of the brute force attack on our proposed algorithm as an example for the first type of attack and the differential cryptanalysis as an example for the second type of attack.

Resistance to Brute Force Attack

The brute Force Attack is based on trying all possible keys in decrypting some ciphertext that the attacker obtains, until he finds a reasonable or expected plaintext, and thus that specific key is the correct key that was used in encryption and decryption.

When DES was originally designed it was impossible to think about this kind of attacks. Trying 2^{56} different keys was not feasible and impossible because it would take an infinite amount of time. However, this mission has become easier, because with a multi processor computer an attacker can check all 2^{56} keys in less than 20 hours. In the new algorithm the key length becomes $56 + 80 = 136$ bits, which produces immunity against the brute force attack because 2^{136} keys will need to be tested in order for an intruder to break the algorithm using this attack, a task that cannot be accomplished on any of today's computers. Furthermore, assuming that the key of DES can be cracked in 1 second (testing all 2^{56} keys), the key of the new algorithm needs more than 100 trillion years to be cracked.

Resistance to Differential Cryptanalysis

The differential cryptanalysis attack was introduced by Eli Biham and Adi Shamir as a type of known plaintext attack. The attack idea supposes that the attacker gains access to one of the communication party's computer, and gets a number of pairs of plaintexts and their corresponding ciphertexts from the victim's database. In this type of attack, the attacker analyzes the relationship between the different plaintext and analyze their propagation at the ciphertexts. It has been shown that 2^{47} pairs of $< plaintext, ciphertext >$ is enough to break DES using this attack.

In M-DES, the reduction of the number of S-Boxes from 8 to 2, has reduced the number of pairs needed to crack the algorithm using differential cryptanalysis from 2^{47} to 2^{13}. However, in DES if those 2^{47} pairs are available, the algorithm can be cracked in a specific amount of time. On the other hand, if the 2^{13} pairs are available in M-DES, the intruder can not use those pairs directly. The useful 64-bit ciphertext[1] will need to be extracted out of the 128-bit cipher for each of the 2^{13} pairs. Therefore, the probability of obtaining one useful pair out of the 2^{13} pairs is given by

$$p = \frac{1}{\binom{32}{16} \times 16!} = 7.9515 \times 10^{-23}. \tag{2.7.1}$$

And the experiment of guessing the useful ciphers and their correct order for all the $N = 2^{13}$ ciphers is a bernoulli trial (repeating the experiment N times out N ciphers). Therefore, the probability of a successful attack on M-DES using differential cryptanalysis is given by

$$\begin{aligned} P_{attack} &= \binom{N}{N} p^N (1-p)^{N-N} \\ &= 3.011 \times 10^{-181040} \\ &\approx 0. \end{aligned} \tag{2.7.2}$$

This probability is vanishes to zero, so the algorithm is considered immune to differential cryptanalysis.

Another way to look at the possibility of attacking the algorithm using the differential cryptanalysis is to calculate the required time needed to do so. The number of trials an attacker needs to perform to achieve a successful attack is

$$\begin{aligned} N_{trials} &= \left(\binom{32}{16} \times 16!\right)^{2^{13}} \\ &= 7.7 \times 10^{172825} trials. \end{aligned} \tag{2.7.3}$$

However, the fastest super computer today can perform 2.5×10^{12} operations per second. Therefore, the total time needed to attack M-DES using differential cryptanalysis is given

[1]A useful ciphertext is the 64-bit cipher that was scrambled in the 128-bit cipher. This process will make the intruders task harder, because the likelihood of guessing the correct 16 sub-frames out of the 32 sub-frames in a feasible time is very low.

by

$$\begin{aligned} T_{attack} &= \frac{7.7 \times 10^{172825}}{2.5 \times 10^{12}} \\ &= 3.08 \times 10^{172813} seconds \\ &= 1 \times 10^{172806} years \end{aligned} \tag{2.7.4}$$

which is about 14400 trillion years.

2.7.3 Error Performance in the Wireless Channel

In this section we evaluate the performance of the proposed algorithm in wireless channels, in terms of its error performance in these channels. Two scenarios are assumed: in one channel scenario the channel is assumed to expose additive white gaussian noise (AWGN), and in the other scenario, the channel expose AWGN in addition to Nakagami-m fading. Both scenarios are analyzed next. We also analyze the performance of the use of one asymmetric encryption algorithm in Appendix A

Performance in AWGN Channels

Computer simulations are used to evaluate the performance of the proposed algorithm assuming AWGN wireless communication channel with binary phase shift keying (BPSK) modulation. A large sequence of bits (one million bits in our simulation experiments) is generated randomly at the sender. The generated bits are then divided into 64-bit blocks each encrypted into a 128-bit block cipher using our proposed M-DES. The ciphers are then transmitted over the channel. The received ciphers are then decrypted block by block at the destination. The resulting sequence of bits at the receiver after decryption is compared with the sequence of bits at the sender before encryption, and the error is calculated for different SNR values of the channel. This same simulation procedure is repeated for DES as well.

In Fig. 2.24, we investigate the performance payoff from increasing or decreasing the security level by having four (increased security) or one (reduced security) distinct mapping tables, respectively, rather than two distinct mapping tables (case of Fig. 2.26). It can be inferred from the figure that having one distinct mapping table results in an enhanced BER performance while having four distinct mapping tables degrades the BER performance. Fig. 2.25, shows the relationship between the BER before and after decryption for both DES and M-DES. It is clear from the figure that all versions of M-DES (1, 2, and 4 distinct mapping tables) outperform DES. It is also clear that M-DES in its two and one distinct S-Boxes mapping tables versions outperforms the use of the mode of operation described in [6].

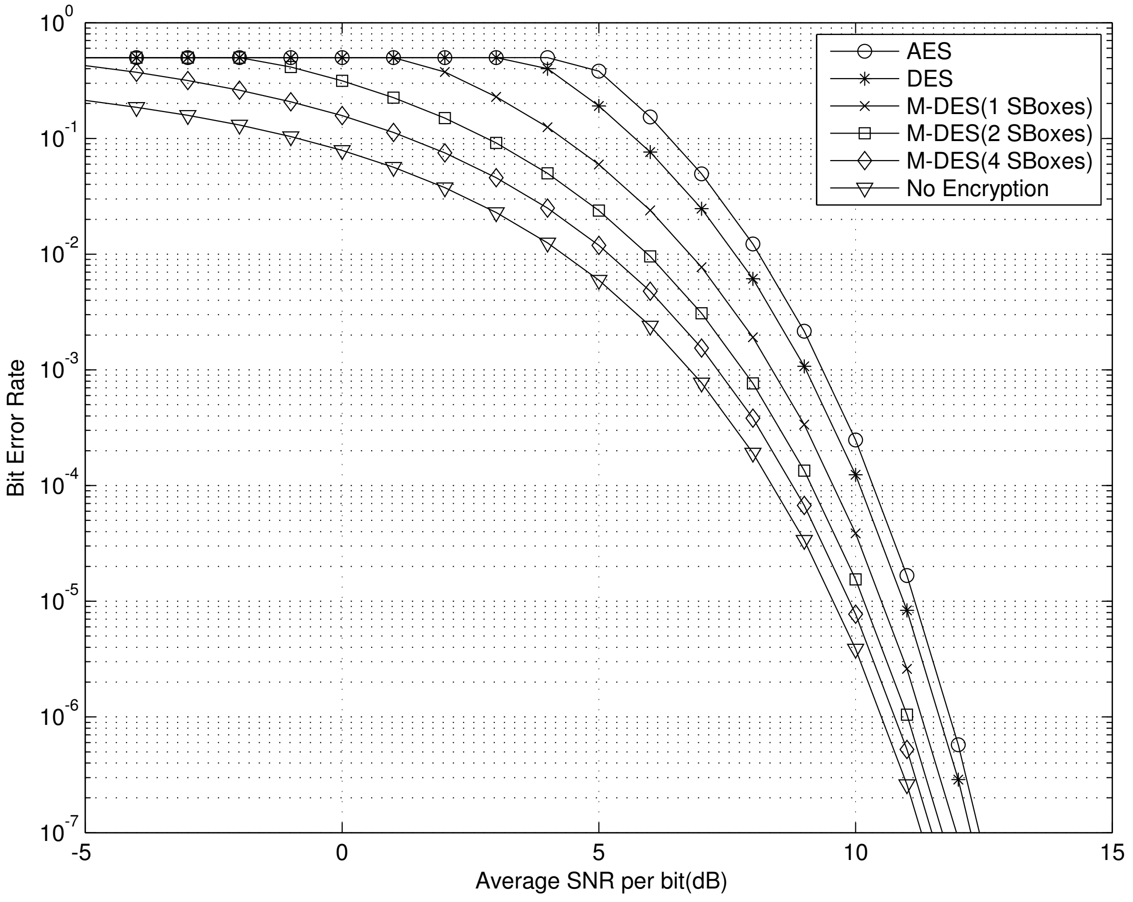


Figure 2.24. BER vs. SNR in AWGN channel with BPSK modulation, comparison of AES, DES vs. different modes of M-DES

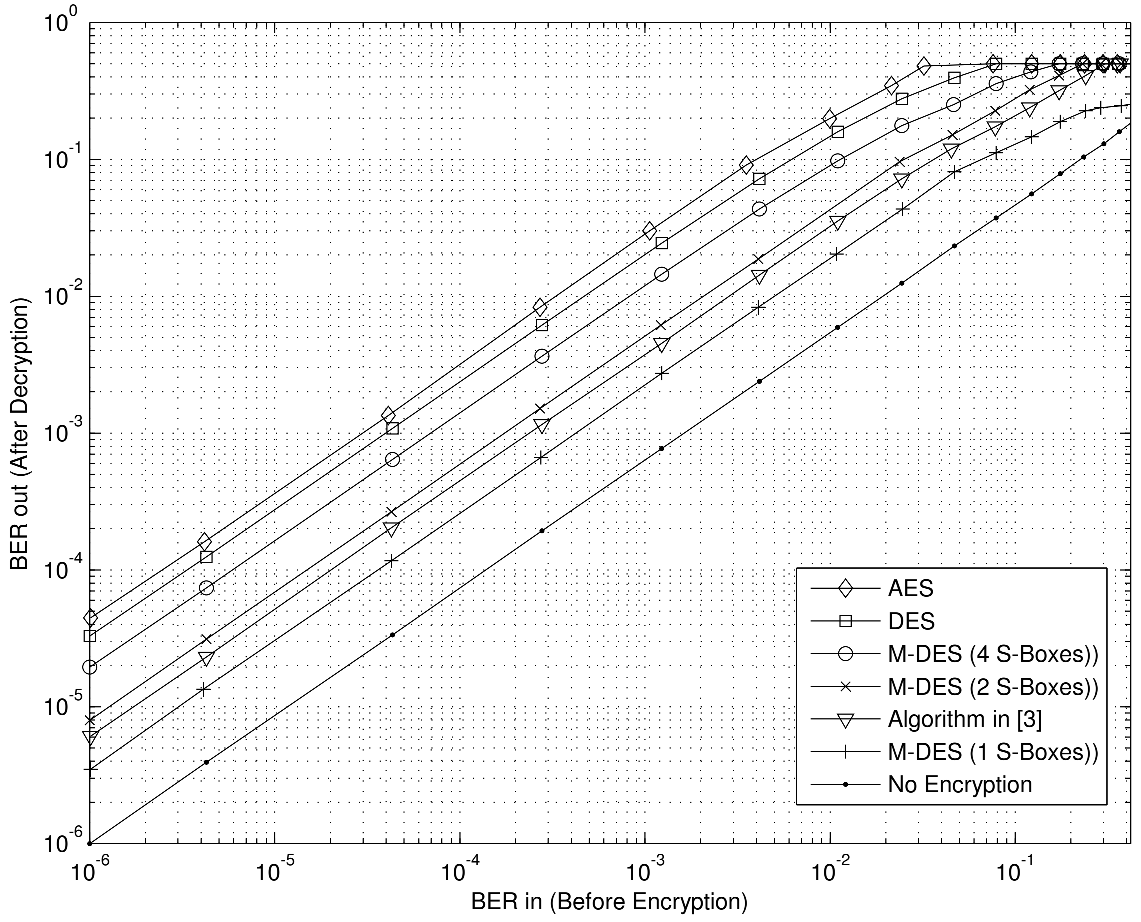


Figure 2.25. BERout vs. BERin, comparison of DES vs. different modes of M-DES vs. Haleems mode of operation. BERin is the error at the receiver before decryption(imposed by channel noise only) while BERout is the error after decryption.

Performance in Nakagami-m Fading Channels

Computer simulations are used to evaluate the performance of the proposed algorithm assuming an AWGN wireless communication channel with fading and BPSK modulation. We assume Nakagami-m fading (with Rayleigh as a special case when $m = 1$). The same procedure as in the previous sub-section is repeated, with the assumption that fading is affecting the signal in addition to gaussian noise.

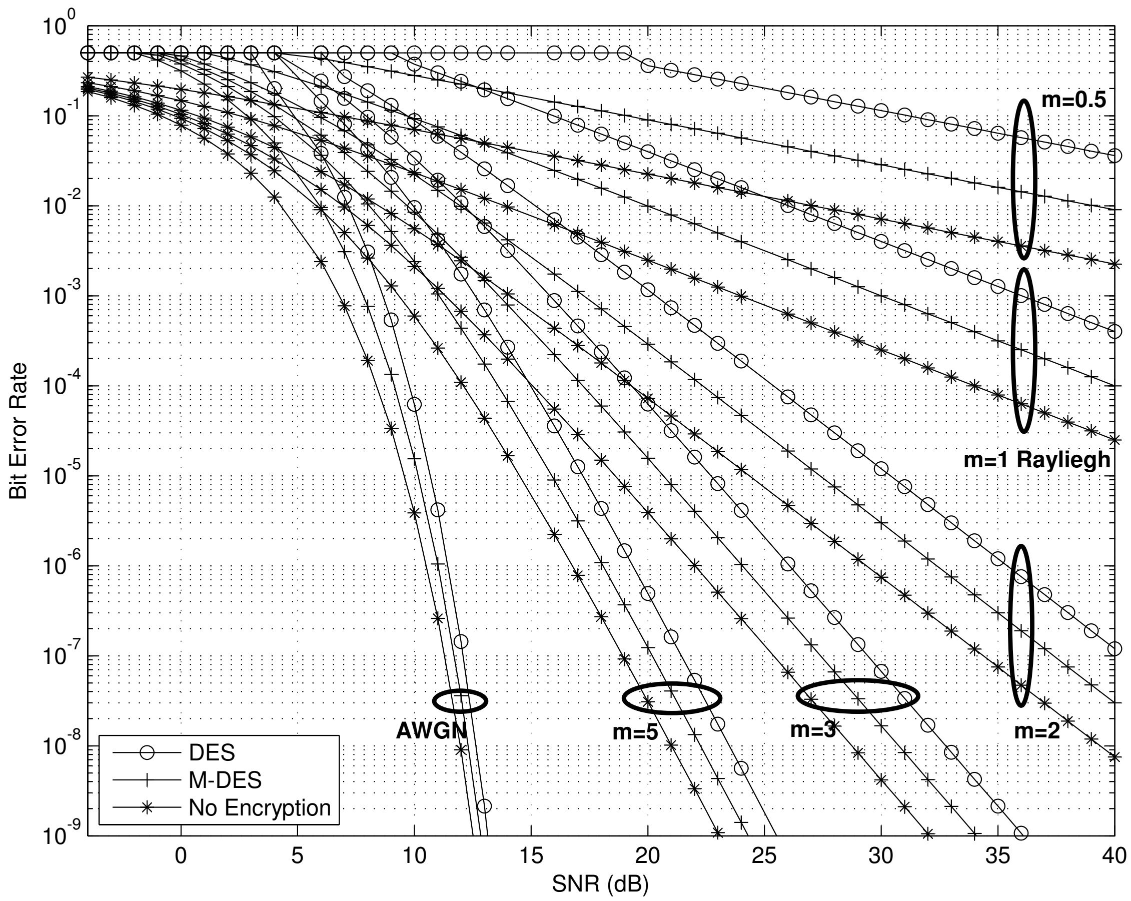


Figure 2.26. BER vs. SNR assuming AWGN channel and Nakagami-m fading channel with BPSK modulation, comparison of the BER when DES and M-DES algorithms are used, the effect of not using encryption is also compared with DES and M-DES encryption algorithms

Fig. 2.26 depicts the BER performance versus the average SNR per bit for both M-DES and DES assuming AWGN channel and Nakagami-m fading channel and BPSK modulation. The results in the figure cover the cases when no fading and fading exists for different values of m. We also plotted the curves when no encryption is used as a reference. It should be clear that in Fig. 2.26 the BER of DES is constant (avalanche) between -5 to 2 dB in AWGN and -5 to 10 in Rayliegh (i.e: Rayliegh is a spacial case of Nakagami-m when $m = 1$). On the other hand, these errors will result in less than half the bits to be in error (no avalanche) when M-DES is used. It can be clearly noticed from the figure that the BER increases when m decreases as expected, because the channel conditions are worse due to the large amount of fading. Also it is obvious from the figure that the performance gain obtained by M-DES becomes larger when the fading is more severe (e.g., $m = 1$, $m = 0.5$).

It should be noted here that M-DES uses 64 extra bits in the cipher as compared to DES, which results in half rate loss in system throughput. This is in fact the cost paid for the gain in the error performance and security level obtained by our proposed algorithm as compared to DES.

2.8 Conclusion

In this chapter we presented the main well-known encryption algorithms, we also introduced the state of the art in the area of joint encryption and error correction techniques in wireless channels. In the next chapter, we are studying the effect of each distinct S-Box mapping table and each round on the error performance and on the security.

Chapter 3

The Effect of the Number of Rounds and S-Box Mapping Tables in Symmetric Encryption Algorithms

3.1 Introduction

The number of rounds and S-Boxes in any symmetric cryptosystem plays a great role on its security as well as on the probability of correct reception at the receiver after decryption. This work considers a class of symmetric encryption algorithms based on the data encryption standard (DES) and sheds lights into understanding the effect of each the number of rounds and number of S-Boxes on the security of the encrypted data as well as on the probability of correct reception in the wireless channel. In wireless channels, some encrypted bits could be flipped while transmitted over the channel due to noise, interference or fading. This degradation in the wireless channel conditions causes the data after decryption to have some amount of error that depends on the number of bits in error received in the encrypted data, before decryption, and on the location of these bits in the cipher block. We consider the number of rounds, the number of S-Boxes and the channel conditions (in terms of signal-to-noise power ratio (SNR)); and study their effect on security level and detection error performance. Using numerical computations and computer simulations, when considering a certain encryption mechanism, we present qualitative and quantitative analysis on the tradeoff between communication reliability and security levels versus the fluctuations in the wireless channel conditions.

3.2 Motivation

The importance of this work is due to the fact that we need to understand the effect of each S-Box and each round on the probability of correct reception at the receiver as well as on the security of the encrypted data. These effects are studied in wireless environment to clearly understand the effect of the number of S-Boxes and rounds in wireless communication channels.

3.3 Contributions

In this chapter, we study how the error performance and the security are affected by each distinct S-Box mapping table and by each round. The effect on the error performance is studied assuming different signal-to-noise values, which implies different channel conditions, therefore, the uniqueness of this analysis is due to the fact that it has been evaluated based on the number of rounds, the number of S-Boxes and the channel conditions. On the other hand, we show that the security analysis in this chapter is independent of the channel conditions and is a function of the number of rounds and S-Boxes only.

3.4 Effect of the number of rounds and S-Boxes on the error performance

In this section we evaluate the effect of number of rounds and S-Boxes in different channel conditions (i.e., different SNR values) on the probability of correct reception for both DES and M-DES.

3.4.1 The Effect on DES Error Performance

Given that the probability of correct reception is evaluated based on three different factors (i.e., number of rounds, number of S-Boxes and the SNR), we assume three different scenarios in our analysis:

1. Fixing the number of S-Boxes and changing the number of rounds and the SNR
2. Fixing the SNR and changing the number of S-Boxes and the number of rounds
3. Fixing the number of rounds and changing the number of S-Boxes and the SNR

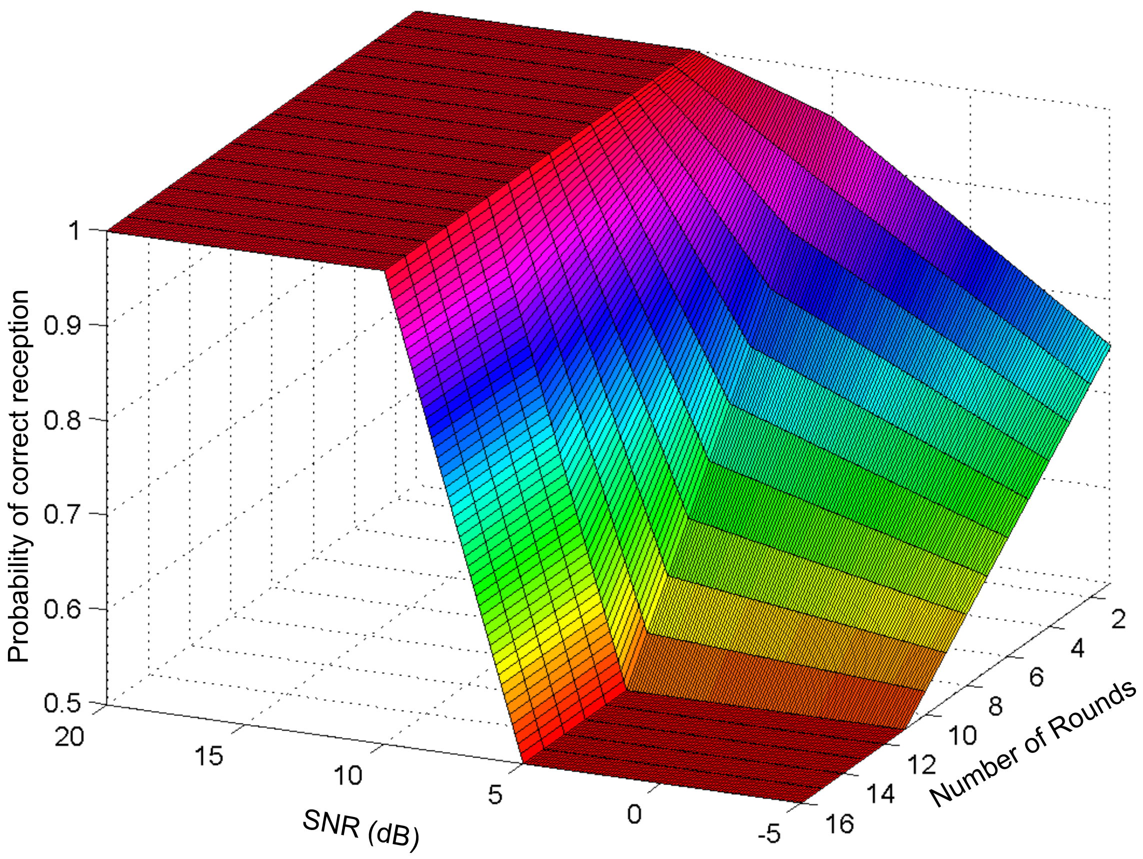


Figure 3.1. The effect of the number of rounds and SNR on the probability of correct reception for a fixed number of distinct S-Boxes equal to eight

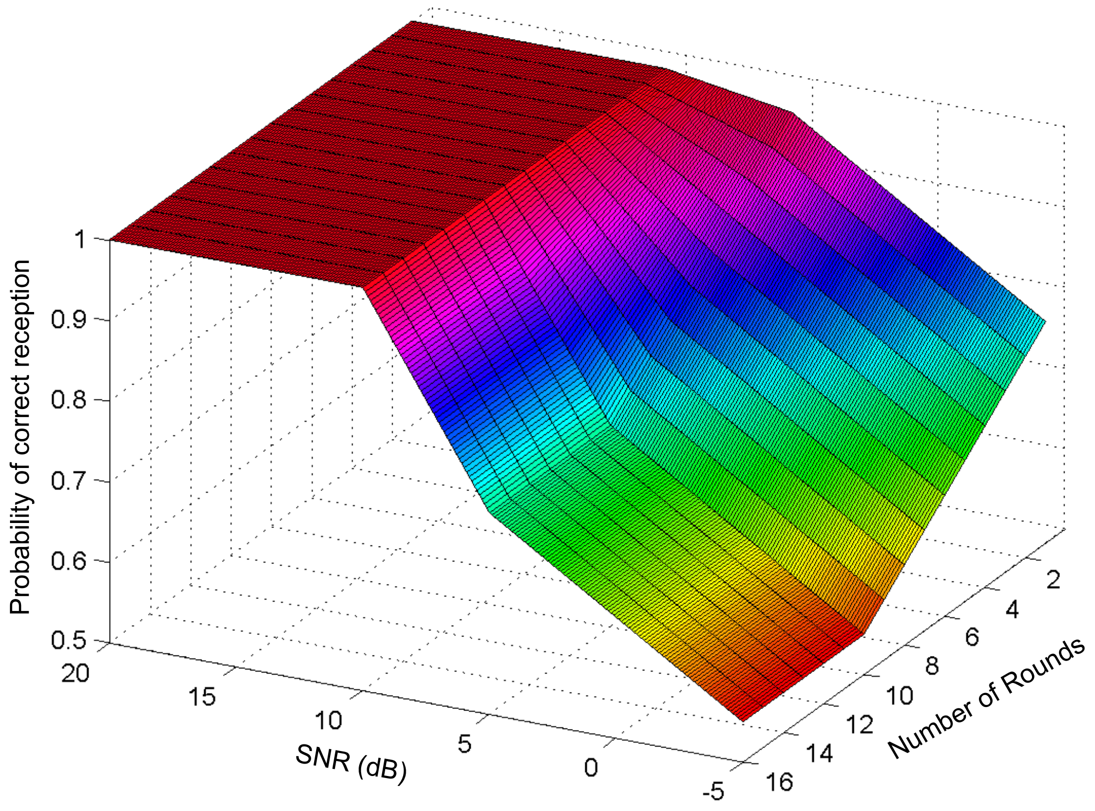


Figure 3.2. The effect of the number of rounds and SNR on the probability of correct reception for a fixed number of distinct S-Boxes equal to four

Figures 3.1 - 3.4 show different simulation results for the first scenario where the number of S-Boxes is fixed to eight, four, two and one in Figures 3.1, 3.2, 3.3 and 3.4, respectively. It can be noticed from Figure 3.1, where the number of S-Boxes is eight, the number of rounds is changed from 1 to 16 and the SNR is changed from -5 to 20 dB, that the area bordered by SNR values from -5 to 5 dB and 16 to 12 rounds represents the Strict avalanche effect (SAC) area, because the probability of correct reception within this area is equal to 0.5. However, if we move to an area outside this range for example 8 rounds with an SNR equal to 0 dB, the probability of correct reception increases to 0.57. In general, we can conclude that when the number of rounds decreases, the probability of correct reception increases and when the SNR value increases the probability of correct reception will increase as well, as expected. The effect of these changes on the security of the encrypted data will be analyzed in the next section. Figures 3.5 - 3.8 show different simulation results for the second scenario where the

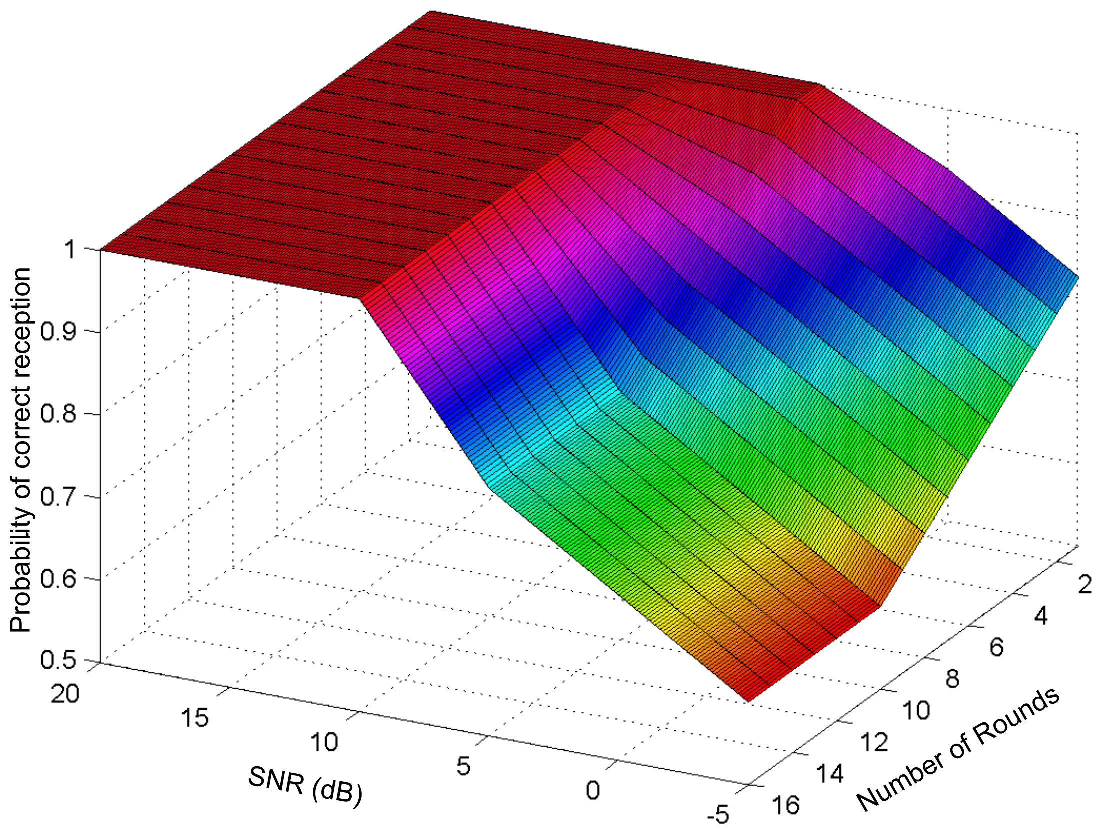


Figure 3.3. The effect of the number of rounds and SNR on the probability of correct reception for a fixed number of distinct S-Boxes equal to two

SNR is fixed to -5, 0, 5 and 10 dB in Figures 3.5, 3.6, 3.7 and 3.8 respectively. It can be noticed from Figure 3.5 where the SNR is fixed to -5 dB and the number of rounds is changed from one to sixteen and the number of S-Boxes is changed from one to eight, that the SAC is only present when the number of rounds is between twelve and sixteen while the number of S-Boxes is at eight. It can be clearly noticed from Figures 3.5 - 3.8 that the probability of correct reception is improved when the number of S-Boxes and rounds decrease. While the probability of correct reception is enhanced significantly when the SNR is increased, which can be noticed in Figure 3.8, where the probability of correct reception is always equal to 1, due to the fact the SNR is equal to 10 dB. Therefore, we can also notice that when the channel conditions are perfect (i.e, $SNR \geq 10\ db$) the probability of correct reception is equal to one, for any number of rounds or S-Boxes.

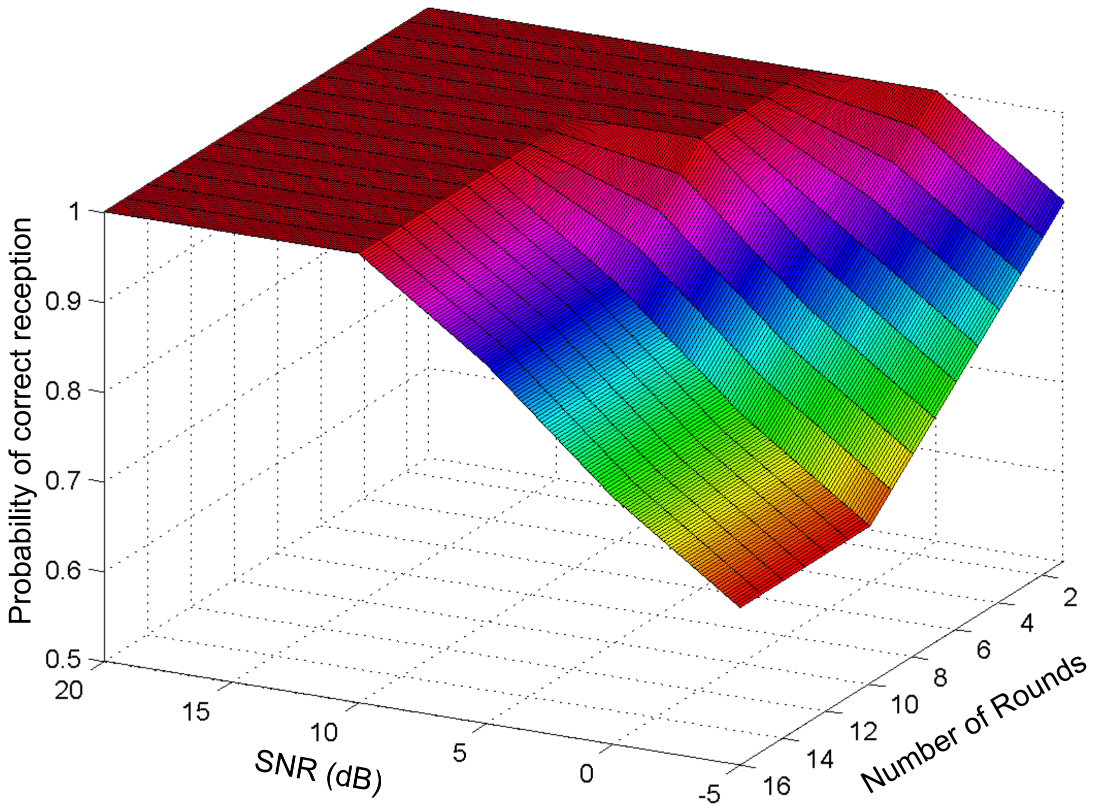


Figure 3.4. The effect of the number of rounds and SNR on the probability of correct reception for a fixed number of distinct S-Boxes equal to one

Figures 3.9 - 3.12 depict simulation results for the third scenario where the number of rounds is fixed and is equal to one, four, eight and twelve rounds in Figures 3.9, 3.10, 3.11 and 3.12, respectively. It can be noticed from Figure 3.9 where the number of rounds is fixed to one round and the number of S-Boxes is changed from one to eight and the SNR is changed from -5 to 20 dB, that the SAC is not present in this case, due to the fact that the number of rounds is equal to the minimum. Similarly, when the number of round is equal to four and eight in Figures 3.10 and 3.11, the SAC is still not present. However, the SAC starts to show when the number of rounds is equal to 12. Therefore, we can conclude that that the SAC is not present in DES until the number of rounds is equal to 12 or more, regardless of the channel condition and the number of S-Boxes.

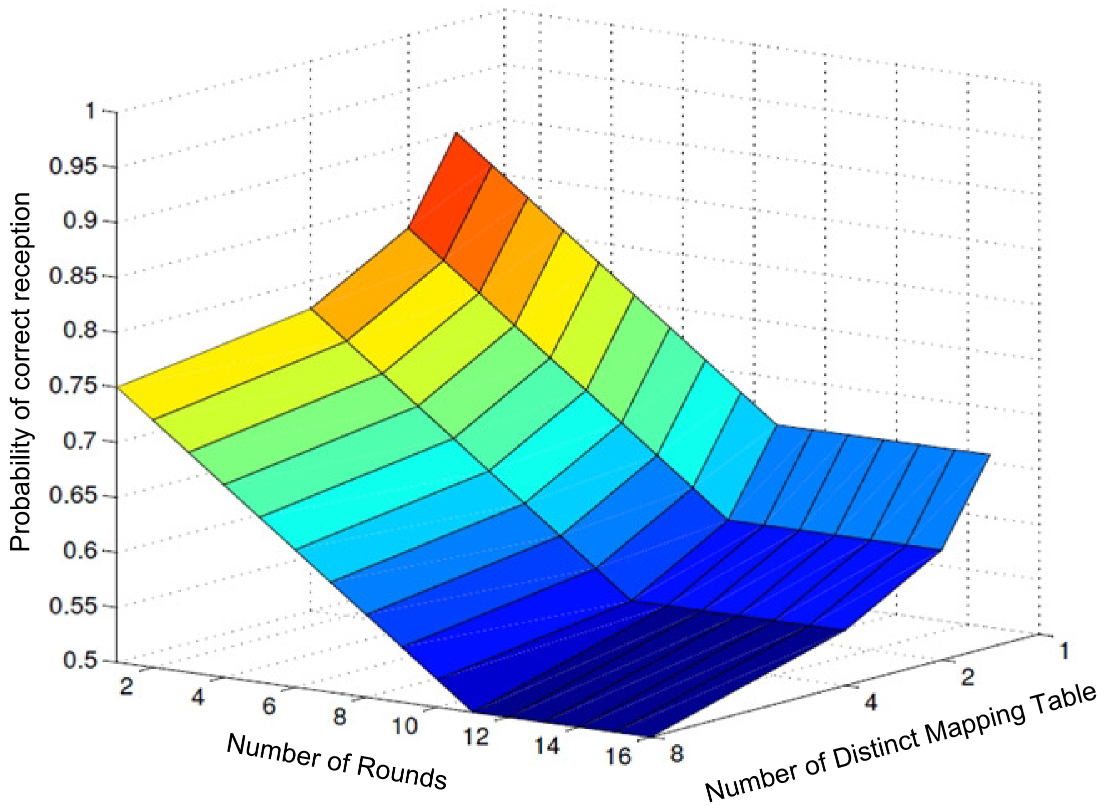


Figure 3.5. The effect of the number rounds and the number of distinct number of S-Box mapping tables on the probability of correct reception for a fixed SNR equal to -5 dB

By the end of this section, we can notice from the three different simulation scenarios, that SAC is present only when the following three conditions are present all together

1. The number of S-Boxes is eight
2. The number of rounds is between 12 and 16
3. The SNR value is between -5 and 5 dB.

3.4.2 The Effect on M-DES Error Performance

By introducing round 17, reducing the number of S-Boxes and changing the design of the S-Box mapping tables in M-DES, the algorithm is shown to alleviate the SAC under all circumstances. The number of the S-Box mapping tables in M-DES are two. Therefore, by

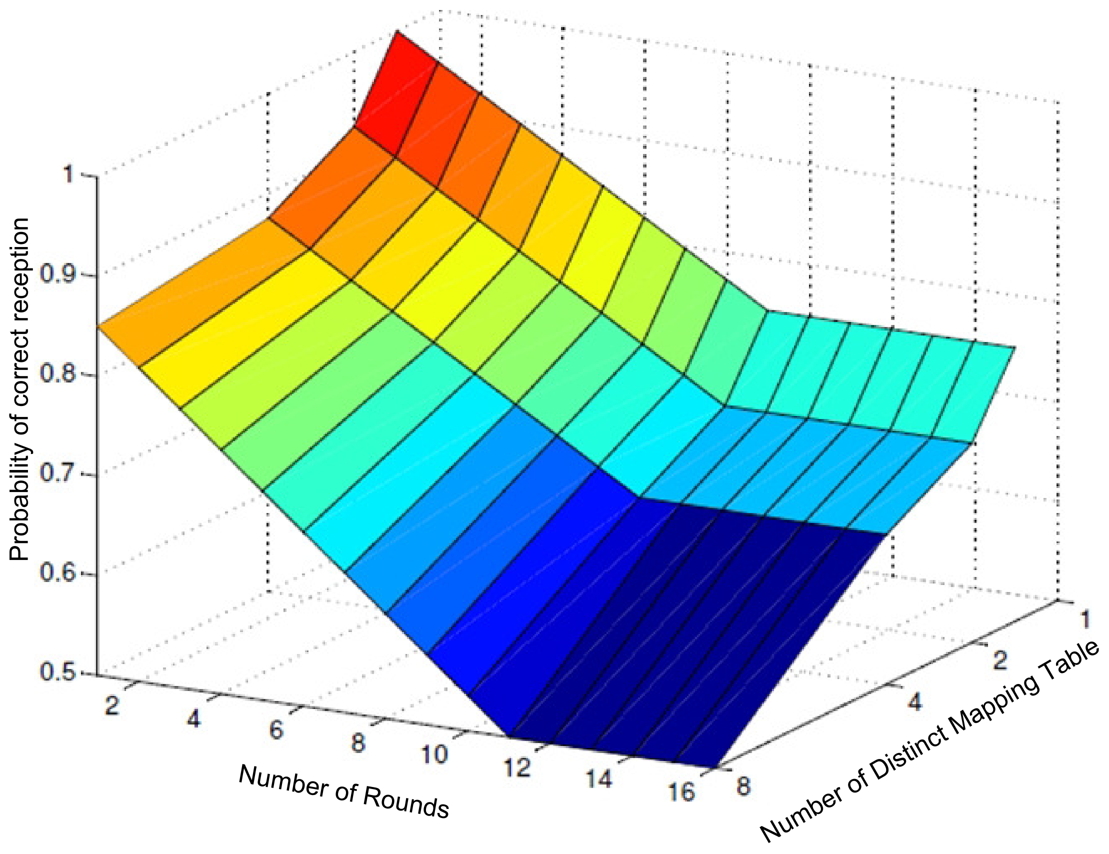


Figure 3.6. The effect of the number rounds and the number of distinct number of S-Box mapping tables on the probability of correct reception for a fixed SNR equal to 0 dB

alternating the number of rounds in M-DES as well as changing the channel conditions (i.e., SNR of the channel), the probability of correct reception will be changed. However, when the number of rounds is changed, round 17 is still applied after the selected number of the original rounds to alleviate the SAC effect. To summarize, in this simulation the number of S-Boxes is fixed to two S-Boxes (M-DES number of S-Boxes) and the probability of correct reception is computed for every single value of SNR between -5 and 20 dB and for every different number of rounds between one and sixteen (with round 17 applied in all the cases). Table 3.1 summarizes the probability of correct reception for the case when the SNR is equal to -5 dB for different number of rounds from one to sixteen with round 17 applied in all cases.

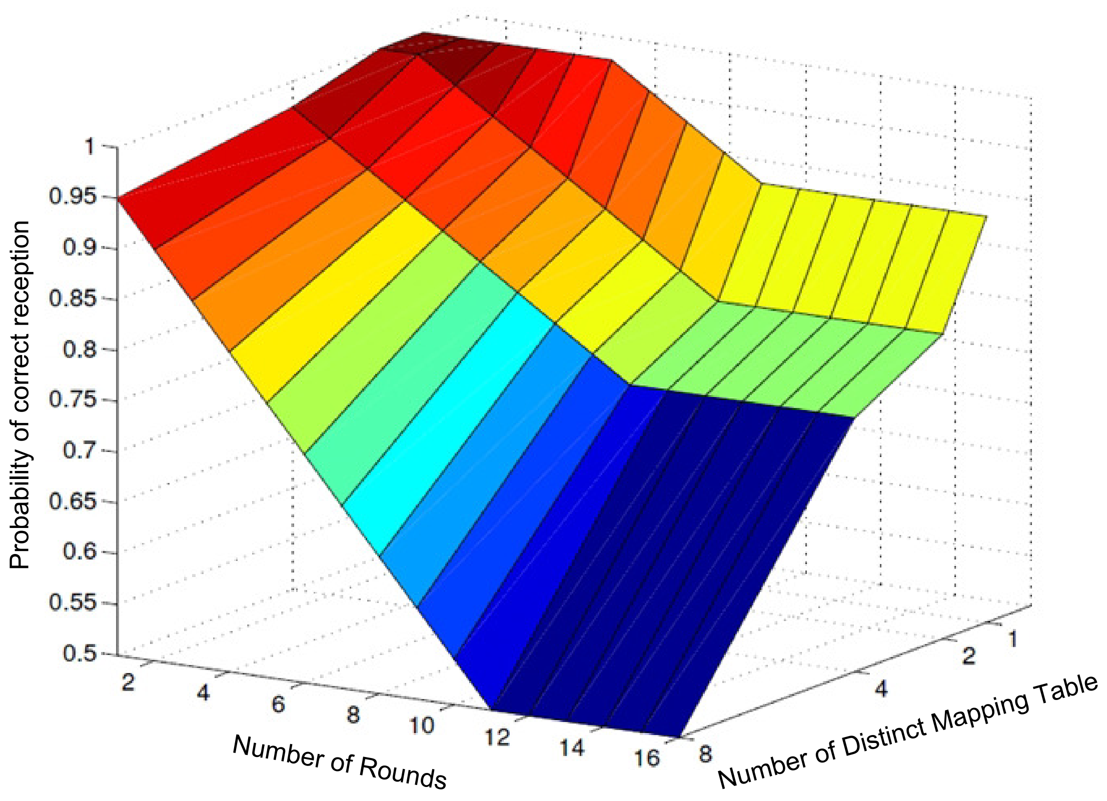


Figure 3.7. The effect of the number rounds and the number of distinct number of S-Box mapping tables on the probability of correct reception for a fixed SNR equal to 5 dB

It can be noticed from Table 3.1 that the probability of correct reception in M-DES is significantly improved compared to that of DES for the same number of rounds, S-Boxes and SNR value. This improvement in the error performance is a result of the change in the S-Boxes design as well as to the addition of round 17.

Hence, in this section we show that the probability of correct reception in M-DES is a function of two variables, which are the number of rounds and the SNR. Moreover, we show that M-DES significantly improves the error performance compared to DES. In the next section, we show the effect of the number of rounds and S-Boxes on the security of the encrypted data. In addition, we evaluate the security addition of round 17 to M-DES.

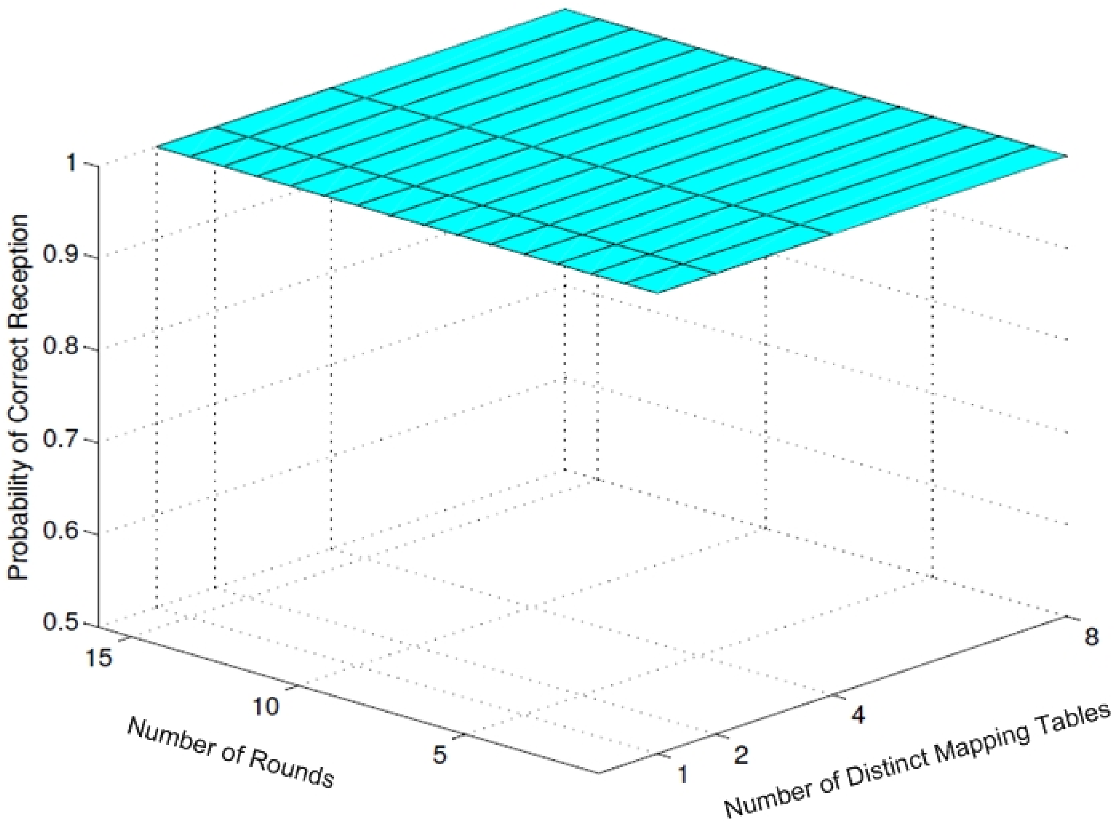


Figure 3.8. The effect of the number rounds and the number of distinct number of S-Box mapping tables on the probability of correct reception for a fixed SNR equal to 10 dB

3.5 Effect of the number of rounds and S-Boxes on the Security

When the security of symmetric encryption algorithms is evaluated, two main attacks are considered. The Brute force attack where the key is attacked; and the differential or linear attacks where the actual data is attacked [19]. The number of rounds and S-Boxes does not have an effect on the brute force attack due to the fact that the brute force attack is an attack where all possible keys are tried in decryption to get the correct or expected plaintext. Moreover, the attacker who obtains a ciphertext, tries all possible keys to decrypt the ciphertext he obtained, until he gets a plaintext that he expects out of the decryption process. The obtained key can be further used to decrypt all other ciphertext obtained by the attacker. However, the number of rounds and S-Boxes directly impacts the strength of the

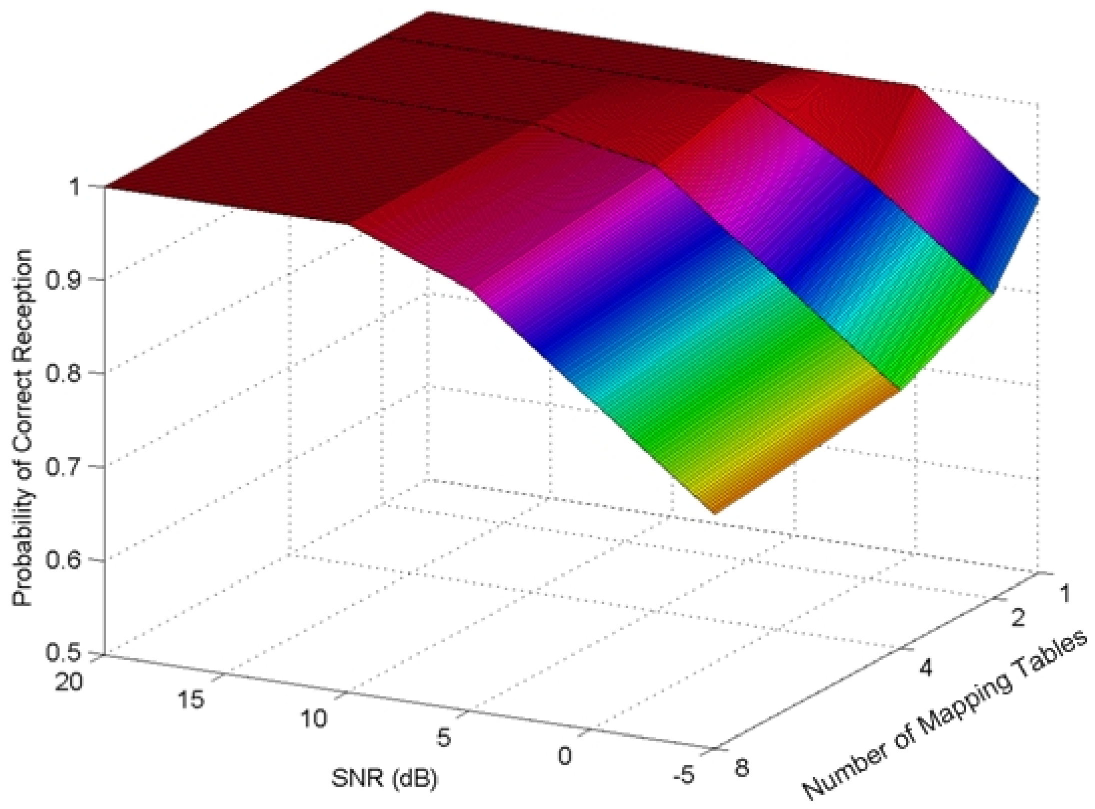


Figure 3.9. The effect of the number of distinct S-Box mapping tables and SNR on the probability of correct reception for a fixed number of rounds equal to 1

differential cryptanalysis attack due to the fact that this attack is based on the confusion and diffusion of the data bits, which are mainly affected by the number of rounds and S-Boxes. Moreover, the attack assumes that the attacker already have access to a number of plaintexts and their associated ciphertexts without knowing the key by which they were encrypted. By having enough number of plaintexts and their associated ciphertexts, the attacker can study the relationship between each bit of the plaintext and each bit of the ciphertext. It is assumed that if the enough number of pairs to perform the attack are available, the attacker can obtain the key immediately. The complexity of the attack increases when the number of rounds increases and when the operations inside each round gets more complicated [21]. Therefore, the number of rounds and S-Boxes directly impacts the strength of the differential cryptanalysis attack.

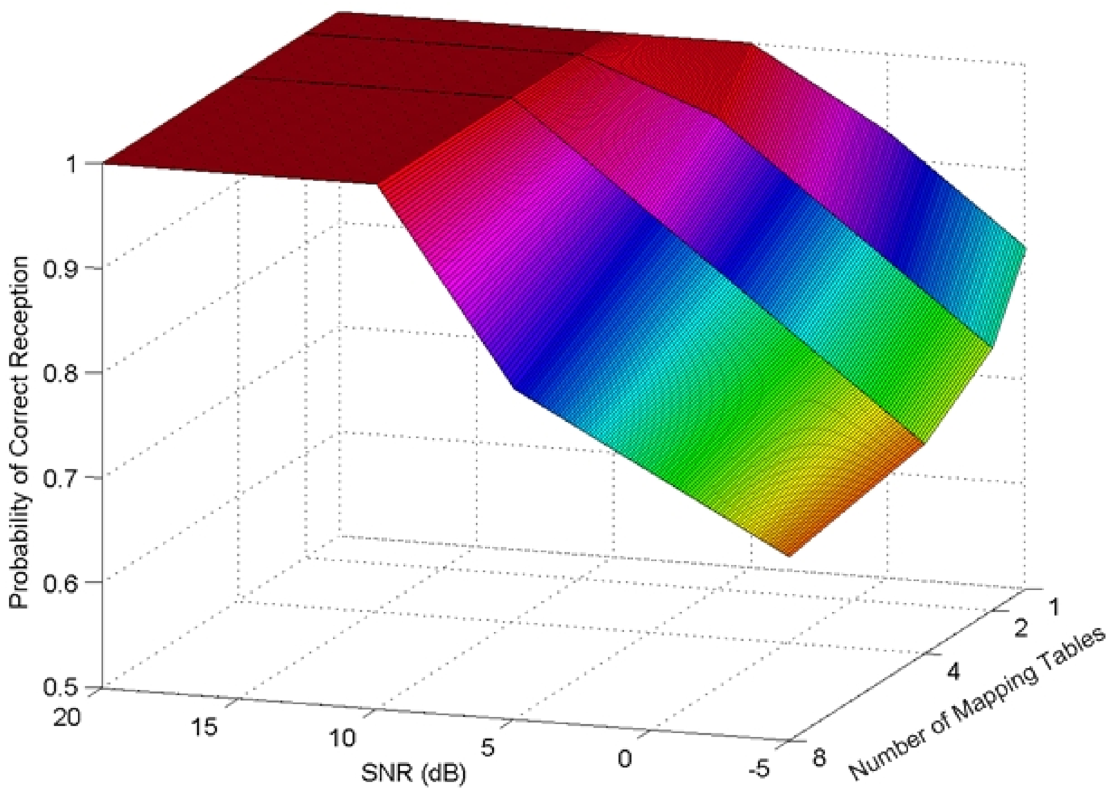


Figure 3.10. The effect of the number of distinct S-Box mapping tables and SNR on the probability of correct reception for a fixed number of rounds equal to 4

3.5.1 The Effect on DES Security

In [21] it is shown that when 2^{47} pairs of plaintexts and their associated ciphertexts are available to the attacker, the key by which these pairs were encrypted can be immediately obtained. Table 3.2 summarizes the number of pairs required to break the standard DES when different number of S-Boxes and rounds are selected in DES. For example, if we take the case where two S-Boxes and sixteen rounds are selected, 2^{13} pairs of plaintext and their associated ciphertexts are enough to break the algorithm and discover the key, compared to the case where 2^{47} pairs are required to break the algorithm when eight S-Boxes and sixteen rounds are selected. Therefore, it can be noticed that increasing the number of rounds and S-Boxes enhances the security of the encrypted data. In our analysis, we assume that if the attacker has all the required pairs required to break the algorithm using the differential

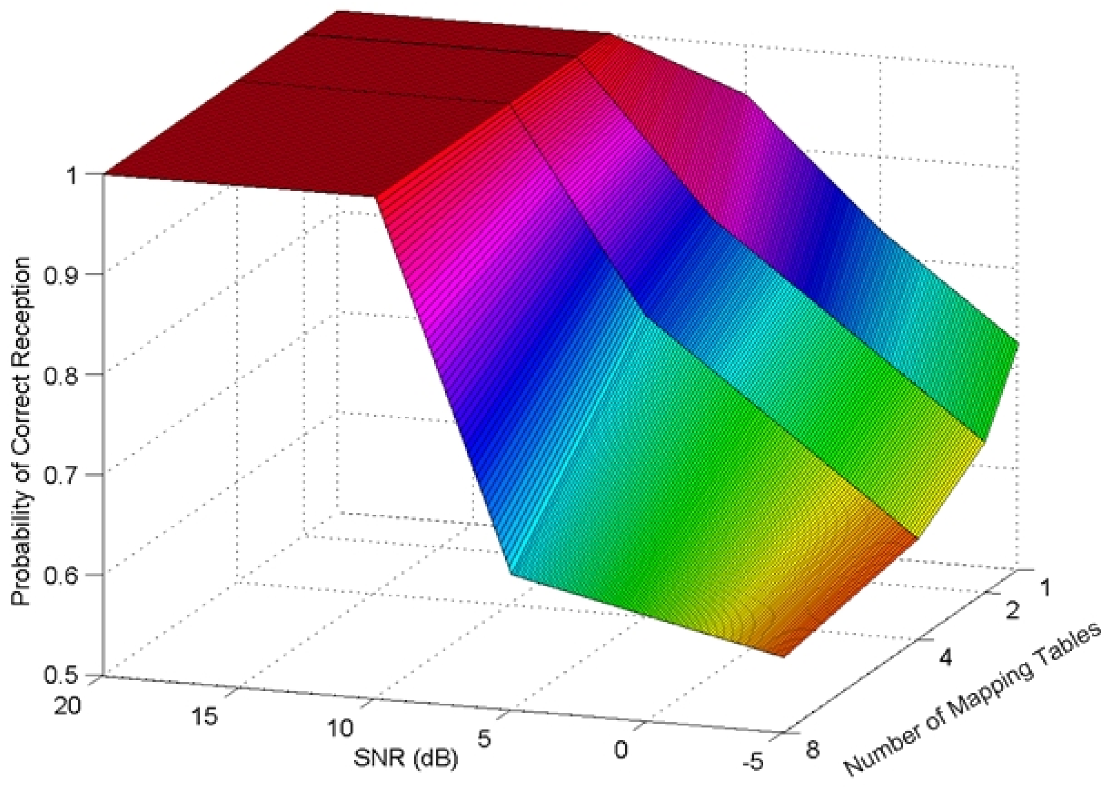


Figure 3.11. The effect of the number of distinct S-Box mapping tables and SNR on the probability of correct reception for a fixed number of rounds equal to 8

cryptanalysis attack, the algorithm can be broken in zero seconds, and the algorithm is no longer considered secure. Based on Table 3.1, given that the combination of two rounds/one S-Box is the least secure and the combination of 16 rounds/8 S-Boxes is the most secure. Let us define a normalized (w.r.t. the most secure case) security level metric as follows

$$S_d = \frac{log_2 N_p}{log_2 N_{pm}}, \tag{3.5.1}$$

where N_p is the number of pairs needed to break the algorithm for a given rounds/S-Boxes combination and N_{pm} is the maximum number of pairs needed to crack the algorithm in the most secure case, which is equal to 2^{47} pairs. Figure 3.13 shows the relationship between the number of rounds and the normalized security level S_d for different number of distinct S-Boxes (using Eq. 3.5.1). It can be noticed from the figure that the security is enhanced whenever the number of rounds or the number of S-Boxes increase. Therefore, the most

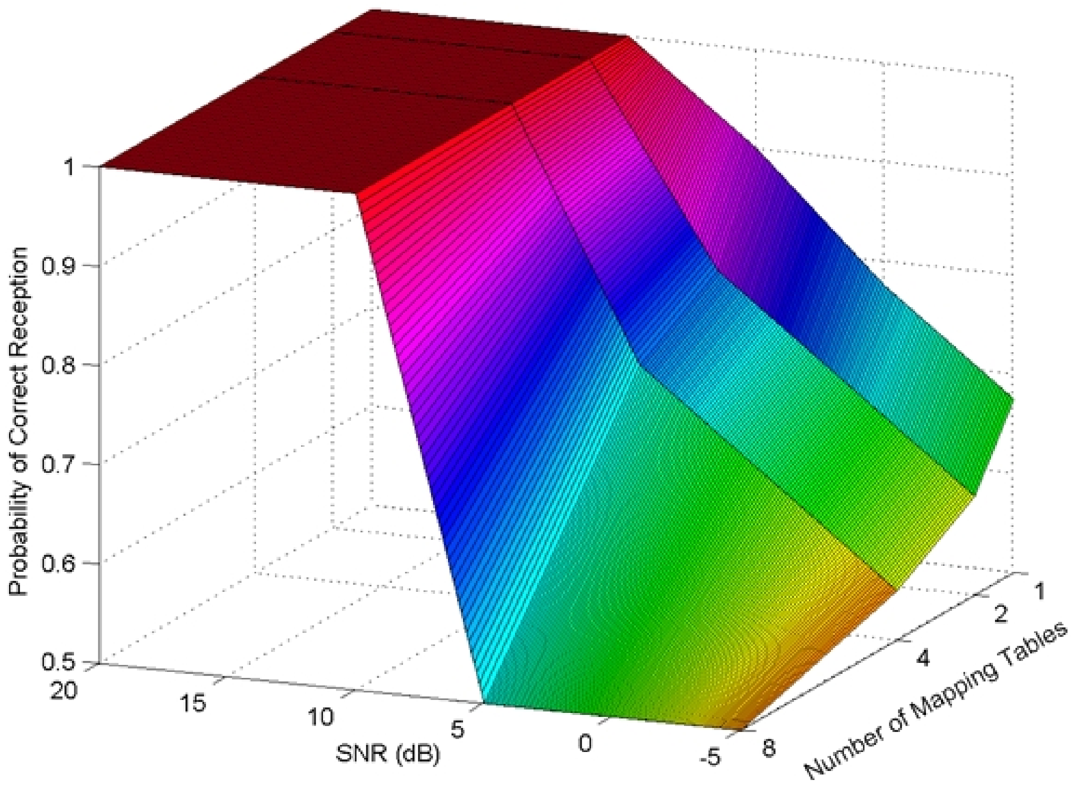


Figure 3.12. The effect of the number of distinct S-Box mapping tables and SNR on the probability of correct reception for a fixed number of rounds equal to 12

secure case ($S_d = 1$) is when we have eight S-Boxes and 16 Rounds (original DES design), conversely the worst case ($S_d = 0$) is when we have one S-Box and one round only. However, DES with its original design (16 rounds and 8 S-Boxes) has been shown to be no longer considered secure [21], although we show later in this section that its modified version M-DES proposed in [7] solves its security problem.

3.5.2 The Effect on M-DES Security

The encryption algorithm proposed in [7] and presented earlier in chapter 2 alleviates the SAC by reducing the number of S-Boxes from eight to two in addition to modifying the design of their mapping tables as well as the addition of round 17. However, to overcome the security reduction due to the dropping of six S-Boxes, the authors introduced a new

Number of rounds	1	2	3	4	5	6	7	8	9	10	11	12
M-DES	0.9	0.89	0.88	0.87	0.86	0.85	0.84	0.83	0.81	0.8	0.78	0.77
DES	0.79	0.78	0.77	0.76	0.75	0.74	0.7	0.72	0.70	0.69	0.67	0.66

Table 3.1. The probability of correct reception for M-DES (2 S-Boxes) when the SNR is equal to -5 dB and the number of rounds is varied from one to sixteen, with round 17 applied in all cases of M-DES compared to the same scenario of DES with two S-Boxes only

round with a new key. In original DES as shown in the previous table, 2^{13} pairs of plaintext and their associated ciphertext are required to attack the algorithm when two S-Boxes and sixteen rounds are chosen. However, by the introduction of round 17, the existence and availability of those pairs is not enough to break the algorithm and obtain the key. This is due to the fact that the differential cryptanalysis on DES requires both the plaintext and the ciphertext to be of a length equal to 64 bits. However, in M-DES, the 64-bit cipher is hidden in a larger 128-bit key. Therefore, assuming that 2^{13} pairs of 64-bit plaintexts and their associated 128-bit M-DES ciphertext are available to the attacker, the attack can't be directly applied until the actual 64-bit cipher is extracted from each 128-bit cipher. Hence, a new security measure can be defined by the time required to attack the algorithm. Assuming that the required pairs are available and already extracted, it is further assumed that the key is obtained immediately. Round 17 in M-DES adds an additional time overhead to perform the attack, which is the time required to extract the 64-bit useful ciphers from each 128-bit

Number of Rounds	1 S-Box	2 S-Boxes	4 S-Boxes	8 S-Boxes
2	2^0	2^0	2^2	2^3
4	2^0	2^1	2^3	2^5
6	2^1	2^2	2^5	2^8
8	2^2	2^4	2^8	2^{14}
10	2^3	2^7	2^{14}	2^{24}
12	2^4	2^9	2^{18}	2^{31}
14	2^5	2^{11}	2^{22}	2^{39}
16	2^6	2^{13}	2^{27}	2^{47}

Table 3.2. Number of pairs needed to crack DES using differential cryptanalysis for different number of rounds and S-Boxes in DES

N_r	1 S-Box	2 S-Boxes	4 S-Boxes	8 S-Boxes
2	7.9515×10^{-23}	7.9515×10^{-23}	$(7.9515 \times 10^{-23})^4$	$(7.9515 \times 10^{-23})^8$
4	7.9515×10^{-23}	$(7.9515 \times 10^{-23})^2$	$(7.9515 \times 10^{-23})^8$	$(7.9515 \times 10^{-23})^{32}$
6	$(7.9515 \times 10^{-23})^2$	$(7.9515 \times 10^{-23})^4$	$(7.9515 \times 10^{-23})^{32}$	$(7.9515 \times 10^{-23})^{128}$
8	$(7.9515 \times 10^{-23})^4$	$(7.9515 \times 10^{-23})^{16}$	$(7.9515 \times 10^{-23})^{128}$	$(7.9515 \times 10^{-23})^{2^{14}}$
10	$(7.9515 \times 10^{-23})^8$	$(7.9515 \times 10^{-23})^{128}$	$(7.9515 \times 10^{-23})^{2^{14}}$	$(7.9515 \times 10^{-23})^{2^{24}}$
12	$(7.9515 \times 10^{-23})^{16}$	$(7.9515 \times 10^{-23})^{512}$	$(7.9515 \times 10^{-23})^{2^{18}}$	$(7.9515 \times 10^{-23})^{2^{31}}$
14	$(7.9515 \times 10^{-23})^{32}$	$(7.9515 \times 10^{-23})^{2^{11}}$	$(7.9515 \times 10^{-23})^{2^{22}}$	$(7.9515 \times 10^{-23})^{2^{39}}$
16	$(7.9515 \times 10^{-23})^{64}$	$(7.9515 \times 10^{-23})^{2^{13}}$	$(7.9515 \times 10^{-23})^{2^{27}}$	$(7.9515 \times 10^{-23})^{2^{47}}$

Table 3.3. The probability of extracting all 64-bit ciphertext from the 128-bit ciphertext of M-DES in [7]

cipher and is called t_1. We find t_1 for all the cases described in Table 3.2. Assuming that the number of pairs given in an entry of Table 3.2 is called N_P, therefor the time t_1 is given by

$$t_1 = \frac{(N_1)^{N_P}}{N_2} \tag{3.5.2}$$

where N_1 is the number of operation needed to extract one useful 64-bit cipher out of one 128-bit cipher, which is given by

$$\begin{aligned} N_1 &= \binom{32}{16} \times 16! \\ &= 1.276 \times 10^{22} trials, \end{aligned} \tag{3.5.3}$$

and N_2 is the number of operations that can be performed during one second by the microprocessor. Therefore, Eq. 3.5.2 can be given by

$$t_1 = \frac{(1.276 \times 10^{22})^P}{N_2}. \tag{3.5.4}$$

As a case study, if we take a super computer that is able to run 2.5×10^{12} operations per second, eq. 3.5.4 can be given by

$$t_1 = \frac{(1.276 \times 10^{22})^P}{2.5 \times 10^{12}}. \tag{3.5.5}$$

For example, if we take the case where we have 10 rounds and 4 S-Boxes, $N_P = 2^{14}$ pairs of plaintext and their associated ciphertexts are required to crack the algorithm, however the

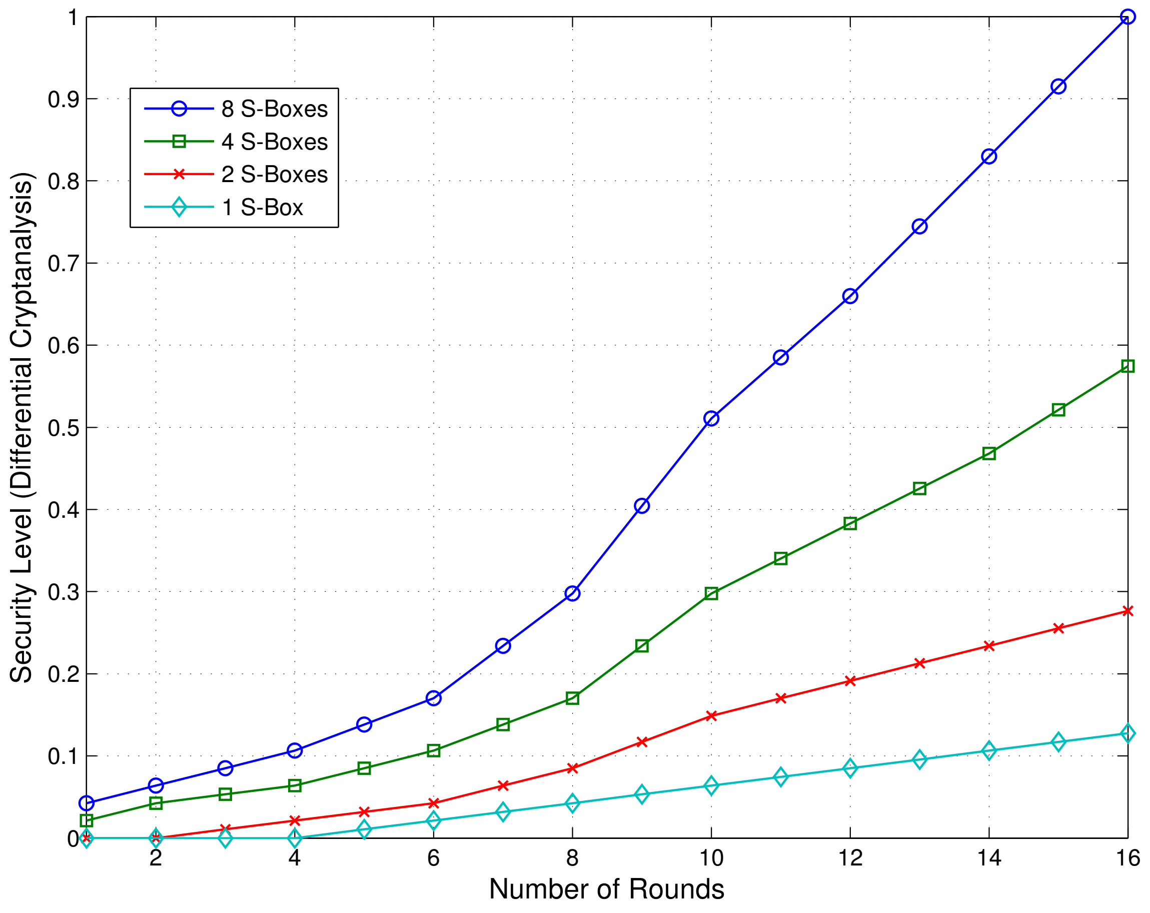


Figure 3.13. The relationship between the number of rounds and the security level for different different number of distinct S-Boxes

useful 64-bit ciphertext need to be extracted out of each 128-bit ciphertext. Hence, the time (in seconds) to extract all the useful 64-bit ciphertexts is given by

$$t_1 = \frac{(1.276 \times 10^{22})^{2^{14}}}{2.5 \times 10^{12}} = 7.236 \times 10^{362169} \qquad (3.5.6)$$

where N_{c2} is the number of clock cycles required for each extraction trial which is approximated by 32 clock cycles. Hence the number of clock cycles required is given by

$$N_{cc} = \frac{(1.276 \times 10^{22})^{P}}{32}. \qquad (3.5.7)$$

Another metric that can be used to evaluate the security is the probability of attack given that all the required pairs are available (before the extraction of the useful ciphertexts), this probability is assumed to be equal to 1 without the use of round 17 (no ciphertext extraction is required). The probability of extracting one useful ciphertext out of a 128-bit cipher is given by

$$p_1 = \frac{1}{\binom{32}{16} \times 16!} = 7.9515 \times 10^{-23}. \tag{3.5.8}$$

The experiment of guessing the useful ciphers and their correct order for all the N_P pairs in M-DES is a bernoulli trial (repeating the experiment N times out N_P ciphers). Hence, the probability of a successful attack on M-DES using differential cryptanalysis is given by

$$\begin{aligned} P_{attack} &= \binom{N_P}{N_P} p_1^{N_P} (1 - p_1)^{N_P - N_P} \\ &= p_1^{N_P} \\ &= (7.9515 \times 10^{-23})^{N_P}. \end{aligned} \tag{3.5.9}$$

Therefore, the probability of attack decreases when the number of pairs required in Table 3.2 increase (increased number of rounds and S-Box mapping tables). For example if we take the same case we discussed earlier of the 10 rounds with four S-Boxes, the probability of attacking the algorithm (extracting all useful ciphertexts) is given by

$$P_{attack} = (7.9515 \times 10^{-23})^{2^{14}} \tag{3.5.10}$$

which is a very small probability. Table 3.3 summarizes the probability of attack for all cases described in Table 3.2.

To summarize, in this section we have shown that the security of the encrypted data decreases when the number of rounds and S-Boxes decreases. We also showed that the addition of round 17 in M-DES [7] significantly improves the security by increasing the time required for a successful attack which implies a very low probability of successful attack on M-DES. In addition, we evaluated the security enchantment of the new round proposed in M-DES in a more comprehensive way, where we numerically evaluated the probability of a successful attack on the proposed algorithm as well as calculating the time required for such an attack.

3.6 Conclusions

In this chapter we evaluated the effect of the number of rounds and the number of S-Boxes on the probability of correct reception assuming different channel conditions. In addition, we also evaluated the effect of the number of rounds and the number of S-Boxes on the security level of the encrypted data in DES as well as in M-DES. We have shown that the probability of correct reception is improved when the number of rounds and S-Boxes is decreased. Moreover, we have shown that the security of the encryption algorithm is enhanced when the number of rounds and S-Boxes is increased. We have also evaluated the security enhancement of the new round proposed in M-DES in a more comprehensive way, where we numerically evaluated the probability of a successful attack on the proposed algorithm as well as calculating the time required for such an attack. We have shown that the time to achieve a successful attack on M-DES is tremendously high given that a very fast super computer is used in the attack, which implies an insignificant probability of such an attack on M-DES.

Chapter 4

Adaptive M-DES: A Trade-Off between Security and Throughput

4.1 Introduction

Encryption in wireless channels is becoming more and more important and challenging process. Encrypted information might encounter some amount of error at the receiver side due to SAC. In [3]-[5] the authors developed effective coding schemes and mechanisms in the physical layer, in order to correct and minimize the errors caused by the channel. While in [6], [8] the authors were more focused on inventing and implementing new encryption algorithms that encounter less error propagation, and hence alleviating the SAC. In [6] a new mode of operation is used with the use of DES to transmit the encrypted data over the wireless channel. While in M-DES introduced earlier, a new encryption algorithm is introduced, which is a modification to the Data Encryption Standard (DES) and is shown to outperform the standard DES when implemented in wireless channels, moreover, the proposed algorithm is shown to outperform the used mode of operation in [6]. In [8], the authors developed an optimized framework for encryption over wireless channels using the Rijndael algorithm. The proposed mechanism in [8] is mainly based on using longer block and key sizes whenever the channel conditions are better.

In this chapter we use the mechanism presented in [8] and apply it to an adaptive variable block size version of the modified DES. The proposed algorithm has variable key and cipher sizes rather than the fixed 136-bit key and 128-bit cipher used in M-DES. Analytical results are used to show that the proposed algorithm outperforms the use of Rijndael as well as the 256-bit AES algorithm, in terms of throughput for a fixed required security level.

4.2 Motivation

A secure encryption algorithm could be designed with a very large block and key sizes to ensure the secrecy of the encrypted information over the channel. However, the throughput of channel using any encryption algorithm should be taken into account to ensure an efficient use of the channel resources. In this chapter we maximize the throughput of the channel which was ignored earlier in the design of M-DES.

4.3 Contributions

In this chapter we propose a new encryption algorithm based on M-DES, that takes into account the throughput of the channel. This algorithm uses an encryption block and key sizes such that the throughput is maximized for a specific security level. Therefore, for a certain required security level the throughput of the channel is maximized, which ensures that the security constraints are met and the throughput is optimized for that required security.

4.4 Adaptive M-DES Design and Architecture

In M-DES, a new round called round 17 was added to a modified version of the standard DES. The new round uses the output of the standard DES (with some modification in the S-Boxes mapping tables) and a new 80-bit key to generate a ciphertext of 128 bits. However, the bandwidth efficiency is decreased because the new design requires sending 128 bits to transmit an actual 64 bits of useful information. The 64-bit useful ciphertext (i.e., output of round 16) is mapped into a 128-bit ciphertext using a new 80-bit key. The 64-bit output of round 16 is divided into 16 sub-frames of 4 bits each, and each of those sub-frames is mapped to one of the 32 output sub-frames (i.e., $128/4 = 32$) using 5 bits of the key. The remaining 16 sub-frames of the output are randomly filled with ones and zeros. Therefore, the M-DES is using 50% of the bandwidth to transmit encrypted information.

In this work, we will further introduce two modifications to the M-DES, by making the length of the ciphertext variable rather than being fixed and equal to 128 bits. While in the other modification, we will be making use of the wasted space in round 17 by using the same ciphertext for more than one block of the plaintext. To illustrate the modifications, we assume an example where the channel conditions are fixed on a certain Signal-to-Noise ratio (SNR), and the optimal ciphertext size required to maintain a certain security level is 138 bits. In [8], the 138-bit block size is not available, therefore the 160-bit block size is used. However, in our case the 138 bits should be rounded to the first number that divides 4 (i.e., because each sub-frame is 4 bits long). Therefore, the ciphertext size that M-DES is going to use is 140 bits rather than 160 bits in the case of [8]. However, the algorithm used in [8] encrypts a 160-bit plaintext into a 160-bit ciphertext, but for M-DES, a 64-bit plaintext is encrypted to a 140-bit ciphertext. Therefore, the algorithm in [8] uses 100% of the channel bandwidth while M-DES only uses 46 % of the same bandwidth, due to the fact the 16 sub-frames (i.e., output of round 16) are hidden in the 35 output sub-frames (i.e., $35 = 140/4$), while the remaining $35 - 16 = 19$ sub-frames are filled with random zeros and ones. This is how M-DES would handle the encryption. However, by introducing the second modification in this work we will make use of the remaining 19 sub-frames to include the 16 encrypted sub-frames of the next block (i.e., encrypted with the first 16 rounds only, therefore the output would be 64 bits at this point) and the first 3 sub-frames of the third block. Therefore, the algorithm is now using 100 % of the bandwidth to encrypt a 140-bit plaintext into a 140-bit ciphertext. So, although the algorithm in [8] is using the bandwidth efficiently, it is not using the exact optimal block size that depends on the channel conditions and the required security level, while in the new optimized M-DES we use a very close block size to the optimal block size (i.e., maximum difference from the optimal block size is 3 bits only, because we need a number that divides four without a remainder).

To handle these modifications, more than one key need to be introduced to round 17 in order to map the sub-frames of the different blocks. Assuming the same example where a 140-bit cipher is used, the cipher consists of 35 sub-frames, therefore, two blocks and the first 3 sub-frames of the third block can fit in the 140-bit cipher. In this case three blocks are mapped to the same cipher, so three keys are required for the mapping. The first block will have the option to choose 16 different keys from the 35 key pool 000000 to 100010 to map its 16 sub-frames. While the second block will have the option to choose 16 different keys from the remaining $35 - 16 = 19$ key pool, and the third block will have the remaining $35 - 16 - 16 = 3$ keys. Theses three keys will be concatenated together along with the original 56-bit key of DES. Figure 4.1 shows the block diagram for the proposed modification introduced to M-DES. The figure explains the example above, where the cipher size is 140-bit. This cipher can fit two blocks and the first three sub-frames of the third block.

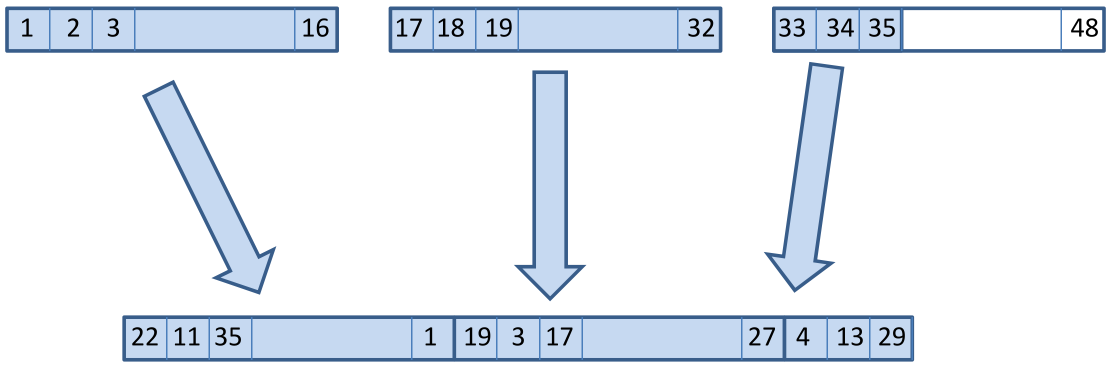


Figure 4.1. Block diagram of the modifications introduced to M-DES, which makes more flexible and adaptable to the optimal cipher size. In this example, the cipher size is 140-bit

4.5 Security Enhancement of the Proposed Algorithm

In this section, we provide the analysis of the security enhancement of the new modifications in the proposed algorithm, in terms of its strength to brute force and differential cryptanalysis attacks.

4.5.1 Against Brute Force Attack

By introducing the new keys of round 17 and by concatenating those keys with the original key of DES, the algorithm is more immune against this attack, which is based on trying all possible keys to decrypt a ciphertext. Assuming that the 56-bit key of the standard DES can be obtained in one second using this attack, by a simple calculation the attacker needs 2^{39} years to obtain the key in the worst case scenario where the ciphertext size is 64 bits (i.e., 64-bit input of round 17 is mapped to a 64-bit output of the round).

4.5.2 Against Differential Cryptanalysis Attack

In M-DES, it was illustrated that 2^{13} pairs of plaintexts and their corresponding useful ciphertexts (i.e., before entering round 17) are required to break M-DES using this attack, however, the probability of getting one useful pair (i.e., 64-64 pair) from the 64-128 pair is given by

$$p = \frac{1}{\binom{32}{16} \times 16!} = 7.9515 \times 10^{-23}, \tag{4.5.1}$$

where the 32 is the number of sub-frames of the 128-bit cipher, that are used to hide the 16 sub-frames of the 64-bit useful cipher. Therefore, the probabilities of getting one useful pair for a cipher of N sub-frames

$$p_1 = \frac{1}{\binom{N}{16} \times 16! \times (N-16)!} \tag{4.5.2a}$$

$$p_2 = \frac{1}{\binom{N}{16} \times 16! \times 16!(N-32)!} \tag{4.5.2b}$$

$$p_3 = \frac{1}{\binom{N}{16} \times 16! \times 16! \times 16!(N-48)!} \tag{4.5.2c}$$

$$p_n = \frac{1/(16!)^{n+1}}{\binom{N}{16} \times \binom{N-16(n+1)}{16}^{n} \times (N-16(n+1))!}, \tag{4.5.2d}$$

where (4.5.2a) is for $16 \leq N \leq 32$, (4.5.2b) is for $32 \leq N \leq 48$, (4.5.2c) is for $48 \leq N \leq 64$ and (4.5.2d) is for any given N. Two, three, four, $n+1$ ciphers can be fitted in (4.5.2a), (4.5.2b), (4.5.2c) and (4.5.2d), respectively. It can be clearly noticed from the equations that when the cipher size increases, the probability of breaking the algorithm using differential cryptanalysis attack will decrease significantly.

4.6 Throughput-Security Trade-offs

In this section, we present the throughput-security problem defined in [8] and we apply it to our proposed algorithm. The problem objective is to maximize the overall throughput while maintaining a minimum required security level. It is assumed that the transmitter has a knowledge of the channel state information (CSI) over a super frame duration, which consists of n sub-frames, where the throughput per sub-frame is given by $R_i(1-P_i)^{N_i} \approx R_i(1-N_iP_i)$ for $P_i << 1$ for a given and fixed N_i. The normalized throughput and security level of the system can be respectively, defined by

$$T = \frac{1}{nR_{max}} \sum_{i=0}^{n} R_i(1-N_iP_i) \tag{4.6.1}$$

$$S = \frac{1}{nS_{max}} \sum_{i=0}^{n} log_2 N_i, \tag{4.6.2}$$

where R_i is the transmission rate, N_i is the encryption block length, R_{max} is the maximum transmission rate among all sub-frames, S_{max} is the maximum security level which is defined by $S_{max} = log_2 N_{max}$ where N_{max} is the maximum ciphertext length, P_i is the channel bit error probability and n is the number of the sub-frames in the input data stream. We calculate the optimum encryption block size N for each sub-frame and will be the same during that sub-frame. As stated earlier, the security of the system increases whenever the block length increases. The problem here is to maximize the throughput across the whole data stream (all sub-frames) while maintaining a certain required level of security and to be within the allowable encryption block sizes which are from 64 bits to 256 bits. Therefore, the optimization problem can be formulated as

$$\max_{N_i} \frac{1}{nR_{max}} \sum_{i=0}^{n} R_i(1 - N_i P_i)$$

$$\textit{subject to} \tag{4.6.3}$$

$$\frac{1}{nS_{max}} \sum_{i=0}^{n} log_2 N_i = S_{req} \tag{4.6.4}$$

$$\textit{and} \tag{4.6.5}$$

$$64 \leqslant N_i \leqslant 256 \tag{4.6.6}$$

where S_{req} is the required level of security. It should be noted that P_i is a function of the channel Signal to Noise Ratio (SNR). The optimization problem can be transformed to an unconstrained problem using Lagrange multipliers method. Using this method, the object function can be written as

$$C = \frac{1}{nR_{max}} \sum_{i=0}^{n} R_i(1 - N_i P_i) + \lambda\left(\frac{1}{nS_{max}} \sum_{i=0}^{n} log_2 N_i - S_{req}\right) \tag{4.6.7}$$

where λ is the lagrange multiplier. By taking partial derivatives with respect to N_i and setting the result to zero, we obtain

$$N_i^* = \frac{(\Pi_{i=1}^{n} R_i P_i)^{\frac{1}{n}}}{R_i P_i} e^{(S_{max} S_{req}) log_e 2} \tag{4.6.8}$$

where N_i^* is the optimized block length.

For each sub-frame, the channel condition are assumed to be well-known, hence the SNR of the channel and P_i are known as well. Therefore, the optimal ciphertext block size is calculated based on the solution provided in (4.6.8). Afterwards, the sender divides the sub-frames into blocks of 64 bits each and encrypt each into its 64-bit cipher (before round 17). Multiple 64-bit cipher are scrambled into the N_i^*-bit optimal ciphertext block. The sender then transmits the N_i^*-bit optimal ciphertext blocks for that specific sub-frame. At the receiver side, the receiver will have the keys that were used in the scrambling and he will be able to reconstruct the 64-bit cipher in order to decrypt them into their corresponding 64-bit plaintexts. The steps of the algorithm are stated in algorithm 1.

```
Algorithm 1: Optimized M-DES
  Input: Data stream as plaintext
  Output: Data stream as ciphertext
  Divide the data sequence into $n$ frames;
  foreach frame $n_i$, the channel conditions are assumed to
  be fixed and well known do
      Divide the frame $n_i$ into blocks of 64 bits each;
      Compute the optimal ciphertext size $N_i^*$ based on
      equation 8;
      Round the optimal ciphertext size up to the closest
      number that divides 4;
      Encrypt all of the 64-bit plaintexts into their 64-bit
      cipher (before round 17);
      Fit the created 64-bit ciphers into the $N_i^*$-bit ciphers;
  Repeat for other frames
```

4.7 Numerical Results and Discussions

In this section, we use Numerical Results to evaluate the performance of the proposed algorithm in a wireless fading channel. We assume that the channel fading gain (α) follows a Rayleigh distribution with a 1/5 block coding rate and a binary phase shift keying (BPSK) modulation. he normalized throughput of the channel is computed based on (4.6.1), where

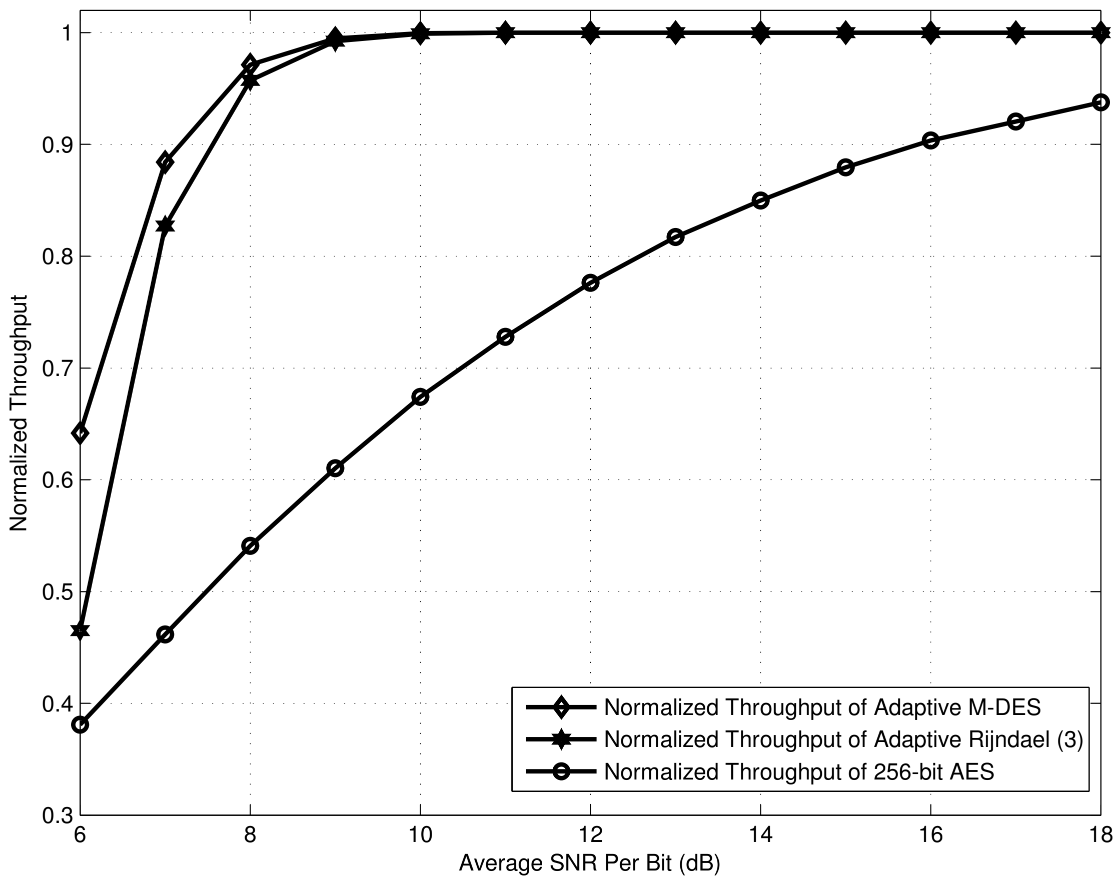


Figure 4.2. Normalized throughput of the proposed algorithm as compared to the optimized Rijndael introduced in [8] and the fixed 256-bit AES for a security level of $S_{req} = 0.975$, in Rayleigh fading channel with a block code rate of 1/5.

P_i is the bit error probability for the n-coded BPSK with rayleigh fading channel and is given in [47] by

$$P_i = \left(\frac{1-\tilde{\sigma}}{2}\right)^n \sum_{q=0}^{n-1} \binom{n-1+q}{q} \left(\frac{1+\tilde{\sigma}}{2}\right)^q, \tag{4.7.1}$$

where $\tilde{\sigma}$ is given by

$$\tilde{\sigma} = \sqrt{\overline{\gamma_b}/(\overline{\gamma_b}+n)} \tag{4.7.2}$$

and n is the channel block coding rate and is equal to 5. While $\overline{\gamma_b}$ is given by

$$\overline{\gamma_b} = \overline{\alpha^2} E_b/N_0, \tag{4.7.3}$$

where E_b/N_0 is the average signal-to-noise ratio (SNR) with a range of 0 to 35 dB. The average fading power α is assumed to be equal to one. Results of the normalized throughput versus the average SNR per bit for adaptive M-DES, adaptive Rijndael, and fixed length 256-bit AES are depicted in Figure 4.2.

The figure shows the normalized throughput obtained from the use of our proposed mechanism as compared to the throughput obtained from using the same optimization mechanism over the variable length Rijndael algorithm proposed in [8] as well as to the throughput of using the fixed length 256-bit AES encryption. It is clearly noticed from the figure that our proposed algorithm outperforms the proposed algorithm in [8], in addition, it significantly outperforms the use of the fixed 256-bit AES algorithm. Moreover, our algorithm is shown to be more powerful when the signal-to-noise ratio is low (worst channel conditions, i.e., lower SNR values). It should be noted that the proposed algorithm requires more computational time; however, given fast microprocessors available today this computational time is insignificant.

4.8 Conclusions

In this chapter we proposed a new throughput-security optimized encryption algorithm where we introduced a variable ciphertext size for encryption and we encrypt multiple blocks of the plaintext into one ciphertext. The ciphertext size was chosen optimally to achieve the maximum throughput while maintaining a required certain security level. We were able to chose a ciphertext size that is very close to the optimal size specified by the optimization problem, unlike other algorithms where the size should be rounded up to the closest size provided by the algorithm. In addition, we analyzed the security of the proposed algorithm in terms of its strength to applicable attacks, and we showed how the new mapping of multiple encrypted blocks into the same ciphertext had strengthened the security of the algorithm. Moreover, we showed that the proposed algorithm outperforms the use of the variable block length Rijndael and the fixed block length 256-bit AES encryption algorithms, over the same optimization framework, for the same required security level.

Chapter 5

Key-Based Coded Permutation Modified-DES (KBCP-M-DES)

5.1 Introduction

In this chapter we propose a new key based coded permutation box at the output of DES instead of round 17, while keeping the other modifications M-DES introduced to DES. In the KBCP box we make use of the empty blocks of round 17 to implement channel coding. Channel coding techniques are common modern techniques that are used for error detection and correction over wireless channels. Since M-DES uses a 128-bit cipher, out of which 64 bits only hold useful information, the remaining 64 bits can be used to implement channel coding, which significantly improves the probability of correct reception, and make use of the wasted bandwidth, however, the complexity of the encryption algorithm may be increased. We call the proposed algorithm KBCP-M-DES.

5.2 Motivation

Our motivation for the work in this chapter is based on the fact that we are interested in using the available space in the M-DES cipher to increase the confidentiality and reliability of the transmitted information, by using the remaining space to implement channel coding techniques and/or adding other keys to increase the confusion of the encrypted blocks.

5.3 Contributions

In this chapter, we propose what we call the Key-Based Coded Permutation Modified-DES where we use hamming codes in the remaining space of M-DES last round. In another design we add another key to map the coded blocks which increase the confusion of the encrypted information and hence increases the security of the algorithm. Therefore, in this chapter, we include new techniques that makes use of the remaining space of M-DES to implement channel coding in addition to using a new key. By these modifications, we experience more reliable data transmission and enhanced security. Therefore, in this chapter we propose a new algorithm that utilizes the throughput of the channel by introducing channel coding without an additional cost to the size of the cipher, by making use of the empty space in M-DES. Furthermore, a third key is introduced to map the blocks that hold the codes which significantly enhances the security of the algorithm.

5.4 Review of the Hamming Codes

We are going to use the hamming (7,4) coding technique with an additional parity bit. This techniques adds 3 control bits to a 4 data bits with an additional bit for parity [44]. Therefore, this technique is perfect for round 17 structure where the 64-bit input of the round is divided into 16 sub-frames, of four bits each. Hence, for each one block of information (4 bits), we will generate one block of control (3+1 bits), rather than just having a random block entries for each block of data as the case of round 17.

Hamming codes were invented in 1950 by Richard Hamming, and it can detect two errors at lease in each block and is able to correct at least one bit in hamming (7,4) technique. Hamming codes are widely used in many applications such as computer memory RAM due to their simplicity and effectiveness in detecting and correcting errors. Being able to detect two bit errors and correct one bit error in each 4-bit block is very effective and sufficient enough to be used in our analysis and simulations. For example, in the worst channel conditions of our assumptions where SNR can be as low as -5 db, the probability of error would be around 0.3 theoretically, which implies that 1-2 bits will be in error out of each 4-bit block under these conditions. However, in better channel conditions the probability of encountering a single error within 4 bits decreases significantly, which makes the use of hamming (7,4) the simplest, fastest and most efficient to be used in our application.

Hamming (7,4) code is based on adding three parity bits p_1, p_2, p_3 to 4 bits of data d_1, d_2, d_3, d_4. Each parity bit depends on three data bits out of the four, such that if the number of locations that holds a value of one out of those three locations is an odd number, the parity bit will be one. So p_1 is set to 1 if the bits d_1, d_2 and d_4 have an odd number of ones, while it set to zero if the number of ones is even. In a similar way p_2 depends on data bits d_1, d_3 and d_4 and p_3 depends on data bits d_2, d_3 and d_4. The following table summarizes theses dependencies. To increase the security of the code the three bits of parity are not

Table 5.1. Parity bits dependencies

	d_1	d_2	d_3	d_4
p_1	Yes	Yes	No	Yes
p_2	Yes	No	Yes	Yes
p_3	No	Yes	Yes	Yes

simply added to the four data bits, instead it is placed in between the data bits according to the following table.

Table 5.2. Hamming (7,4) code word structure

p_1	p_2	d_1	p_3	d_2	d_3	d_4

Because hamming codes are linear codes, linear algebra can be used to compute the hamming (7,4) code. The hamming matrix can be defined by two matrices, G the generator matrix and H the parity check matrix. Where G and H are given by

$$\begin{aligned} G &= \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\ H &= \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} \end{aligned} \tag{5.4.1}$$

Assuming that the data bits are ordered in a vector $p = (1, 0, 1, 1)$, to construct a codeword x

$$x = Gp = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \\ 3 \\ 1 \\ 2 \\ 0 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} \tag{5.4.2}$$

where the results are taken in modulo 2 format to produce zeros and ones. To check if an error has occurred in the received coded word, the received word is multiplied by the matrix H

$$z = Hr = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \\ 4 \\ 2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}.$$

5.5 KBCP-M-DES Design and Architecture

In this section we show the design and architecture of the key based coded permutation box (KBCP), where we implement two different designs based on the level of security required and based on the available resources.

5.5.1 KBCP-M-DES with One Key

Motivated by the idea of round 17 of M-DES in 2, we introduce the KBCP by introducing channel coding in the unutilized blocks of round 17. Transferring from round 17 to KBCP, the empty blocks of round 17 are used to hold useful coded information rather than being filed randomly as in the case of M-DES. In M-DES, the output of round 16 consists of 64 bits of encrypted information which are divided into 16 blocks of four bits each. Using a key of 80 bits, each five bits of the key are used to map each 4-bit block of the encrypted information to a new block of the 32 output blocks.

In KBCP, these 16 blocks ($B_1, B_2 .... B_{16}$) go through an extra operation before being mapped to the 32 block output. Each block of the input is coded using hamming (7,4) coding technique into a 7-bit codeword with an additional parity bit making the total size of the codeword of 8 bits. Therefore, 16 codewords ($C_1, C_2 .... C_{16}$) each of 8 bits will be generated at this stage with a total size of 128 bits.

Each of the sixteen 8-bit codeword is then divided into two smaller blocks a and b, with 4 bits in each block (C_{ia}, C_{ib}), where $i = 1$ to 16. Each five bits of the 80-bit key are used to map the a blocks of the codewords $C_{1a}, C_{2a} .... C_{16a}$ into their new location in the 128-bit output that consist of 32 blocks of 4 bits each. At the end of this step, 16 blocks of the output are full with the a blocks of the codewords, while 16 blocks are still empty. These empty blocks will hold the b blocks of the codewords $C_{1b}, C_{2b} .... C_{16b}$ one by one without being mapped by a key. Meaning that C_{1b} will be located in the first empty block, C_{2b} will be located in the second empty block and so on. At the receiver side, the 80-bit key is

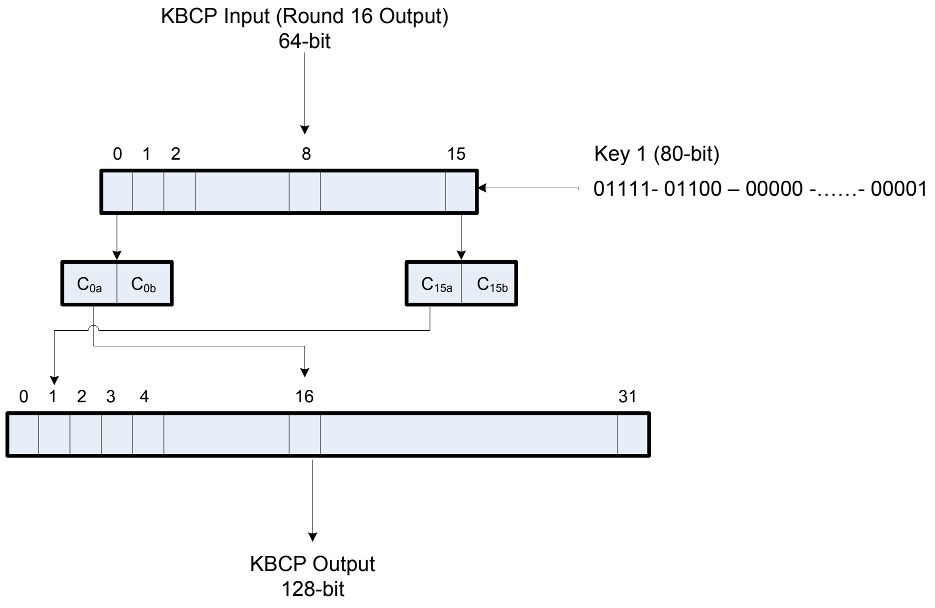


Figure 5.1. The block diagram for the KBCP with one key, where each 5 bits of the key is responsible of mapping the C_{ia} blocks to their new locations. While the C_{ib} blocks are mapped to the empty blocks according to their order. The input to the KBCP box is coming from round 16 output of DES after the modifications we introduced (i.e, The first four S-Boxes uses a similar mapping tables which are the same as the first mapping table of the standard DES, while the fifth through the eighth S-Boxes use similar mapping tables that are the same as the fourth S-Box mapping table in the standard DES)

used to locate $C_{1a}, C_{2a}....C_{16a}$, the remaining blocks that were not mapped by the key are $C_{1b}, C_{2b}....C_{16b}$ based on the order they are located with. Figure 5.1 shows the block diagram for KBCP with one key.

5.5.2 KBCP-M-DES with Two Keys

In this section another 80-bit key is introduced to map the b blocks of the codewords to the empty 16 locations of the output rather than just allocating those blocks according to their order within the empty blocks. However, the selection of the second key can't be random within the locations, it should be chosen such that it takes the empty 16 blocks only. This will add one more restriction on selecting the second key, while the first key only had one restriction that each 5-bit block of the key should be unique. Therefore, rather than locating the b blocks of the coded words in one single way (i.e, according to their order as

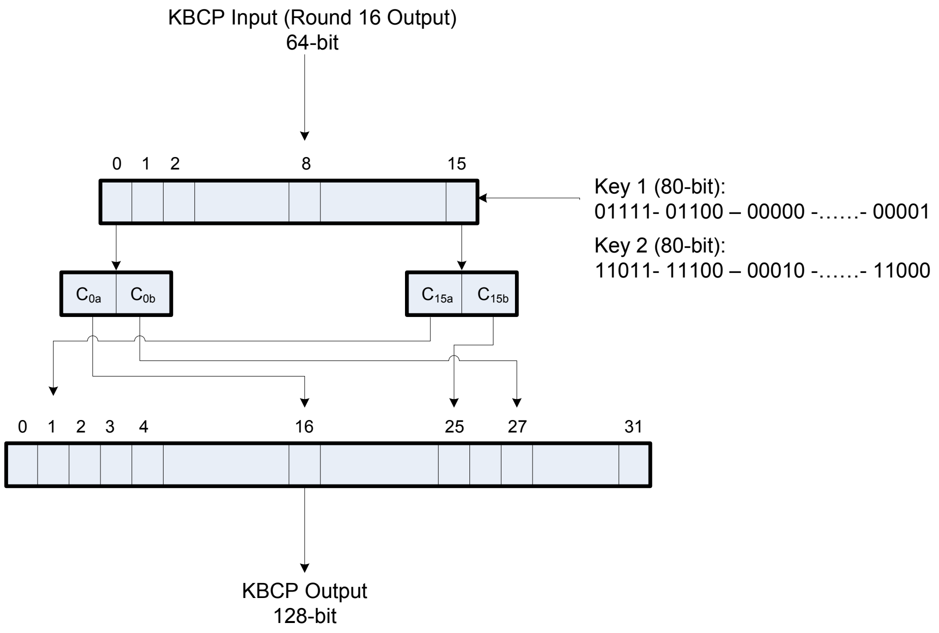


Figure 5.2. Block diagram for the KBCP with two keys, where each 5 bits of the first key is responsible of mapping the C_{ia} blocks to their new locations. While each 5 bits of the second key is responsible of mapping the C_{ib} blocks to the remaining locations. The input to the KBCP box is coming from round 16 output of DES after the modifications we introduced (i.e, The first four S-Boxes uses a similar mapping tables which are the same as the first mapping table of the standard DES, while the fifth through the eighth S-Boxes use similar mapping tables that are the same as the fourth S-Box mapping table in the standard DES)

shown in the previous sub section), there would be 16! ways to place those blocks within the 16 empty blocks of the output. Although the addition of this new key will not affect the error performance due to the fact that nothing is changed in the coding mechanism, it will significantly improves the security of the algorithm by making it harder for the attacker to figure out the location of the other half of each coded word as we will show later in this chapter. On the other hand the complexity of the two key version of KBCP will add to the complexity of the algorithm in terms of the number of asthmatic operations and memory accesses as we will show in next chapters. Figure 5.2 shows the block diagram for KBCP-M-DES with two keys.

5.6 Error Performance of KBCP-M-DES

In this section we show the performance of the proposed algorithm on the bit error rate (BER) performance. By introducing channel coding in KBCP, the error at the receiver is expected to be improved and hence the error after decryption is expected to be improved as well. The hamming code is able to discover and correct errors. The error performance of KBCP-M-DES is the same in both cases of one key and two keys, because the addition of the second key does not enhance the coding mechanism, it only increases the security of the cipher. Using computer simulations, the performance of the proposed algorithm on the error performance is evaluated as shown in Figure 5.3. We assume that the channel is an additive white gaussian noise (AWGN) channel with binary phase shift keying (BPSK) modulation. We encrypt a large amount of information using AES, DES, M-DES, AES with hamming (7,4) channel coding, DES with hamming (7,4) soft decision channel coding and the proposed KBCP-M-DES, then the encrypted information are transmitted over the channel and decrypted at the receiver side. The error is computed by comparing the data after decryption at the receiver side with the data before encryption at the transmitter side.

We compare the effect of the KBCP-M-DES versus AES, DES and versus the original M-DES. It can be noticed from the figure that the use of the proposed algorithm outperforms the use of AES and DES with hamming (7,4) channel coding as well as M-DES. Figure 5.4 shows the effect of using block code rate of 1/2 rather than the hamming(7,4) code. It can be noticed that the hamming codes degrades the error performance for low values of SNR, while the 1/2 block code enhances the performance for all SNR ranges.

5.7 Security of KBCP-M-DES

The security of the proposed encryption algorithm is measured by the attackers ability to crack it. There are five general types of attacks that may be applied to any encryption

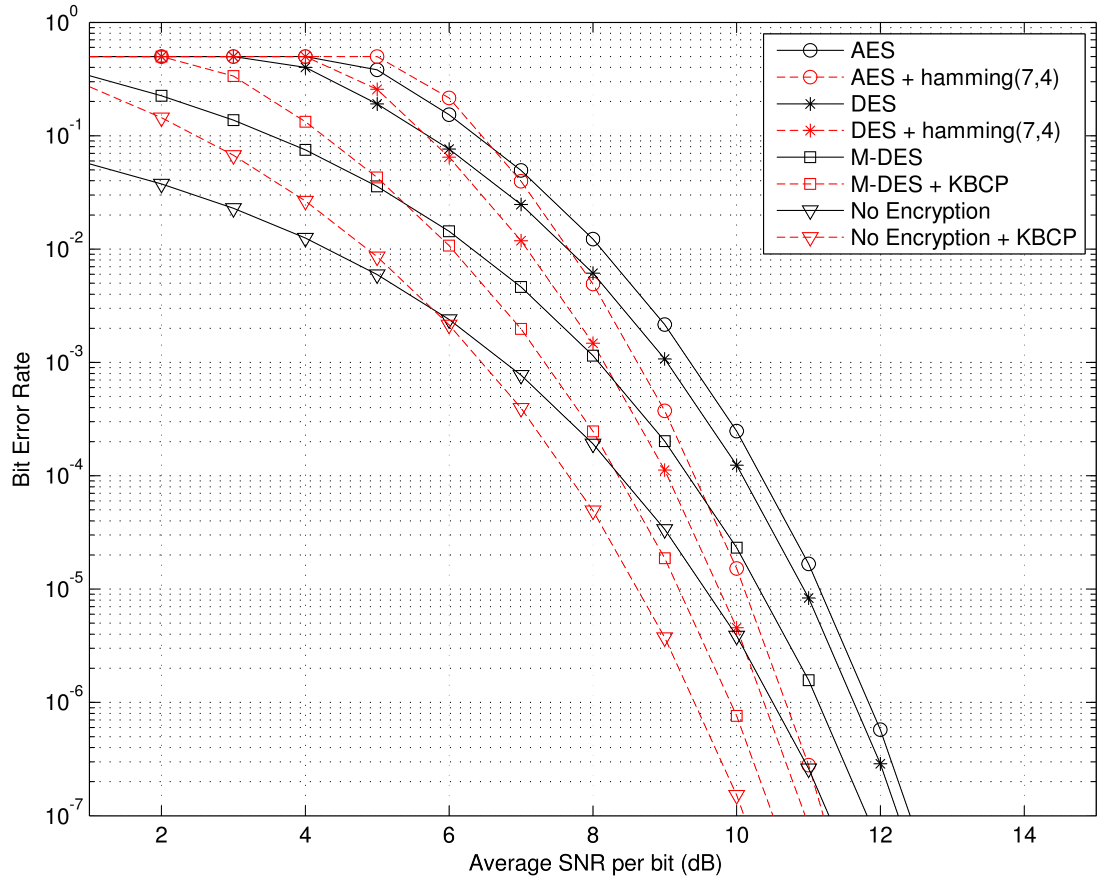


Figure 5.3. BER vs. SNR in AWGN channel with BPSK modulation, comparison of using AES, DES, M-DES (no channel coding), AES with hamming (7,4) channel coding, DES with hamming (7,4) channel coding and the proposed encryption algorithm KBCP-M-DES

algorithm: ciphertext only, known plaintext, chosen plaintext, chosen ciphertext and chosen text attacks. However, in symmetric block cipher algorithms the first and second types are the most applicable attacks [19]. The most known example of the first type is the brute force attack which is only successful on short keys. While the differential cryptanalysis is an example of the second type of attacks, which we are going to study the possibility of such an attack on the proposed algorithm.

In this section we study the effect of the differential cryptanalysis attack on KBCP. Starting by the case where only one key is used in KBCP, the security of the algorithm is not upgraded or downgraded compared to the security of M-DES. This is due to the fact

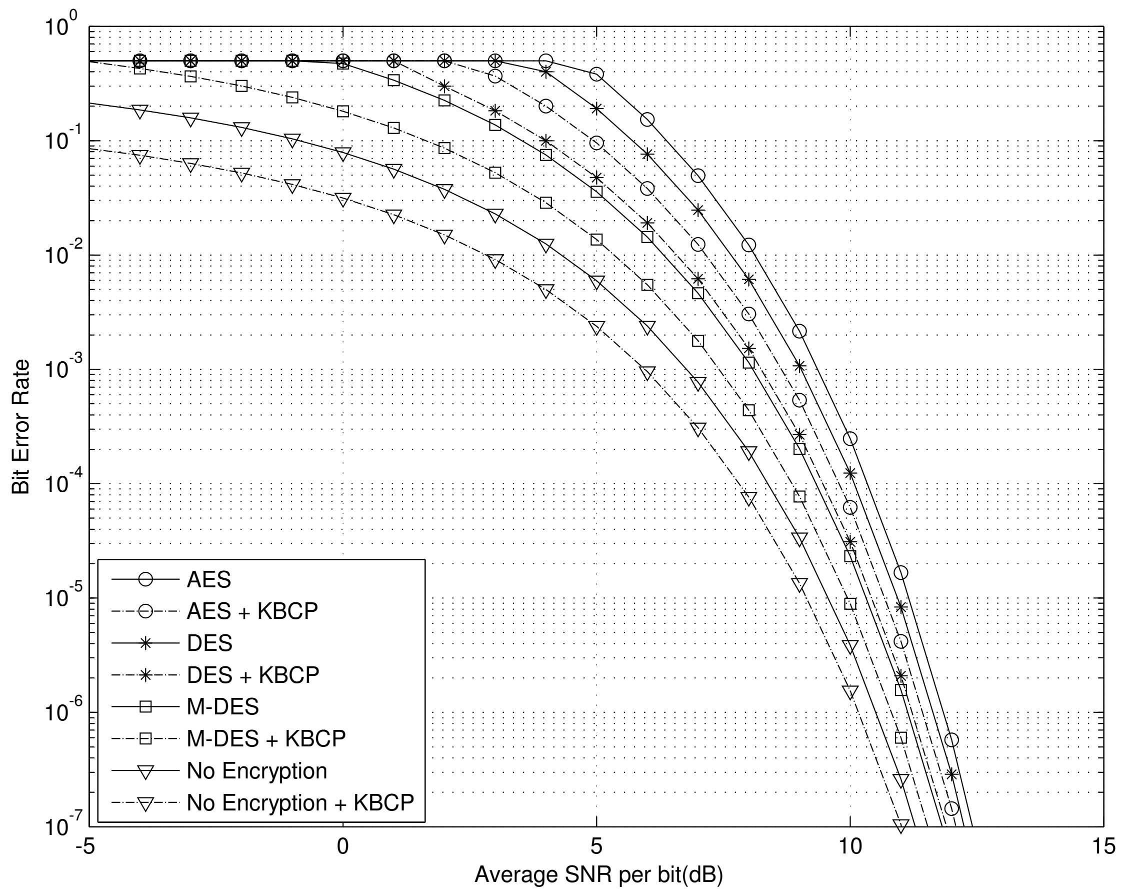


Figure 5.4. BER vs. SNR in AWGN channel with BPSK modulation, comparison of using AES, DES, M-DES (no channel coding), AES with hamming (7,4) channel coding, DES with hamming (7,4) channel coding and the proposed encryption algorithm KBCP-M-DES

that the key is used to place the a blocks, while the b blocks were placed in order. Given that the attacker by any way was able to access the 80-bit key, he will be able to use the key to figure out the locations of the a blocks, and the b blocks will be placed in their order. Hence, the main advantage of the first design of KBCP (i.e. one key only) is to improve the error performance with the least addition to the complexity. So, the availability of the 80-bit key is enough to figure out the location of the a and b blocks, and hence to reconstruct the code words and by decoding the codewords the actual data can be retrieved. The probability of a successful attack on KBCP with one key only is similar to that of M-DES.

As mentioned earlier in this chapter, the output of the first 16 rounds of M-DES is entered into the KBCP phase, where the 64-bit cipher of M-DES is converted to a 128-bit cipher.

In order to apply the differential cryptanalysis attack, the actual 64-bit cipher should be extracted by the attacker from the 128-bit cipher, for a large number of cipher, which helps the attacker to obtain the key. In KBCP with one key, the probability of extracting one 64-bit cipher out of its 128-bit cipher is equivalent to that of M-DES and is given by

$$p_1 = \frac{1}{\binom{32}{16} \times 16!} = 7.9515 \times 10^{-23}. \tag{5.7.1}$$

Similar to M-DES, KBCP requires 2^{14} pairs of plaintext and their associated 64-bit ciphertext to have a successful attack on the algorithm [7], the probability of attacking KBCP with one key using differential cryptanalysis is given by

$$\begin{aligned} P_{attack} &= \binom{N_P}{N_P} p_1^{N_P} (1-p_1)^{N_P - N_P} \\ &= p_1^{N_P} \\ &= (7.9515 \times 10^{-23})^{N_P} \\ &= (7.9515 \times 10^{-23})^{2^{14}} \\ &= 9.0841 \times 10^{-362080} \\ &= 0. \end{aligned} \tag{5.7.2}$$

On the other hand, the addition of the second key has added another security factor to the algorithm. Assuming that the attacker has access to the 128-bit ciphertext and to the 80-bit first key, it is not directly easy to find the 64-bit cipher of DES, due to the fact that the other halves of the codewords are not placed in order and there is 16! ways to arrange them through the cipher. In the case of KBCP with two keys, assuming that the attacker has access to 2^{14} pairs of plaintexts and their corresponding ciphertexts (64-bit to 128-bit), the probability of obtaining one useful pair will decrease by 16! compared to the case of KBCP with one key and M-DES. The probability of extracting one useful pair is given by

$$p_1 = \frac{1}{\binom{32}{16} \times 16! \times 16!} = 3.8 \times 10^{-36}. \tag{5.7.3}$$

Therefore, the probability of guessing all the useful ciphers is given by

$$\begin{aligned} P_{attack} &= (3.8 \times 10^{-36})^{2^{14}} \\ &= 1.4916 \times 10^{-580325} \\ &= 0. \end{aligned} \tag{5.7.4}$$

Hence, the probability of attack in KBCP two key mode significantly decreases compared to that of the one key mode and M-DES.

Another way to look at the possibility of attacking the algorithm using the differential cryptanalysis is to calculate the required time to do so. The number of trials an attacker needs to perform in order to achieve a successful attack on the one key mode of KBCP is given by

$$\begin{aligned} N_{trials} &= \left(\binom{32}{16} \times 16!\right)^{2^{14}} \\ &= 7.9 \times 10^{362078} trials. \end{aligned} \tag{5.7.5}$$

However, the fastest super computer today can perform 2.5×10^{12} operations per second. Therefore, the total time needed to attack KBCP one key mode using differential cryptanalysis is given by

$$\begin{aligned} T_{attack} &= \frac{7.9 \times 10^{362078}}{2.5 \times 10^{12}} \\ &= 3.16 \times 10^{362066} seconds \\ &= 1.002 \times 10^{362059} years \end{aligned} \tag{5.7.6}$$

to perform a successful attack on the one key mode of KBCP. On the other hand, for the two key mode KBCP, the number of trails that the attacker needs to perform increases significantly and is given by

$$\begin{aligned} N_{trials} &= \left(\binom{32}{16} \times 16! \times 16!\right)^{2^{14}} \\ &= 1.18 \times 10^{580324} trials. \end{aligned} \tag{5.7.7}$$

Therefore, the total time needed to attack KBCP two key mode using differential cryptanalysis is given by

$$\begin{aligned} T_{attack} &= \frac{1.18 \times 10^{580324}}{2.5 \times 10^{12}} \\ &= 4.7 \times 10^{580311} seconds \\ &= 1.5 \times 10^{580304} years \end{aligned} \tag{5.7.8}$$

to perform a successful attack on the two key mode of KBCP. Hence, the two key mode of KBCP requires 10^{218248} years more than the one key KBCP to attack the algorithm.

5.8 Conclusions

In this chapter we proposed a new encryption mechanism called KBCP-M-DES in which we introduce a key-based permutation and channel coding on top of the first 16 rounds of DES, as well as by keeping the modifications to the S-Boxes design in the first 16 rounds introduced earlier in M-DES. In one design we only added channel coding to the cipher which improved the error performance, and in another design we added another key in addition to the channel coding to further enhance the security of the encrypted information. We used computer simulations to show that the proposed algorithm improves the error performance at the receiver compared to the use of AES and DES with the same channel coding as well as to M-DES in a wireless application. We have also shown using numerical results that it is impossible to attack the proposed algorithm using the differential cryptanalysis unlike DES which was cracked by the differential cryptanalysis attack. As a result, we introduced a new encryption mechanism, that outperforms AES and DES in terms of the error performance in wireless channels as well as being secure against attacks.

Chapter 6

Complexity Analysis of Symmetric Encryption Algorithms

6.1 Introduction

Most of the encryption algorithms mentioned in earlier chapters are being used for years, therefore, a comprehensive complexity analysis is required to compare the performance of the algorithms independently from the platform implementations. The complexity analysis of this class of encryption algorithms (Symmetric Encryption Algorithms) had attracted many researchers, who developed and used different methodologies to analyze and compute the complexity [9]-[11]. However, those methodologies lacked the comprehensiveness in their analysis, due to the ignorance of the memory access time, which greatly affects the performance as we will show later in this chapter when we take it into consideration in our analysis. Moreover, we use computer simulations on two different platforms, to compute the effective encryption rate for symmetric encryption algorithms.

6.2 Motivation

A secure encryption algorithm could be very complicated, time consuming and hard to implement. Therefore, the complexity of an algorithm which we measure by the time required for encryption and/or decryption should be taken into account when the algorithm is designed and implemented. In this chapter we compute the complexity for a class of symmetric encryption algorithms in terms of the number of clock cycles required for encryption and decryption and hence in terms of the required time for encryption and decryption.

6.3 Contributions

In this chapter we provide a complete comprehensive complexity analysis for a class of symmetric-encryption algorithm. This is the first complexity analysis of symmetric-encryption algorithm that considers the required time to retrieve information such as the S-Box entries from memory. Therefore, the analysis is more comprehensive and more accurate than previous work. In addition, we use computer simulations on two different machines with two different microprocessors to encrypt various files with various sizes. Finally, we compare the analytical and simulation results to truly evaluate the accuracy of our analysis.

6.4 Analytical Complexity Analysis

In this section we present the methodology that we use to compare the complexity of various symmetric encryption algorithms. The analysis is based on counting the number of the byte-wise arithmetic operations required to perform encryption and decryption using each algorithm. This analysis is based on calculating the number of operations required to encrypt and decrypt one block of data. The operations of DES, 3-DEA, AES M-DES and KBCP-M-DES are limited to the logical XOR and shift operations. In addition to the memory access time that is required to access and retrieve some information necessary to perform encryption and decryption.

The number of byte-wise arithmetic operations and the number of byte-wise memory access can be simply transformed to the number of clock cycles required to perform each operation. The number of clock cycles is general for any platform, because the clock cycle is the smallest time entity on any platform. Therefore, the total time of encryption or decryption can be simply measured by multiplying the number of clock cycles by the time of each clock cycle which varies based on the used platform. In the next subsections we will apply this methodology on DES, 3-DEA, AES, M-DES and KBCP-M-DES encryption algorithms.

6.4.1 Analytical Complexity Analysis of DES

The complexity of DES can be expressed in terms of the number of required operation to perform encryption [10], and hence it can be translated to the number of clock cycles required for encryption. Both expressions make the analysis more generalized and valid for any platform with any processor(s) speed.

Number of Operations

The data encryption standard consist of an initial and final permutation stages which can be broke down to simple byte-wise shift operations. While the 16 rounds can be broke down to byte-wise XOR and shift operations.

The initial permutation and final permutations are based on changing bits locations only. Since the input to both phases is a 64-bit input, each of the initial and final permutation consists of exactly 8 *byte-wise* shift operations.

Now let us count the number of byte-wise operations for each of the rounds, the 64-bit input of each round is first divided into two 32-bit blocks. The operations inside the round are as follows.

- The left 32-bit block is XOR-ed with the 32-bit output of the F-Box. The output of the XOR operation is considered as the 32 right bits output of the round. Therefore, $32/8 = 4$ *byte-wise* XOR operations are required at this step.

- The right 32-bit block enters along with the round key as inputs to the F-Box. The 32-bit input is mapped to 48 bits using the P-Box. The P-Box is a direct permutation operation. So, $48/8 = 6$ *byte-wise* shift operations are required at this stage. The

48-bit of the P-Box is XORed with 48-bit round key. 6 *byte-wise* XOR operations are needed at this stage. The last and most complicated phase of the F-Box is the S-Box phase, where each 6-bits are mapped to a different 4-bit output using a pre-stored S-Box. Each 6 bits of the 48-bit input uses a different 4×16 bytes table. Eight different bytes need to be accessed and retrieved from the memory, one for each S-Box . So, 8 *byte-wise* memory accesses are required for this stage.

- The 1-1 permutation needs 4 *byte-wise* shift operations.
- Hence 10 *byte-wise* shift operations, 6 *byte-wise* XOR operations and 8 *byte-wise* memory access are required for the F-Box stage.

The total number of operations required for each round is

- $6 + 4 = 10$ *byte-wise* shift operations.
- $6 + 4 = 10$ *byte-wise* XOR operations.
- 8 *byte-wise* memory accesses.

Number of clock cycles

Assuming that Intel x86 instruction set is used. The number of clock cycles required to perform a shift or an XOR operation is 1 clock cycle only. While the number of clock cycles needed for the memory access is given in table 6.1 [43].

The total number of clock cycles needed to encrypt or decrypt one block of data is given by the following equation

$$\begin{aligned} N_{cc} &= (8+8) \times T_{shift} \\ &+ R(10 \times T_{shift} + 10 \times T_{xor} + 8 \times T_{ma}) \end{aligned} \tag{6.4.1}$$

where T_{xor}, T_{shift}, T_{ma} are the number of cycles needed to perform one XOR operation, one shift operation and one memory access operation respectively. And R is the number of

Table 6.1. Number of clock cycles required to access various memory types

Memory Type	Clock Cycles
Registers	1
L1 Cache	4
L2 Cache	10
L3 Cache	40-75
Main Memory	60-100

rounds which is 16 in DES.

Based on the Intel x86 instruction set, $T_{xor} = 3$ clock cycles, $T_{shift} = 1$ clock cycle and $T_{ma} = 80$ because the tables are originally stored in the main memory. Therefore, the total number of clock cycles cycles needed to encrypt the first block of data using DES is given by,

$$\begin{aligned} N_{cc} &= (8+8) \times 1 + 16(10 \times 1 + 10 \times 3 \\ &+ 8 \times 80) = 10896 \ \textit{clock cycles}. \end{aligned} \tag{6.4.2}$$

To find the total number of clock cycles required to encrypt any number of blocks, the number of blocks N is multiplied by the average number of clock cycles required to encrypt any block. The number of clock cycles required to encrypt the $n-th$ block is supposed to be less than or equal to the $(n-1)th$ block. This is due to the fact that the time required to access the memory to retrieve the S-Box entries will vary from block to another. Moreover, the total memory access time of the current block is less than that required for the previous block, because some entries may already been retrieved earlier and hence stored at L1 or L2 cache, which saves a significant amount of time, especially when the number of encrypted blocks increase,

$$N_{cc1} \geq N_{cc1} \geq \ldots \geq N_{cc(n-1)} \geq N_{ccn}. \tag{6.4.3}$$

To be able to estimate the number of required clock cycles analytically, we find the upper and lower bounds for this number, by assuming that all memory accesses are main memory access for all blocks (upper bound), and by assuming that all memory accesses are L2 cache memory accesses (lower bound). Moreover, the time required for encryption is given by the total number of clock cycles multiplied by the time of each clock cycles (clock duration), which varies based on the implemented platform. The upper and lower bounds for the total number of clock cycles are given by

$$N_{upper} = N \times 10896 \tag{6.4.4}$$

$$N_{lower} = (N-1) \times 1168 + 10896. \tag{6.4.5}$$

Therefore, the average number of clock cycles needed to encrypt a stream of bits using DES is given by

$$N_{avg} = \frac{N_{upper} + N_{lower}}{2}. \tag{6.4.6}$$

Hence, the required time to encrypt a stream of bits can be calculated analytically by

$$T_{avg} = N_{avg} \times \frac{1}{f} \tag{6.4.7}$$

where f is the frequency of the processor used in encryption. For example if we take the case of encrypting 50 KB of information, the variable N in equation 6.4.4 is given by

$$N = \frac{50 \times 1024 \times 8}{64} = 6400. \tag{6.4.8}$$

Therefore, the maximum and minimum number of clock cycles required to encrypt the 6400 frames are given by

$$\begin{aligned} N_{upper} &= 6400 \times 10896 = 69734400 \\ N_{lower} &= 6399 \times 1168 + 10896 = 7484928 \\ N_{avg} &= \frac{N_{upper} + N_{lower}}{2} = 38609664. \end{aligned} \tag{6.4.9}$$

Hence, if a machine with $f = 2.66GHZ$ is used to encrypt the 50 KB file, the average time (in milliseconds) required for encryption is given by

$$T_{avg} = 38609664 \times \frac{1}{2.66 \times 10^6} = 14.515. \tag{6.4.10}$$

On the other hand, if a machine with Dual processors each having a frequency $f = 2.6GHZ$ is used to encrypt the 50 KB file, the maximum and minimum amount of times (in milliseconds) required for encryption is given by

$$T_{avg} = 38609664 \times \frac{1}{2 \times 2.6 \times 10^6} = 7.42. \quad (6.4.11)$$

So, the average time needed on machine 1 to encrypt a 50 KB is equal to 14.515 milliseconds, while the average time required to encrypt the same file size on machine 2 is equal to 7.42 milliseconds. Table 6.2 shows the average times required to encrypt various file sizes calculated analytically in the same procedure presented above.

Table 6.2. Encryption time required to encrypt different file sizes on two different machines, calculated analytically when DES is used for encryption

Input File Size (Kbytes)	Machine 1 (Analytical)	Machine 2 (Analytical)
50	14.5	7.4
75	21.8	11.1
100	29	14.8
200	58.1	29.7
350	101.6	52
500	145.1	74.2
800	232.2	118.8
5,000	1451.3	742.4
10,000	2902.6	1484.8
15,000	4353.9	2227.2
20,000	5805.2	2969.6
Encryption Rate (Mbytes/sec)	3.46	6.71

6.4.2 Analytical Complexity Analysis of 3-DES

The complexity of 3-DES in terms of encryption would be simply triple that of DES, because the algorithm is based on repeating DES three times with different keys. However, the number of clock cycles maybe slightly less than that of DES due to the fact that some entries in the S-Boxes maybe already accessed and retrieved, hence it will be saved in the L1 or L2 cache which requires less time for accessing compared to that of the main memory.

To find the total number of clock cycles required on 3-DES, we first assume that all the S-Box entries required in the second and third stages of 3-DES were already retrieved in the first stage, then each memory access requires between 4 and 10 memory cycles rather than 80 clock cycles (depending on whether the retrieved entries are stored in L1 or L2 cache). Hence T_{ma} in equation 6.4.1 is substituted by 7 (average of 4 and 10) rather than 80 in DES. Moreover, in this case the Number of clock cycles is given by

$$\begin{aligned} N_{cc} &= 10896 + 2 \times [(8+8) \times 1 + 16(10 \times 1 + 10 \times 3 \\ &+ 8 \times 7)] = 14000 \ \textit{clock cycles}. \end{aligned} \tag{6.4.12}$$

On the other hand, if we assume that non of the S-Box entries in the second and third stages were already retrieved in the first stage, the number of clock cycles is given by

$$N_{cc} = 3 \times 10896 = 32688 \ \textit{clock cycles}. \tag{6.4.13}$$

Based on this, the number of clock cycles required to encrypt the first block using 3-DES would be greater than or equal to 14000 clock cycles and less than or equal to 32688 clock cycles. To generalize the analysis for any number of blocks, we find upper and lower bounds for the total number of clock cycles, which is given by

$$\begin{aligned} N_{upper} &= N \times 32688 \\ N_{lower} &= (N-1) \times 14000 + 32688. \end{aligned} \tag{6.4.14}$$

Using the same procedure presented at the end of the previous sub-section, table 6.3 shows the average times analytically required to encrypt various file sizes using 3-DES.

6.4.3 Analytical Complexity Analysis of AES

To implement the initial stage of the AES, 16 *byte-wise* XOR operations are required. Each of the AES rounds includes the S-Box phase, shift row phase, mix columns phase and the

Table 6.3. Encryption time needed to encrypt different file sizes on two different machines, calculated analytically when 3DES is used for encryption

Input File Size (Kbytes)	Machine 1 (Analytical)	Machine 2 (Analytical)
50	56	29
75	84	43
100	112	57
200	225	115
350	393	201
500	562	287
800	899	460
5,000	5617	2873
10,000	11233	5746
15,000	16850	8619
20,000	22466	11492
Encryption Rate (Mbytes/sec)	0.89	1.73

add key phase. Based on the same analysis done earlier in DES; to implement one round of AES, 184 *byte-wise* AND operations, 136 *byte-wise* OR operations and 352 *byte-wise* shift operations are required. Similarly, the final rounds requires 16 *byte-wise* XOR operations, 12 *byte-wise* OR operations and 12 *byte-wise* shift operations. In addition each round will require 16 *byte-wise* memory access operations for the S-Box stage. Therefore the total number of clock cycles required to encrypt one block of data using AES is given by,

$$\begin{aligned} N_{cc} &= 16 \times T_{xor} \\ &+ (R-1)(181 \times T_{AND} + 136 \times T_{or} + 352 \times T_{shift} \\ &+ 16 \times T_{ma}) \\ &+ (16 \times T_{xor} + 12 \times T_{shift} + 12 \times T_{or}) \end{aligned} \tag{6.4.15}$$

where T_{xor}, T_{shift}, T_{ma}, T_{AND} and T_{OR} are the number of cycles required to perform one XOR operation, one shift operation, one memory access operation, one AND operation and one OR operation, respectively; R is the number of rounds. R might be 10, 12 or 14 depending on the key size selected for encryption.

As mentioned earlier, based on the Intel x86 instruction set, $T_{xor} = 3$ clock cycles, $T_{shift} = 1$ clock cycle and $T_{ma} = 80$ and $T_{AND} = 1$ clock cycle and $T_{OR} = 1$ clock cycle. Hence the number of clock cycles needed to encrypt a block of data using AES is given by

$$\begin{aligned} N_{cc} &= 16 \times 3 \\ &+ (R-1)(181 \times 1 + 136 \times 1 + 352 \times 1 \\ &+ 16 \times 80) + (16 \times 3 + 12 \times 1 + 12 \times 1). \end{aligned} \tag{6.4.16}$$

Therefore, $N_{cc} = 17688$ clock cycles when $R = 10$, $N_{cc} = 21592$ clock cycles when $R = 12$ and $N_{cc} = 25496$ clock cycles when $R = 14$. Hence, AES requires 60%, 98% and 114% more clock cycles than DES, when the key length is 128, 192 and 256 bits respectively. As done earlier, to generalize the total required number of clock cycles, we find upper and lower bounds for all the 128-bit, 192-bit and 256-bit AES cases, which are given respectively by

$$\begin{aligned} N_{upper} &= N \times 17688 \\ N_{upper} &= N \times 21592 \\ N_{upper} &= N \times 25496 \end{aligned} \tag{6.4.17}$$

$$\begin{aligned} N_{lower} &= (N-1) \times 6717 + 17688 \\ N_{lower} &= (N-1) \times 8183 + 21592 \\ N_{lower} &= (N-1) \times 9649 + 25496. \end{aligned} \tag{6.4.18}$$

Using the same procedure presented at the end of the previous sub-sections, table 6.4 shows the average times analytically required to encrypt various file sizes using AES.

6.4.4 Analytical Complexity Analysis of M-DES

M-DES was introduced as an efficient, reliable and secure encryption algorithm that can be used in wireless communication channels. M-DES is based on DES, however the key and the ciphertext sizes are increased [7]. In M-DES, the first 16 round structure is exactly the same

Table 6.4. Encryption time needed to encrypt different file sizes on two different machines, calculated analytically when AES is used for encryption

Input File Size (Kbytes)	Machine 1 (Analytical)	Machine 2 (Analytical)
50	29	15
75	44	22.5
100	59	30
200	117	60.1
350	206	105.1
500	294	150.2
800	470	240.3
5,000	2936	1501.8
10,000	5872	3003.7
15,000	8808	4505.5
20,000	11744	6007.4
Encryption Rate (Mbytes/sec)	1.7	3.31

as DES, except that there are two distinct mapping tables rather than eight. However, this will not reduce the number of required byte-wise memory access, because 8 bytes still need to be extracted from the memory. Although in DES the 8 bytes need to be retrieved from 8 distinct mapping tables and in M-DES the 8 bytes need to be extracted from 2 distinct mapping tables, this will not decrease the number of memory access when encrypting the first data block, because initially all tables are stored in the main memory.

Therefore, the first 16 rounds of M-DES requires the same number of clock cycles that are required for DES. The addition of round 17, requires 10 more *byte-wise* memory access and 8 *byte-wise* shift operations. Hence, $10 \times 8 + 8 \times 1 = 880$ more clock cycles are needed for round 17. As a result M-DES requires $10896 + 880 = 11776$ clock cycles to encrypt one block of data. So M-DES requires 9% more clock cycles than DES. The upper and lower bounds for the total number of clock cycles for N blocks is calculated similarly to the previous cases and are given by

$$\begin{aligned} N_{upper} &= N \times 11776 \\ N_{lower} &= (N-1) \times 2048 + 11776. \end{aligned} \tag{6.4.19}$$

Using the same procedure presented at the end of the previous sub-sections, table 6.5 shows the average times analytically required to encrypt various file sizes using M-DES.

Table 6.5. Encryption time needed to encrypt different file sizes on two different machines, calculated analytically when M-DES is used for encryption

Input File Size (Kbytes)	Machine 1 (Analytical)	Machine 2 (Analytical)
50	16.6	8.5
75	24.9	12.8
100	33.3	17
200	66.5	34
350	116.4	59.6
500	166.3	85.1
800	266.1	136.1
5,000	1663	850.7
10,000	3326.1	1701.4
15,000	4989.1	2552.1
20,000	6652.2	3402.8
Encryption Rate (Mbytes/sec)	3.0	5.85

6.4.5 Analytical Complexity Analysis of KBCP-M-DES

In this subsection we evaluate the complexity increase of KBCP-M-DES compared to that of DES. First lets analyze the complexity of the coding procedure, where 8 bytes of data are being coded to 16 bytes. The coding adds 8 byte-wise multiplication operations and 16 byte-wise modulo 2 operations (which are basically division operations). Assuming that the operations are done on the same processor architecture introduced earlier, it will take 12 clock cycles on average to perform one byte-wise multiplication and 14 clock cycles on average to perform one byte-wise division. Therefore, the coding adds an overhead to the total number of clock cycles given by

$$N_{co} = (12 \times 8) + (14 \times 16) = 320. \quad (6.4.20)$$

Based on the same methodology used earlier in the chapter, the maximum and minimum number of clock cycles required to encrypt N blocks of information using KBCP-M-DES in its one key version is given by

$$\begin{aligned} N_{upper} &= N \times 12088 \\ N_{lower} &= (N-1) \times 1596 + 12088. \end{aligned} \quad (6.4.21)$$

Table 6.6 summarizes the average times analytically required to encrypt various file sizes using KBCP-M-DES in its one key version.

Table 6.6. Encryption time needed to encrypt different file sizes on two different machines, calculated analytically when KBCP-M-DES in its one key version is used for encryption

Input File Size (Kbytes)	Machine 1 (Analytical)	Machine 2 (Analytical)
50	17.3	8.9
75	26.1	13.5
100	34.8	17.8
200	69	35.5
350	121	62
500	173.7	90
800	278	142
5,000	1739	890
10,000	3479	1780
15,000	5220	2670
20,000	6960	3550
Encryption Rate (Mbytes/sec)	2.87	5.6

On the other hand the permutation process using the one key mode of the proposed algorithm requires 10 more *byte-wise* memory access and 8 *byte-wise* shift operations. Hence,

$10 \times 8 + 8 \times 1 = 880$ more clock cycles are required. As a result, the proposed algorithm in its one key version requires $10896 + 880 + 320 = 12088$ clock cycles to encrypt one block of data. So the one key version of the KBCP-M-DES requires 11% more clock cycles than DES. On the other hand, by introducing the second key to map the b blocks, the number of clock cycles is further increased 880 clock cycles as we had shown earlier due to the permutation overhead associated with the second key. Therefore, the KBCP in its two keys version requires $10896 + 880 + 880 + 320 = 12976$ clock cycles to encrypt one block of data, which is a 19% increase compared to DES. Based on the same methodology used earlier in the chapter, the maximum and minimum number of clock cycles required to encrypt N blocks of information using KBCP-M-DES in its two key version is given by

$$\begin{aligned} N_{upper} &= N \times 12968 \\ N_{lower} &= (N-1) \times 1704 + 12968. \end{aligned} \qquad (6.4.22)$$

Table 6.7 summarizes the average times analytically required to encrypt various file sizes using KBCP-M-DES in its two key version. To summarize this section, tables 6.8 and 6.9 summarizes the average amount of time required analytically to encrypt different file sizes on Intel Pentium(R) 4 CPU 2.66 GHZ and on Intel(R) 2.60 GHZ, using the encryption algorithms under study. It can be noticed from both tables that DES has the highest encryption rate which means that it uses the least amount of time to encrypt the same amount of data compared to the other encryption algorithms under study, followed by M-DES, KBCP-M-DES in its one key version and KBCP-M-DES in its two key version then AES and finally 3-DES which has the lowest encryption rate and hence the highest amount of time for encryption.

6.5 Simulations Results

The simulation is based on encrypting various files of various sizes using each algorithm, then the time that each algorithm takes for encryption will be measured. The simulation is

Table 6.7. Encryption time needed to encrypt different file sizes on two different machines, calculated analytically when KBCP-M-DES in its two key version is used for encryption

Input File Size (Kbytes)	Machine 1 (Analytical)	Machine 2 (Analytical)
50	17.7	9.0
75	26.5	13.7
100	35.3	18.1
200	70.6	36.1
350	123.6	63.2
500	176.5	90.3
800	282.4	144.5
5,000	1765	903
10,000	3530	1806
15,000	5295	2708
20,000	7060	3612
Encryption Rate (Mbytes/sec)	2.83	5.53

performed on two machines that have different processors with different capabilities. The first machine processer is an Intel Pinetum 4 with a clock cycle of 2.66 GHZ and the other one has a processor of Intel Core i5-2540M with a clock cycle of 2.6 GHZ.

Figure 6.1 and 6.2 shows the time needed for encryption versus the file size, using the four encryption algorithms on both stations. The figures also compares between the analytical and simulation results. It can be noticed from both figures that the simulation time is slightly less than the analytical time due to the fact that when the file size increases the probability of having more data stored in the registers and level 1 cache increases, which reduces the time required to encrypt later frames.

Table 6.8. The time (in milliseconds) required to encrypt different file sizes (in Kbytes) on Intel Pinetum(R) 4 CPU 2.66 GHZ

Input File Size (Kbytes)	DES	M-DES	KBCP-M-DES 1	KBCP-M-DES 2	AES	3-DES
50	14.5	16.6	17.3	17.7	29	56
75	21.8	24.9	26.1	26.5	44	84
100	29	33.3	34.8	35.5	59	112
200	58.1	66.5	69	70.6	117	225
350	101.6	116.4	121	123.6	206	393
500	145.1	166.3	173.7	176.5	294	562
800	232.2	266.1	278	282.4	470	899
5000	1451.3	1663	1739	1765	2936	5617
10000	2902.6	3326.1	3479	3500	5872	11233
15000	4353.9	4989.1	5220	5295	8808	16850
20000	5805.2	6652.2	6960	7060	11744	22466
Encryption Rate (Mbytes/sec)	3.46	3.0	2.87	2.83	1.7	0.89

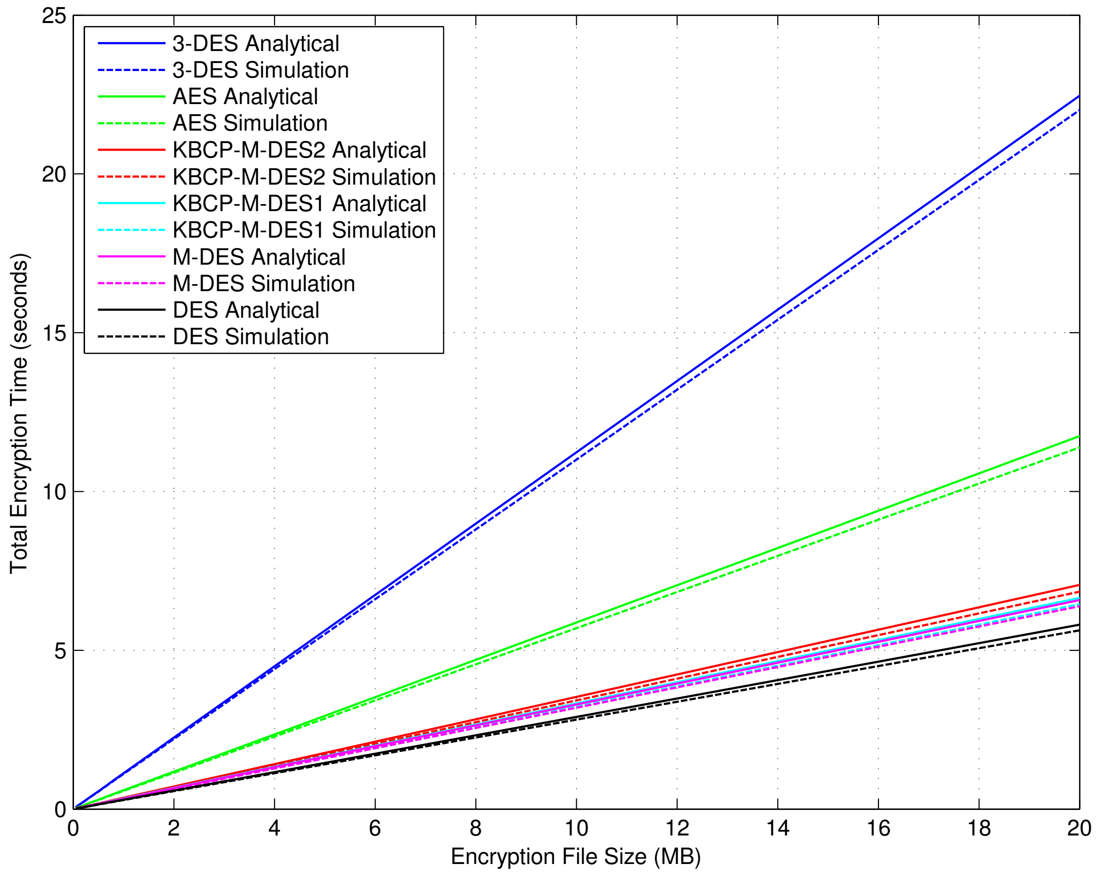


Figure 6.1. The time required to encrypt different file sizes using AES, DES and M-DES on machine 1

Table 6.9. The time (in milliseconds) needed to encrypt different file sizes (in Kbytes) on Intel(R) 2.60 GHZ

Input File Size (Kbytes)	DES	M-DES	KBCP-M-DES 1	KBCP-M-DES 2	AES	3-DES
50	7.4	8.5	8.9	9.0	15	29
75	11.1	12.8	13.5	13.7	22.5	43
100	14.8	17	17.8	18.1	30	57
200	29.7	34	35.5	36.1	60.1	115
350	52	59.6	62	63.2	105.1	201
500	74.2	85.1	90	90.3	150.2	287
800	118.8	136.1	142	144.5	240.3	460
5000	742.4	850.7	890	903	1501.8	2873
10000	1484.8	1701.4	1780	1806	3003.7	5746
15000	2227.2	2552.1	2670	2708	4505.5	8619
20000	2969.6	3402.8	3550	3612	6007.4	11492
Encryption Rate (Mbytes/sec)	6.71	5.85	5.6	5.53	3.31	1.73

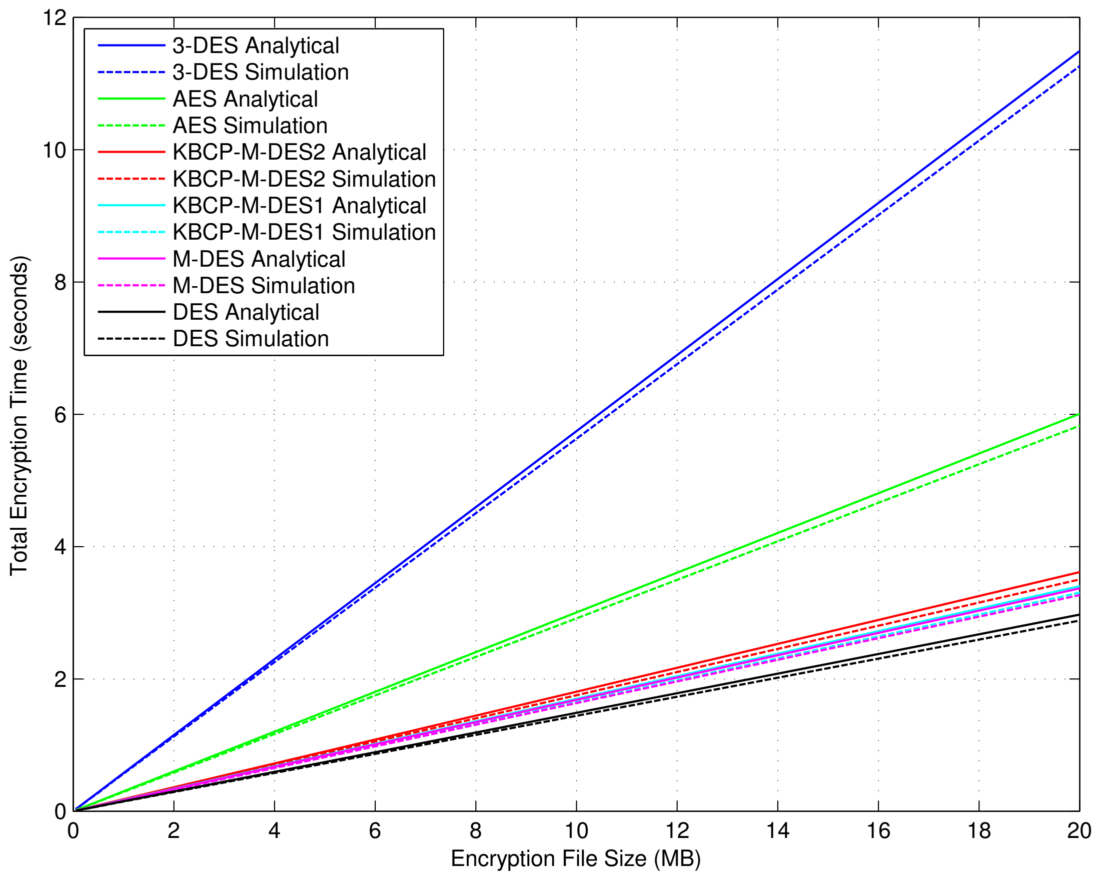


Figure 6.2. The time required to encrypt different file sizes using AES, DES and M-DES on machine 2

It can also be noticed from both figures that KBCP-M-DES in its one and two key version adds a slight addition to the complexity of DES, but are still less than that of AES and 3-DES.

6.6 Conclusions

In this chapter we introduced a new methodology to analytically analyzing the complexity for symmetric block ciphers, which takes into account the time needed to access and retrieve the memory during the encryption and decryption process. We then applied this methodology on a class of symmetric block ciphers that includes DES, M-DES, KBCP-M-DES, 3-DES and AES. We had shown that AES and 3-DES are the most complicated encryption algorithm within the selected class. Finally, we used computer simulations to compare the accuracy of the analytical results and their effective encryption rate of different ciphers on different implemented platforms.

Chapter 7

Energy Consumption Analysis of Symmetric Encryption Algorithm

7.1 Introduction

Due to the increased demand on wireless devices and their applications, the necessity for an efficient, fast and secure encryption algorithms is critical. However, energy resources are limited in wireless devices, therefore, having a secure an energy efficient encryption algorithm for wireless applications is a major challenge. A secure encryption algorithm is considered energy efficient if in addition to being secure, it is easy to implement, use minimum number of CPU operations and hence uses less energy for encrypting and decrypting the transmitted and received data. In this chapter we use numerical calculations to analyze the energy consumption for a class of encryption algorithms including the Advanced encryption standard (AES), the data encryption standard (DES), the triple DES (3-DES) and a modification to the data encryption standard called KBCP-M-DES. We compute the number of arithmetic and logical instructions in addition to computing the number of memory access used by each of the algorithms under study. Given some information about the microprocessor used in encryption, we can compute the energy consumed per each instruction (EPI) and hence compute the total energy consumed by the encryption algorithm. In addition, we use computer simulations to compare the energy savings of transmitting encrypted information over the wireless channel using the encryption algorithms under study.

7.2 Motivation

Given that energy resources are limited in wireless devices and a battery has a limited life cycle, these resources have to be used as efficient as possible in order to increase the device life cycle before recharging or changing the battery. The analysis in this chapter, will help us understand how different parts of the encryption algorithm affects he energy consumption which will in fact give us better understanding of the tradeoffs between energy consumption and other factors such as complexity, error performance and security.

7.3 Contributions

Therefore, in this chapter we introduce a comprehensive approach to analyze the energy consumption for the encryption algorithms under study, where we use two approaches to compare the energy consumption of encryption algorithms. The first approach is focused on computing the energy consumption of the microprocessor used for encryption, while the second approach is more concerned in the energy required to transmit the encrypted information over the wireless channel.

7.4 Energy Per Instruction (EPI)

With the historical advances in wireless communication and encryption technique, the security of encryption algorithms have improved significantly. However, the energy consumption of such algorithms should be now taken into account. Smart phones and tablets are the most used devices in the world today, and the battery life of such devices has played a great factor affecting customers choices. On the other hand, encryption is vital to be implemented on those devices due to the fact the wireless medium is open to intruders and their attacks. Therefore, the energy consumption of encryption algorithms for wireless applications should be considered and minimized in order to maximize the device battery life.

In [48] the authors introduced a new mechanism to analytically compute the energy consumption of the microprocessor when performing a certain task, in terms of the number of instructions required to perform the required task. Hence the energy consumption per each simple instruction is computed for any microprocessor and hence the total energy consumption of the microsporoses can be found by simply multiplying the energy per instruction (EPI) by the number of instructions to perform the task. In [49]-[51], the authors make use of the energy consumption evaluation technique introduced in [48] to measure the energy consumption of the advanced encryption standard and/or other encryption and channel coding techniques. Although the energy consumption analysis in [49]-[51] was based on computing the total number of instructions required for encryption and decryption, the energy consumption was not well evaluated due to the fact that the memory access instructions that are required for encryption and decryption were ignored in the analysis and hence the energy calculations are not accurate. In addition, all these work was concerned in the energy consumption of the microprocessor used for encryption and decryption, while the amount of consumed energy to transmit the encrypted information was not part of these works. However, in [12]-[18] the authors were more focused on the energy consumption of the transmitted information, where they introduced different encryption and/or channel coding techniques and computed the energy of the transmitted encrypted/codded signal over a certain range of signal-to-noise ratio (SNR). The savings in the SNR from one technique to another is used as a major representation of the transmitted energy savings comparison between two or more encryption/channel coding techniques.

In this chapter we use a combination of both approaches to compute the energy of the encryption and decryption process on a certain microprocessor, and to use the bit-error-rate results computed at the receiver to measure the energy savings from one algorithm to the other. We compute the energy consumption of the standardized well-known encryption algorithms, the data encryption standard (DES), the triple data encryption algorithm (3-DES) and the advanced encryption standard (AES), in addition to other modifications introduced on the data encryption standard such as the key based coded permutation modified-DES (KBCP-M-DES) introduced in earlier chapters.

In this section we analyze the energy consumption of the selected encryption algorithms based on their hardware complexities. In other words, we calculate the total number of required instructions (i.e, arithmetic, logic and memory access instructions) to encrypt one or more blocks of data. Based on the hardware used for encryption, we can calculate the energy consumed by each instruction using that particular hardware. Hence, we can calculate the total energy consumed by the encryption algorithm on any particular hardware. In later sections, we analyze the energy savings of each encryption algorithm in regards to the AES algorithm, based on the SNR savings for a certain bit error rate (BER) requirement. This is an excellent indicator of the energy saving in a wireless channel environment as shown in [12]-[18].

The energy per instruction (EPI) concept was first introduced by [48], where a generalized mechanism for computing the energy consumed by a microprocessor to perform a certain task was introduced. Each microprocessor requires x amount of joules to perform one instruction and depending on the total number of instructions required to perform the task, the total energy consumption for the performed task can be measured. EPI can be used as a measure for the microprocessor energy efficiency and is measured in Joules/Instruction. The EPI of a certain microprocessor is a function of the following factors

1. The Microprocessor design in terms of the logic and arithmetic circuits and the layout
2. Process Technology (different fabrication or transistor size, i.e., 0.8 um, 0.6 um, 180 nm, 65 nm)
3. Supply voltage

The energy per instruction depends on the microprocessor clock rate, the number of instructions per cycle or per clock (IPC) and the power used by the microprocessor. The energy per instruction can be simply calculated by the ratio of the power to the performance. Where the performance is measured in terms of the average number of instructions that can be processed within one clock cycle. Therefore, the EPI can be given by

$$EPI \;=\; \frac{P}{F \times IPC}, \tag{7.4.1}$$

where P is the power used by the microprocessor and is measured by watts/hour, F is the clock rate or the frequency of the microprocessor and is measured by HZ and IPC is the number of instructions that the microprocessor can process during one clock cycle and is measured by Instructions/Cycle. Given the Power, IPC and the frequency of some popular microprocessors, we calculate the EPI for these selected microprocessors in Table 7.1.

Table 7.1. Instruction Per Clock Cycle (IPC) and Energy Per Instruction (EPI) for a group of microprocessors

CPU Name	Power (Watt)	F	IPC	EPI
i486	4.9	66 MHZ	7.4	10
Pentium Pro	29.2	150 MHZ	8.1	24
Pentium 4	86	3.6 GHZ	0.5	48
Pentium M	21	2.0 GHZ	0.7	15
Core Due	31	2.167 GHZ	1.3	11

For example, if we take the Pentium Core Due microprocessor, where F is equal to 2.167 GHZ, IPC is equal to 1.3 and it requires a power of 31 Watt, the EPI is given by

$$EPI \;=\; \frac{31}{2.167 \times 10^{9} \times 1.3} = 11 \;\; nj,$$

similarly, the average energy per instruction is calculated for other microprocessors given in Table 7.1.

7.5 Energy Consumption Based on EPI

In this section, we use the Energy Per Instruction (EPI) concept to analytically compute the energy consumption of various encryption algorithms based on the microprocessor of the device running the encryption procedure. As we illustrated earlier through this chapter, the first step to analytically compute the energy consumption of a microprocessor running a certain task or procedure, is to find the total number of instructions required to run that task or procedure.

The complexity analysis for some of the algorithms under study had attracted many researchers, who developed and used different methodologies to analyze and compute the number of instructions required to encrypt and decrypt a certain amount of data [52]-[55]. However, those methodologies lacked the comprehensiveness in their analysis, due to the fact that the memory access time had been ignored, which greatly affects the performance as we will show in this section.

The complexity analysis in [52]-[55] is based on counting the number of the byte-wise arithmetic operations required to perform encryption and decryption using each algorithm, in addition to this, we consider the memory access time required to access and retrieve necessary information to perform encryption and decryption which was ignored in [52]-[55]. The number of byte-wise arithmetic operations and the number of byte-wise memory access can be simply transformed to the number of clock cycles required to perform each operation. The number of clock cycles is general for any platform, because the clock cycle is the smallest time entity on any platform.

Analyzing the data encryption standard, DES requires 160 *byte-wise* shift operations, 160 *byte-wise* XOR operations and 144 *byte-wise* memory accesses. Given that the Intel x86 instruction set is used, the number of clock cycles required to perform a shift or an XOR operation is 1 clock cycle only. While the number of clock cycles required to perform one memory access is averaged at 80 clock cycles [43]. Therefore, the total number of clock cycles required to encrypt one block of information using DES is equal to 10896 clock cycles. on the other hand, KBCP-M-DES proposed in earlier chapters requires 880 more clock cycles in its one key version compared to DES to encrypt one block of information due to the addition of the KBCP round, while the two key version requires 2080 additional clock cycles compared to DES. Similarly, 3-DES requires 160 more XOR operations, 480 more shift operations and 896 more memory access compared to DES, resulting in a total of 32688 clock cycles to encrypt one block of information using 3-DES. Based on the same analysis done for DES, KBCP-M-DES and 3-DES; to encrypt one block of data using AES, 184 *byte-wise* AND operations, 136 *byte-wise* OR operations and 352 *byte-wise* shift operations are required. In addition to 16 *byte-wise* XOR operations which are required for the initial stage of AES. Therefore, $N_{cc} = 17688$ clock cycles are required when $R = 10$ (i.e, 128-bit AES version), $N_{cc} = 21592$ clock cycles are required when $R = 12$ (i.e, 192-bit AES version) and $N_{cc} = 25496$ clock cycles are required when $R = 14$ (i.e, 256-bit AES version).

Given that the number of clock cycles required for encryption is now known, the encryption energy consumption on a certain microprocessor is given by

$$\begin{aligned} Energy &= N_{cc} \times IPC \times EPI \\ &= \frac{N_{cc} \times P}{F}, \end{aligned} \tag{7.5.1}$$

where N_{cc} is the number of clock cycles required for encryption. For example, if we use DES for encryption on an Intel Pentium Core Due microprocessor where the IPC is 1.3, EPI is equal to 11 nj and $N_{cc} = 10896$ as we shown for DES, the energy consumed by the microprocessor is equal to $10896 \times 31/2.167 \times 10^6 = 156\,\mu j$. Table 7.2 summarizes the energy consumption for each of the algorithms under study using each of the IPC and EPI for different microprocessors presented in Table 7.1. It can be noticed from the table that 3-DES requires the highest energy to get one block of data encrypted. We can also notice that KBCP-M-DES adds a little more energy to DES due to the fact that KBCP-M-DES adds a channel coding and permutation round to DES. Furthermore, it can also be noticed that the IPC and EPI of the microprocessor directly affects the total energy consumption of the encryption algorithm.

Next, we use numerical results based on the analysis we introduced earlier in this section

Table 7.2. Energy Consumptions (per byte) of encrypting one data block on a group of Microprocessors in micro joules using DES, M-DES, AES and 3-DES encryption algorithms

CPU Name	DES	KBCP-M-DES (one key)	AES-128	3-DES
i486	806	871	1309	2419
Pentium Pro	2118	2289	3439	6355
Pentium 4	261	282	425	785
Pentium M	114	124	186	343
Core Due	156	168	253	467

to find the total energy consumption to encrypt various amount of information using the different encryption algorithms under study. In figure 7.1 we use the i486 microprocessor to encrypt different amount of information using DES, KBCP-M-DES, AES and 3-DES. Similarly, figures 7.2, 7.3, 7.4 and 7.5 represent the energy consumption of using the Pentium Pro, Pentium 4, Pentium M and the Core Due microprocessors, respectively. It can be noticed from the figures that the microprocessor energy consumption (per byte) of 3-DES is the most and that consumed by DES is the least. While KBCP-M-DES is very close to both AES and DES.

7.6 Energy Consumption Based on The Error Performance

In the previous section we compared the energy consumption for a class of encryption algorithms based on the energy consumed by the microprocessor used for the encryption process. Therefore, the previous section analysis depends on the number of instructions and therefore on the number of clock cycles and the number of instructions per cycles. Moreover, when the microprocessor that is used for encryption is faster, the number of the required instruction will decrease and hence the energy consumption of the microprocessor decreases as well. This analysis is a great representation for the energy consumption of the microprocessor used for encrypting and decrypting the data, but is not sufficient to represent the energy involved with the transmission of the encrypted signal over the wireless channel. In this section, we

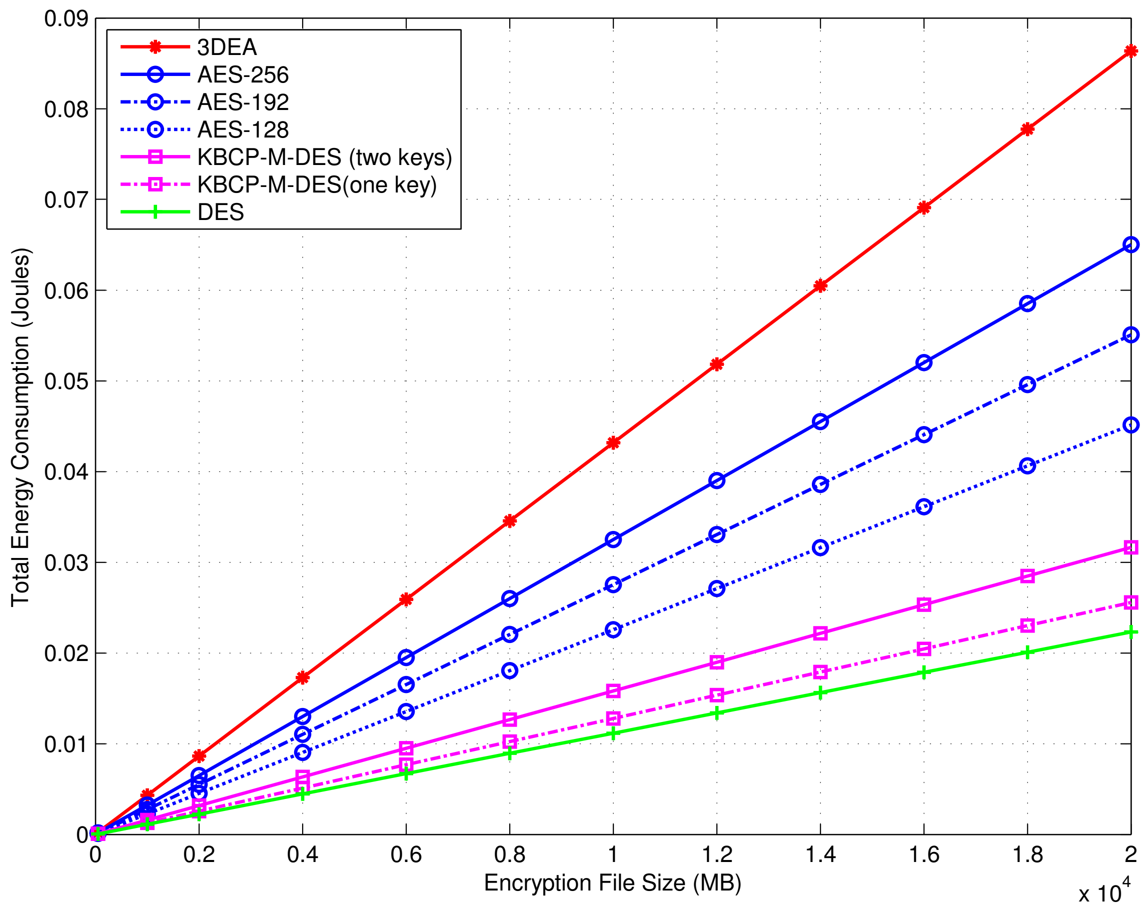


Figure 7.1. Energy consumption per byte for encrypting different file sizes using different encryption algorithms on the Intel i486 Microprocessor

will use a simple but a profound mechanism which is used in the literature to accurately measure the savings in energy between different channel coding and/or encryption mechanisms used to transmit data over the wireless channel [16]-[18]. In this chapter we use a comprehensive method of comparison based on both approaches.

In this section, we use computer simulations to encrypt a large amount of information using the encryption algorithms under study. The encrypted information is then transmitted over an additive white gaussian noise channel (AWGN) with a binary phase shift keying (BPSK) modulation scheme. The receiver then decrypts the received information and the error is computed by comparing the data before encryption at the transmitter with the data

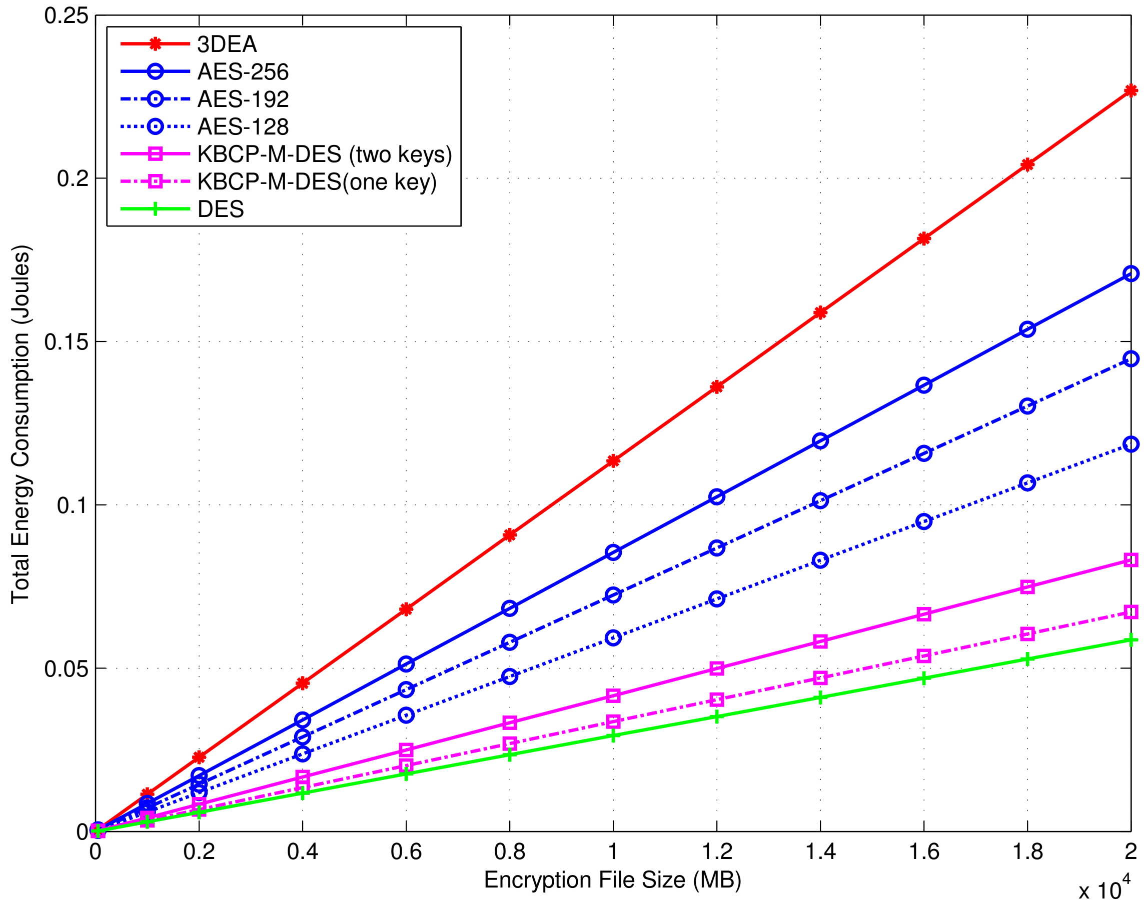


Figure 7.2. Energy consumption per byte for encrypting different file sizes using different encryption algorithms on the Intel Pentium Pro Microprocessor

after decryption at the receiver. Figure 7.6 shows the average bit error rate (BER) of the encryption algorithms under study versus the signal-to-noise ratio (SNR) of the channel. It can be inferred from the figure that AES in its 256 bits version has the lowest tolerance to errors imposed by the wireless channel due to the fact that one bit error at any received block of encrypted information will result in half the bits of that block to be decrypted in error due to the strict avalanche effect criterion (SAC). The performance of KBCP-M-DES in its one key version is similar to that of the two keys version, due to the fact that both transmits the same information but in different order. Therefore, the transmission energy for both encryption algorithms will be the same.

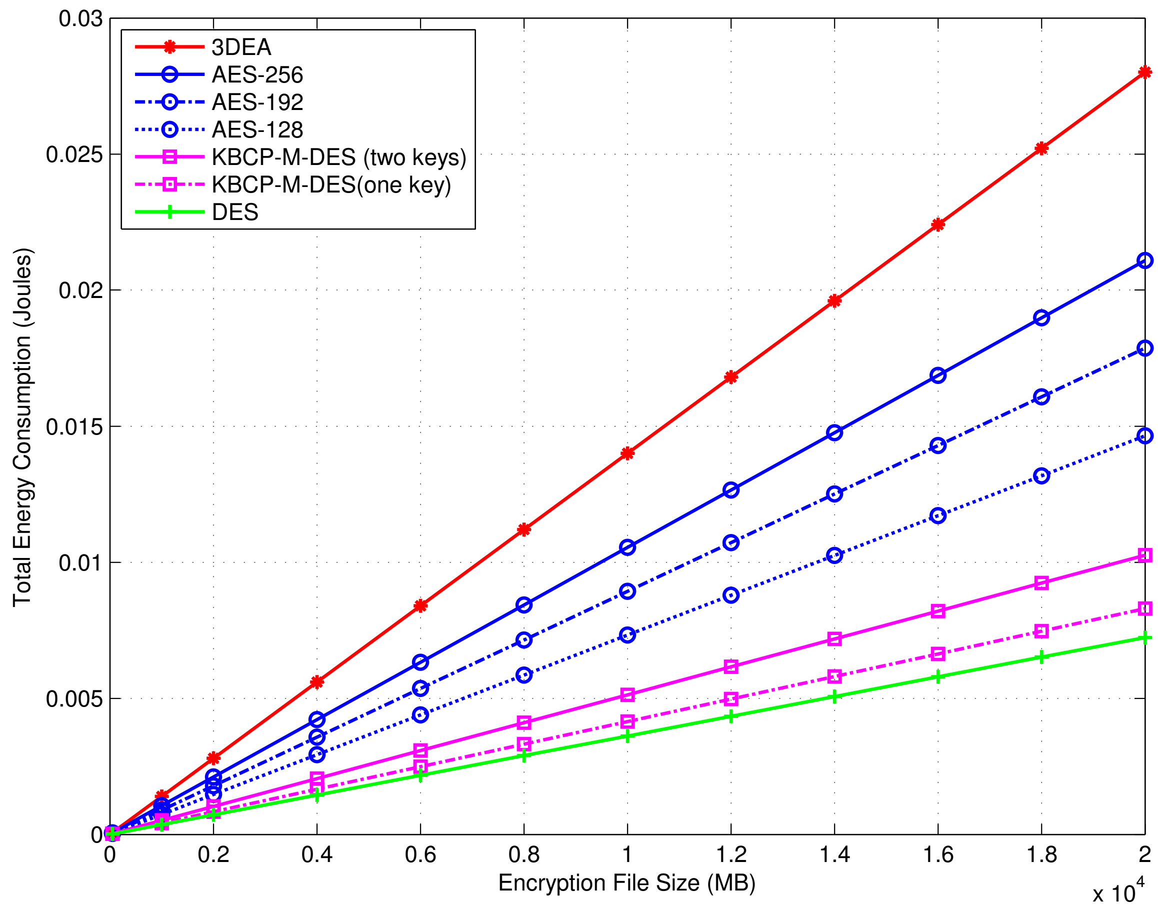


Figure 7.3. Energy consumption per byte for encrypting different file sizes using different encryption algorithms on the Intel Pentium 4 Microprocessor

Next, we compare the energy consumption of the encryption algorithms under study by comparing the loss in the SNR at given bit error rate values, where the SNR is the ratio of the energy of the signal divided by the average energy of the noise and is given by

$$SNR \quad = \quad 10log_{10}\frac{E_b}{N_0}, \tag{7.6.1}$$

where E_b is the energy per bit of the transmitted signal and N_0 is the average power density of the noise which is assumed to be constant and unity. Therefore, for a certain application with a certain required BER level, we can get the SNR values for the different encryption algorithms required to meet that application requirement. And since the noise is assumed to

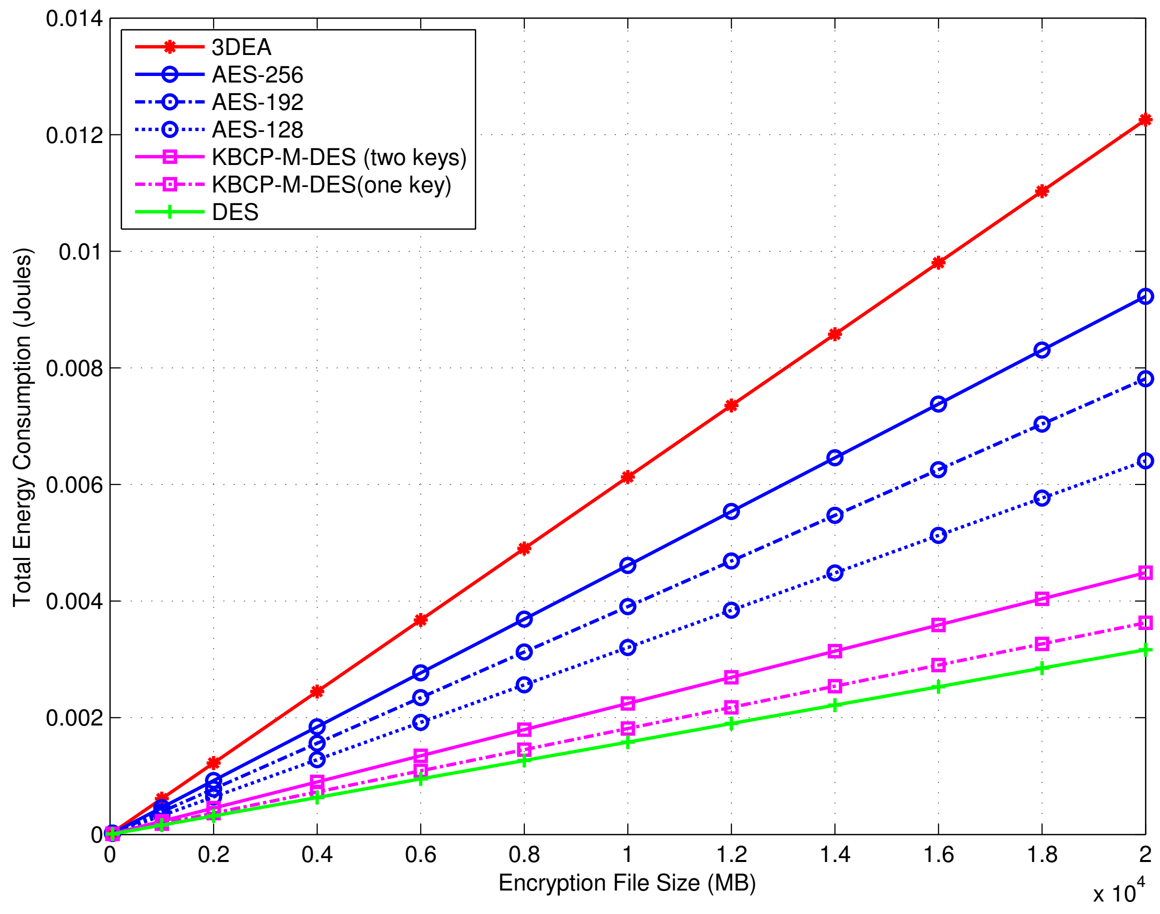


Figure 7.4. Energy consumption per byte for encrypting different file sizes using different encryption algorithms on the Intel Pentium M Microprocessor

be constant and unity we can find the energy of each transmitted encrypted signal. Table 7.3 shows the energy loss imposed by each encryption algorithm in regards to the no encryption case, for two different applications, which are voice application (i.e, $BER = 10^{-3}$) and video applications (i.e, $BER = 10^{-6}$).

Table 7.3 summarizes the energy loss of using AES in its 256 bits version, AES in its 192 bits version, AES in its 128 bits version, DES and KBCP-M-DES compared to the no encryption case, for two required bit error rates applications. It can be noticed from the table that the transmission energy of the information that is encrypted by KBCP-M-DES requires the least amount of energy compared to other encryption algorithm among

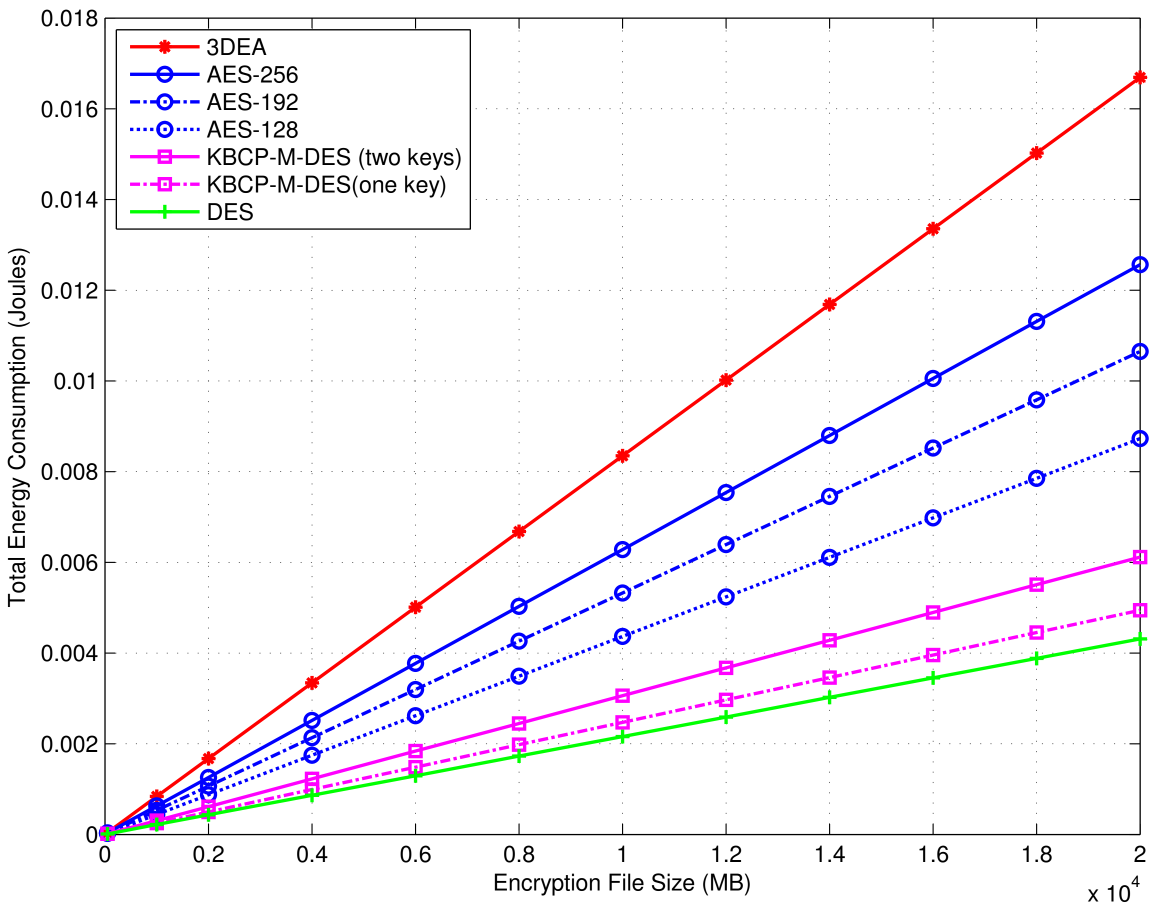


Figure 7.5. Energy consumption per byte for encrypting different file sizes using different encryption algorithms on the Intel Core Due Microprocessor

the encryption algorithms that are under the study. It can be also noticed that encrypting voice application data using KBCP-M-DES saves 56% in energy compared to that of using AES in its 128-bit version, while both algorithms are considered secure against applicable attacks as shown in previous chapters. We can also notice that KBCP-M-DES saves 42% of the transmission energy compared to DES whiles DES is considered insecure and subject to various types of attacks. Therefore using KBCP-M-DES in encryption saves 42% of energy in voice application and 16% in video application compared to DES, while being secure against attacks unlike DES. From the table we can also conclude that we can use a secure encryption algorithms with 29% increase in transmission energy compared to the transmission energy

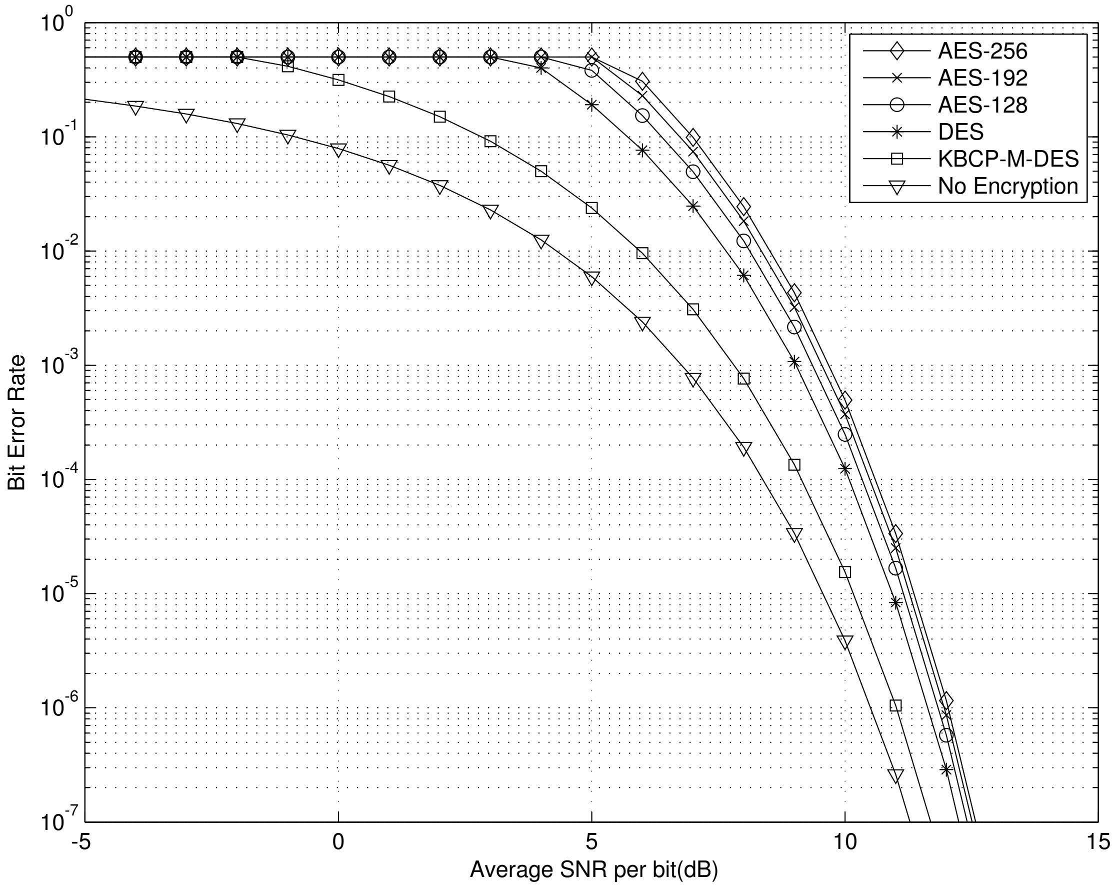


Figure 7.6. BER vs. SNR for a large amount of data encrypted using AES in its 256 bits version, AES in its 192 bits version, AES in its 128 bits version, DES, KBCP-M-DES and the no encryption case techniques in AWGN channel with BPSK modulation

when no encryption is present, for the case of voice applications transmission and only 14% increase in video applications transmission.

Figure 7.7 shows the energy loss percentage of using AES in its 256 bits version, AES in its 192 bits version, AES in its 128 bits version, DES and KBCP-M-DES compared to the no encryption case, for different required levels of bit error rate. It can be noticed from the figure that the energy loss imposed by using the KBCP-M-DES is less compared to other encryption algorithms under study especially for applications that requires a high value of bit error rate.

Table 7.3. Transmission Energy Increase of using AES in its 256 bit version, AES in its 192 bits version, AES in its 128 bits version, DES and KBCP-M-DES compared to the no encryption case in an AWGN channel with BPSK modulation

Encryption Technique	$BER = 10^{-3}$		$BER = 10^{-6}$	
	SNR Loss (dB)	Energy Loss	SNR Loss (dB)	Energy Loss
KBCP-M-DES	1.1	29%	0.56	14%
DES	2.23	71%	1.15	30%
AES-128	2.66	85%	1.36	36%
AES-192	2.84	92%	1.43	39%
AES-256	2.95	97%	1.5	41%

7.7 Conclusions

In this chapter we analyzed the energy consumption for a class of symmetric encryption algorithms in terms of the number of instructions required for encryption, given the time and energy required for each instruction. We also took into account the number of memory access required by each encryption algorithm to encrypt and decrypt information, which significantly affected the accuracy of our energy consumption analysis. As a result of the analysis, we found that DES and KBCP-M-DES uses the less energy followed by AES-128, AES-192, AES-256 and 3-DES. On the other hand, we used the bit error rate of the transmitted encrypted signal and the drop in the signal-to-noise ration as measures for the energy savings for the encryption algorithms under study. We analyzed the energy saving of AES in its 256, 192 bits version, AES in its 128 bits version, DES and KBCP-M-DES compared to the no encryption case. As a result, we concluded that the use of KBCP-M-DES saves almost 68% of the energy compared to AES in its 256 bit version for encrypted voice transmission application and around 27% for encrypted video transmission applications.

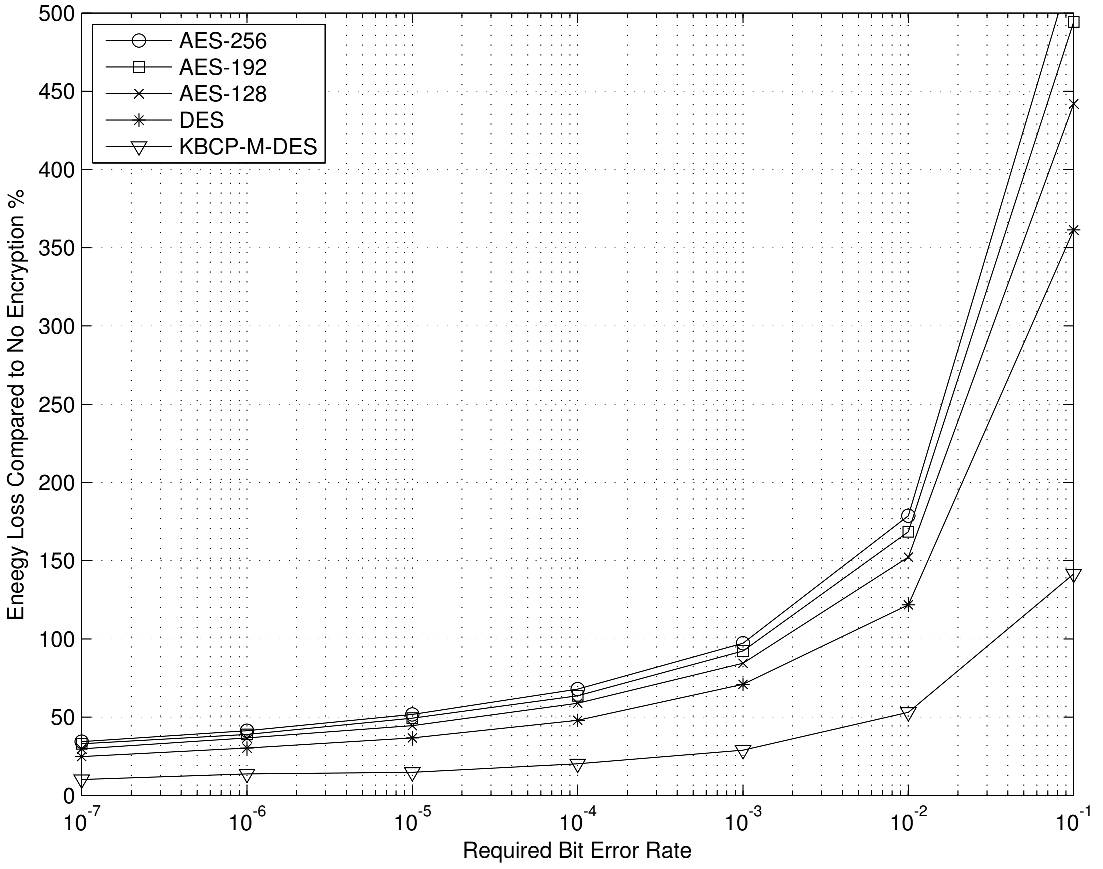


Figure 7.7. BER vs. percentage of energy loss of using AES, DES and KBCP-M-DES in encryption compared to the energy consumption of the no encryption case in AWGN channel with BPSK modulation.

Chapter 8

Conclusions and Future Work

In this dissertation, we propose new encryption algorithms that are more suitable to the nature of the wireless channel. We show that the proposed algorithms experience more reliable data transmission and less encountered errors at the receiver compared to standardized encryption algorithms that are known in the literature. We evaluate the security of the proposed algorithm by measuring the strength of the proposed algorithm to applicable attacks, and we show that it is impossible to attack our proposed algorithm within a reasonable amount of time. In addition, we use a new and comprehensive methodologies to evaluate the performance of the proposed algorithms, where we consider some factors that were ignored in similar works of the literature, such as the memory access time effect on the encryption algorithms complexity and time calculations. Moreover, due to the fact that new encryption algorithm should take into account the energy efficiency in wireless devices and applications which experience a limited resource of energy, we introduce a comprehensive energy consumption evaluation based on the energy consumption of encrypting the information using a certain microprocessor in addition to the energy consumption associated with the transmission of the encrypted information. We show that our proposed algorithms outperforms other secure standardized encryption algorithms in terms of their error performance, security, throughput, complexity and energy consumption. As a future work, we are interested in further evaluating the performance of the proposed encryption algorithm in MIMO diversity channels. we are also interested in further developing new encryption algorithms that are more efficient in wireless channels (better error performance, less complexity, less energy consumption and secure against applicable attacks).

BIBLIOGRAPHY

Bibliography

[1] D. W. Davies and S. Murphy, “Pairs and triplets of DES S-Boxes,” Journal of Cryptology, vol. 8, pp. 1-25, 1995.

[2] M. Matsui, “On correlation between the order of S-boxes and the strength of DES”, in Proceedings of Eurocrypt’94 , Lecture Notes in Computer Science vol. 950, pp. 366-375, Springer-Verlag, 1995.

[3] A.A. Sobhi Afshar, T. Eghlidos, M.R. Aref, “Efficient secure channel coding based on quasi-cyclic low-density parity-check codes”, *IET Commun.*, vol. 3, issue. 2, pp. 279-292, Feb. 2009.

[4] Q. Mao, L. Sun, C. Qin, “Joint Error Correction and Encryption Scheme Based on Turbo Codes”, *Proc. of the International Symposium on Intelligence Information Processing and Trusted Computing*, pp.503-506, Dec. 2010.

[5] O. Adamo and M. R. Varanasi, “Joint Scheme for Physical Layer Error Correction and Security”, *ISRN Communications and Networking*, Jan. 2011.

[6] J. Reason, End-to-end Confidentiality for Continous-media Applications in Wireless Systems, PhD Dissertation- University of California-Berkely, Fall 2000.

[7] W. Zibideh and M. Matalgah, ”Modified Data Encryption Standard with Improved Error Performance and Enahnced Security in Wireless Communication Channels,” *2011 IEEE Radio Wireless Symoysium*, Phenoix, AZ, Jan. 2011.

[8] M. Haleem, C. Mathur, R. Chandramouli, K.P. Subbalakshmi, "Opportunistic Encryption: A Trade-Off between Security and Throughput in Wireless Networks," *IEEE Transaction on Dependable and Secure Computing*, vol. 4, no. 4, pp. 313-324, Oct. 2007.

[9] Y. Xiao, B. Sun, H. Chen, S. Guizani and R. Wang, "Performance Analysis of Advanced Encryption Standard(AES)", *IEEE Global Communication Conference GlobComm*, St. Louis, MO, USA, Nov. 2005.

[10] F. Granelli and G. Boato, "A novel methodology for analysis of the computational complexity of block ciphers: Rijndael, Camellia and Shacal-2 compared," *The 3rd Conference on Security and Network Architectures (SAR'04)*, La Londe, Cote d'Azur, France, June 2004.

[11] Radu Tomoiaga and Mircea Stratulat, "AES Performance Analysis on Several Programming Environments, Operating Systems or Computational", *2010 Fifth International Conference on Systems and Networks Communications Platforms*, Nice, France, Aug. 2010.

[12] J. Kim and I. Park, "Double-Binary Circular Turbo Decoding Based on Border Metric Encoding," *IEEE Transactions on Circuits and Systems*, pp. 79-83, vol. 55, no. 1, Jan. 2008.

[13] J. Grob, R. Avanzi, E. Savas and S. Tillich, "Energy-Efficient Software Implementation of Long Integer Modular Arithmetic," *Procedings of the Seventh International Workshop Cryptographic Hardware and Embedded Systems*, pp. 75-90, Edinburgh, UK, Sep. 2005.

[14] B. Soewito, L. Vespa and N. Weng, "Characterizing Power and Resource Consumption of Encryption/Decryption in Portable Devices," *2008 IEEE Region 5 conference*, pp. 1-6, Kansas City, MO, Apr. 2008.

[15] Creighton T. R. Hager, S. F. Midkiff, J. Park and T. L. Martin, “Performance and Energy Efficiency of Block Ciphers in Personal Digital Assistant,” *the proceddings of the 3rd IEEE International Conference on Pervasive Computing and Communications*, pp. 127 - 136, Kauai, Hawaii, Mar. 2005.

[16] Jagadeesh Kaza and Chaitali Chakrabarti, “Design and Implementation of Low-Energy Turbo Decoders,” *IEEE Transactions on Very Large Scale Integrated Scale*, pp. 968-977, vol. 12, no. 9, Sep. 2004.

[17] S. Chouhan, R. Bose and M. Balakrishnan, “Integrated Energy Analysis of Error Correcting Codes and Modulation for Energy Efficient Wireless Sensor Nodes,” *IEEE Transactions on Wireless Communications*, pp. 5348-5355, vol. 8, no. 10, Oct. 2009.

[18] Ali Iranli, Hanif Fatemi and Massoud Pedram, “A Game Theoretic Approach to Dynamic Energy Minimization in Wireless Transceivers,” *the procedings of the International Conference on Computer Aided Design ICCAD*, pp. 504-509, San Jose, CA, November 2003.

[19] Behrouz A. Forouzan, *Cryptography and Network Security.* First Edition, Mc Graw Hill, 2008.

[20] W. Stallings, *Cryptography and Network Security, Principles and Practice.* First Indian Edition, Pearson Educational, 2003.

[21] E. Biham and A. Shamir, “Differential Cryptanalysis of the Full 16-Round DES,” *Advances in Cryptology CRYPTO 92, Lecture Notes in Computer Science*, vol. 740, pp. 487-496, Springer-Verlag, 1993.

[22] L. R. Knudsen and J. E. Mathiassen, “A chosen-plaintext linear attack on DES”, *Lecture Notes in Computer Science, Fast Software Encryption, FSE 2000*, vol. 1978, pp. 262-272, Springer-Verlag, 2001.

[23] Matsui, M. and Yamagishi, A. ”A new method for known plaintext attack of FEAL cipher,” *Advances in Cryptology CRYPTO 92, Lecture Notes in Computer Science*, vol 658, pp. 81-91, 1993.

[24] M. Matsui, “The first experimental cryptanalysis of the data encryption standard,” *Advances in Cryptology CRYPTO 94, Lecture Notes in Computer Science*, vol 839, pp. 1-11, Springer-Verlag, 1994.

[25] National Institue for Standards and Technology, Data Encryption Standard (DES), FIPS 46-3, US Department of Commerce, May, 1999.

[26] Electronic Frontier Foundation (EFF), DES cracker. http://www.eff.org/DEScracker/, 1998.

[27] J. H. Moore and G. J. Simmons, “Cycle structures of the DES with weak and semi-weak keys,” *in Advances in Cryptology - CRYPTO’86, Lecture Notes in Computer Science*, vol. 263, pp. 9-32, Springer-Verlag, 1987.

[28] Coppersmith, “The Data Encryption Standard (DES) and its strength against attacks,” *IBM Journal of Research and Development*, vol. 38, no. 3, pp. 243-250, 1994.

[29] I. Damgard and L. R. Knudsen, “Two-key triple encryption,” *Journal of Cryptology*, vol. 11, no. 3, pp. 209-218, 1998.

[30] National Institue for Standards and Technology, Advanced Encryption Standard (AES), FIPS 197, US Department of Commerce, November 26, 2001.

[31] Raphael Chung-Wei Phana, “Impossible differential cryptanalysis of 7-round advanced encryption standard (AES),” *Information Processing Letters*, vol. 91, pp. 33-38, July 2004.

[32] Bahrak, B. Aref, M.R., "Impossible differential attack on seven-round AES-128," *IET Information Security*, pp. 28-32, June 2008.

[33] C. N. Mathur, K. Narayan, and K. P. Subbalakshmi, High diffusion cipher: encryption and error correction in a single cryptographic primitive, *4th Internernational Conference on Applied Cryptography and Network Security*, Singapore, June 2006.

[34] H. A. Al Hassan, M. Saeb and H. D. Hamed, "The PYRAMIDS Block Cipher," *International Journal of Network Security*, vol. 1, no. 1, pp. 5260, July 2005.

[35] R. J. McEliece, "A Public-Key Cryptosystem Based On Algebraic Coding Theory," *DSN Progress Report 42-44: 114*, Feb. 1978.

[36] Ohta Kazuo, Pei Dingyi, eds. "Cryptanalysis of the Original McEliece Cryptosystem," *Advances in Cryptology ASIACRYPT98, Lecture Notes in Computer Science*, vol. 1514, pp. 187199, Springer-Verlag, 1998.

[37] C. S. Park. Improving code rate of McElieces public-key cryptosystem, *IET Electronics Letters*, vol. 25, pp. 14661467, Oct 1989.

[38] M. C. Lin and H. L. Fu. Information rate of McElieces public-key cryptosystem, *IET Electronics Letters*, vol. 26, pp. 1618, 1990.

[39] Kak, S.C., "Encryption and Error-Correction Coding Using D Sequences," *IEEE Transactions on Computers*, Sep. 1985.

[40] Godoy, W. and D. Periera,"A proposal of a cryptography algorithm with techniques of error correction," *Computer Communications* vol. 20-15, pp. 13741380, 1997.

[41] Hwang, T. and T. Rao, "Secret Error-Correcting Codes (SECC)," *Advances in Cryptology CRYPTO 88, Lecture Notes in Computer Science*, vol. 403, pp. 540-563, 1990.

[42] D. Gligoroski, S. Knapskog, and S. Andova, "Cryptcoding encryption and errorcorrection coding in a single step," *International Conference on Security and Management*, Las Vigas, NV, USA, June 2006.

[43] Tom Shanley, x86 Instruction Set Architecture, Comprehensive 32/64-bit Coverage. First Edition, MindShare, Inc., 2009.

[44] Martin Bossert, Channel Coding for Telecommunications, Wiley 1st edition, Oct. 1999.

[45] J. Nechvatal, *Public Key Cryptography*, NIST-USA, 1992.

[46] R. Rivest, A. Shamir, L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems." Communications of the AC, pp. 120126, 1978.

[47] C. Pandit, J. Huang, S. Meyn, M. Medard, and V.V. Veeravalli, "Entropy, Inference, and Channel Coding," *The IMA Volumes in Mathematics and its Applications*, vol. 143, pp. 99-124, June 2007.

[48] Ed Grochowski, Murali Annavaram, "Energy per Instruction Trends in Intel Microprocessors," Technology at Intel Magazine, pp. 18, Mar. 2006.

[49] Murali Annavaram, Ed Grochowski, John Shen, "Mitigating Amdahls Law Through EPI Throttling," *Proceedings of the 32nd International Symposium on Computer Architecture (ISCA05)*, pp. 298-309, Madison, Wisconsin, June 2005.

[50] M. Razvi Doomun, KM Sunjiv Soyjaudah, Devesh Bundhoo, "Energy Consumption and Computational Analysis of Rijndael-AES," *Procedings of the 3rd IEEE/IFIP International Conference in Central Asia on Internet*, pp. 1-6, Tashkent, Uzbekistan, Sep. 2007.

[51] P. V. Krishnamurthy, "Analysis of energy consumption of RC4 and AES algorithms in wireless LANs," *procedenings of the IEEE Global Telecommunication Conference GLOBECOM*, pp. 1445-1449, San Fransisico, CA, Dec. 2003.

[52] A. Olteanu and Y. Xiao, "Fragmentation and AES Encryption Overhead in Very High-Speed Wireless LANs," *IEEE International Conference on Communication (ICC) 2009*, Dresden, Germany, June 2009.

[53] A. Olteanu and Y. Xiao, "Security Overhead and Performance for Aggregation with Fragment Retransmission (AFR) in Very High-Speed Wireless 802.11 LANs," *IEEE Transactions on Wireless Communications*, vol. 9, no. 1, Jan. 2010.

[54] F. Granelli and G. Boato, A novel methodology for analysis of the computational complexity of block ciphers: Rijndael, Camellia and Shacal-2 compared," *The 3rd Conference on Security and Network Architectures (SAR04)*, June 2004.

[55] Y. Xiao, B. Sun, H. Chen, S. Guizani and R. Wang, "Performance Analysis of Advanced Encryption Standard(AES)", *IEEE Global Communication Conference GlobComm 2005*, St. Louis, MO, USA, Nov. 2005.

Appendices

Appendix A

Error Performance of the RSA Algorithm in Wireless Channels

Asymmetric encryption is different than symmetric encryption in the way it keeps the system secret and thus the information secret. In symmetric key cryptography the secrecy of the key was shared between two or more users. However, in asymmetric key cryptography each user maintains his own key secret. The main difference between the two approach is that in symmetric key cryptography, the same key is used for encryption and decryption and this key is shared between the two or more entities of the communication. On the other hand, in asymmetric key cryptography the encryption and decryption are performed using different keys [45].

In this approach, each user generates two different keys. The first key is called the public key while the other is called the private key. Each user should then broadcast to the other users his public key, while keeping the private key secret and hidden from every other user. If user A want to send an encrypted message to user B, user A should encrypt the message with user's B public key. Since user B has the private key he should be able to decrypt the encrypted message. If he could not decrypt it, that means that the message was not encrypted with his public key and hence the message is not intended for him. One of the other differences between the two encryption approaches is that in asymmetric encryption the plaintext and ciphertext are treated as integer numbers, therefore the encryption and decryption process are sets of mathematical functions on these integers. However, symmetric encryption as we already discussed is based on bit manipulation operation on the data in its binary format. The functions used in asymmetric encryption should be trapdoor one way function. A trapdoor one way function is a function that satisfies the following

1. Given x, $y = f(x)$ is easy to compute.
2. The inverse of the function is difficult to compute. Given $y = f(x)$, it is difficult to find $x = f^{-1}(x)$.
3. Given y and a trapdoor, x can be easily computed.

Some examples of algorithms that are based on asymmetric key cryptography are

1. RSA cryptosystem
2. ElGamal cryptosystem
3. DSS (Digital Signature Standard)
4. Diffie-Hellman key exchange protcol
5. various elliptic curves cryptosystem
6. Paillier cryptosystem

The asymmetric encryption approach has been used by many protocols such as

1. GPG software (GNU Privacy Guard)
2. Internet key exchange protocol
3. PGP computer program (Pretty Good Privacy)
4. SSH network protocol (Secure Shell)

A.1 Algorithm Design and Structure

RSA is one of the most common asymmetric (public) key cryptosystem, it is named for its inventors Rivest, Shamir and Adleman [46]. It was the first algorithm to be used in signing and encryption, it is considered secure given that the keys used are long enough. As any other public key algorithm, RSA has two major phases which are the key generation phase and the encryption/decryption phase.

A.1.1 Key Generation

RSA uses two keys, public and private keys. The public key is known to everyone, and is used for encryption. Messages encrypted with the public key can only be decrypted by the corresponding private key, which is known to its own creator only. The two keys are generated by the following procedure:

1. Two distinct prime numbers p and q are chosen.
2. Compute $n = pq$
3. Compute $\phi(n) = (p-1)(q-1)$
4. An integer e is chosen, such that $1 < e < \phi(n)$ and $gcd(e, \phi(n)) = 1$
5. Compute $d = e^{-1}(mod\phi(n))$ (often computed using the Extended Euclidian algorithm)
6. The tuple (e, n) is the public key which will be announced, while d is the private key

A.1.2 Encryption and Decryption

Assuming that Alice wants to transmit a message M to Bob, the encryption procedure is desribed by the following procedure:

1. Given that Bob already broadcasted his public key (e, n), Alice has Bob's public key.
2. Alice turns M into an integer m where $0 < m < n$ using an agreed upon protocol.

3. Alice then computes and transmits the cipher $c = m^e(modn)$, this operation can be easily done using the fast exponentiation algorithm.

When Bob receives the cipher c, the decryption process is simply done using the private key d to recover the original message $m = c^d(modn)$.

A.1.3 Simple Example

1. Choose two distinct primes $p = 61$ and $q = 53$.
2. Compute $n = pq = 61 \times 53 = 3233$.
3. Compute $\phi(3233) = (61 - 1) \times (53 - 1) = 3120$.
4. Choose e, such that $1 < e < 3120$ and $gcd(e, \phi(3233) = 1$, let $e = 17$.
5. Compute $d = e^{-1}(mod\phi(n)) = 2753$, the multiplicative inverse of e.
6. The public key is $(e = 17, n = 3233)$, the private key is $d = 2753$.
7. Lets assume that the message that needs to be encrypted is $m = 65$, the ciphertext c equals $65^{17}(mod3233) = 2790$.
8. To decrypt the cipher, m can be calculated by $m = 2790^{2753}(mod3233) = 65$.

A.1.4 Security

Factorization Attack

The security of RSA is based on choosing large modulus n and keeping p and q secret, if an attacker can factorize the modulus n and find p and q then he will be able to compute d. To have RSA secure, the modulus n should be more than 300 decimal digits, that means that n should be at least 1024 bits long. Factorizing such a number would be impossible in a feasible time. There are many other attacks on RSA such as follows:

- Chosen ciphertext attack.

- Encryption exponent attack.
- Related message attack.
- Short pad attack.
- Revealed decryption exponent attack
- Broadcast attack.
- Low decryption exponent attack.
- Short message attack.
- Cycling attack.
- Unconcealed message attack.
- Common modulus attack.
- Timing attack.
- Power attack.

Some recommendation need to be considered in order to have RSA secure.

1. n should be at least 1024 bits.
2. p and q must each be at least 512 bits.
3. The values of p and q should not be very close to each other.
4. Both $p-1$ and $q-1$ should have at least one large prime factor.
5. The modulus n should not be shared.
6. The value of e should be $2^{16}+1$ or integer close to this value.

7. If the private key d is leaked, the sender must immediately change n as well as both e and d.

A.1.5 Complexity

The complexity of RSA is imposed by the following factors.

- Since n should be at least 1024 bits, and both p and q should be at least 512 bits, the memory needed in the key generation, encryption and decryption should be adequate to these minute requirements.
- Each user has to keep a large portion of information secret (n, d), otherwise the algorithm can be easily cracked.
- Depending on the application, it is not always easy to represent the data in an integer format.

A.2 Error Performance in the Wireless Channel

In this section we are studying and evaluating the error characteristics of the RSA encryption algorithms. We are assuming that one or more of the transmitted (encrypted) bits are received in error. Therefore, different cipher will be decrypted at the receiver, which leads to a different plaintext after decryption. We already shown in previous chapters that the average probability of error when the data encryption standard is used is 0.5. Figure A.2 shows that using RSA for encryption and decryption will result in a random average probability of error at the receiver, even if the number of runs is increased. It is shown form the figure if we have 500,000 runs this will result in an average probability of error of 0.6. Similarly, it can be clarly noticed from the figure that if the number of runs increases to a million run, the average probability of error dose not converge to 0.5 as the case of DES.

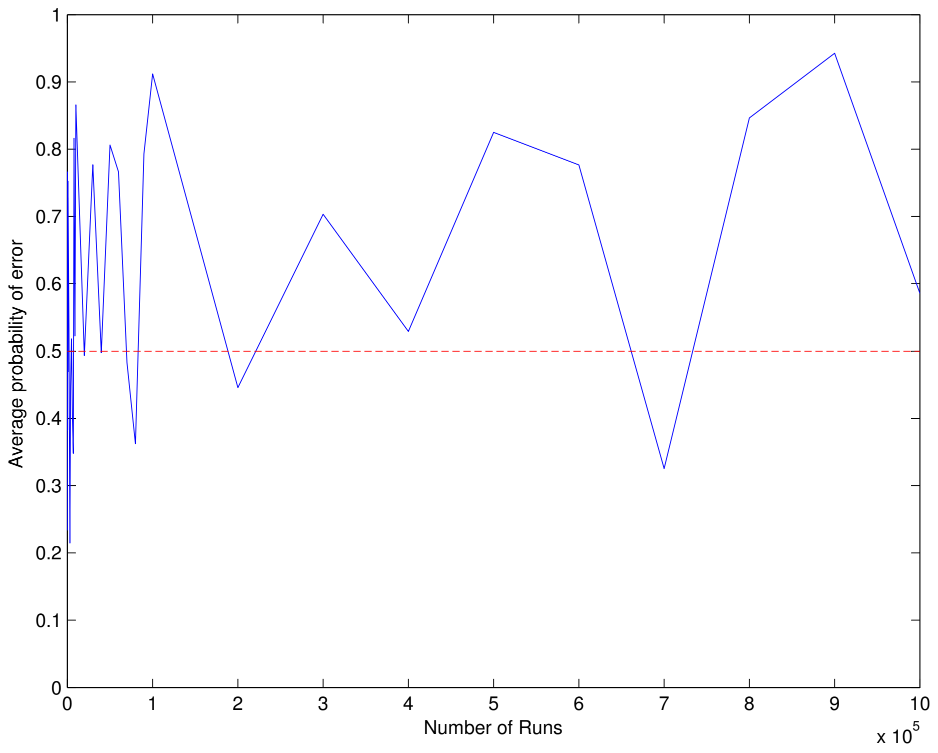


Figure A.1. Average Probability of error over RSA

Figure A.2 shows another experiment for three bits in error at the receiver. The observations are the same as those of the previous figure.

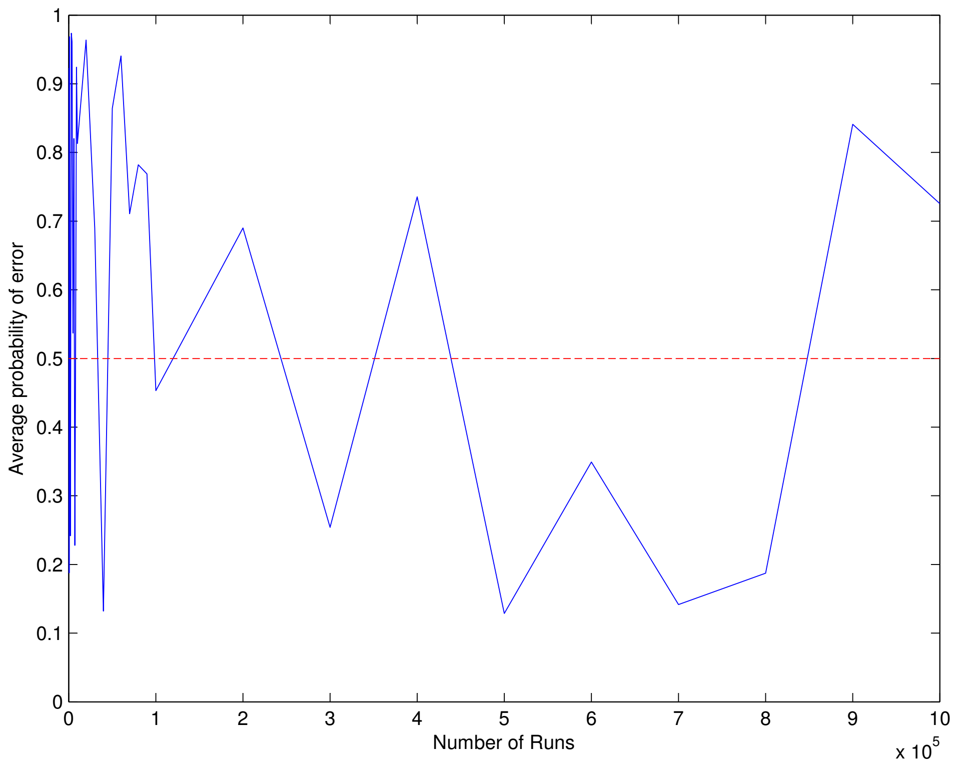


Figure A.2. Average Probability of error over RSA

A.3 Comments

It can be clearly noticed that asymmetric key cryptiosystem and RSA as an example do not satisfy the avalanche effect criteria, and the average error probability is different from run to run, and it dose not converge when the number of runs increase. Therefore, asymmetric key algorithms are out of the scope of this work.

VITA

Walid Y. Al Zibideh received the B.S. degree in computer engineering from Jordan University of Science and Technology (JUST), Irbid, Jordan in June, 2007. He obtained the masters of science in Telecommunications from the University of Mississippi, Oxford, MS, USA in May 2010. He is currently a Ph.D. candidate at the University of Mississippi and is expected to graduate by May 2013. During his study at the Electrical Engineering department at the University of Mississippi, he worked as a Teaching Assistant for many courses and was the lab instructor for the Digital Logic Design and Circuits Design Labs. He also worked as a Research Assistant at the same department were he was part of many projects involving different aspects of wireless communications. He joined Qualcomm Inc. at San Diego, CA as a Modem System Test Engineer in September 2011, and is currently working as a power management (PMIC) test Engineer at Qualcomm Technologies Inc. at San Diego, CA.