Journal of Computer Science

Original Research Paper

Randomness Analysis of DES Ciphers Produced with Various Dynamic Arrangements

[1]Malik Qasaimeh and [2]Raad S. Al-Qassas

[1]Department of Software Engineering,
[2]Department of Computer Science,
Princess Sumaya University for Technology, P.O. Box 1438 Amman 11941, Jordan

Article history
Received: 02-09-2017
Revised: 24-11-2017
Accepted: 14-12-2017

Corresponding Author:
Malik Qasaimeh,
Department of Software Engineering, Princess Sumaya University for Technology,
P.O. Box 1438 Amman 11941,
Jordan
Email: m.qasaimeh@psut.edu.jo

Abstract: Over the past few years, researchers have devoted efforts to enhance the original DES encryption algorithm. These enhancements focus on improving multiple perspectives of the algorithm through enhancing its internal components to deliver a robust DES variant against different kinds of typical attacks such as linear and differential crypto analysis, in addition to the newly evolved attacks such as differential power analysis attacks. In fact the output of existing solutions have enhanced the ciphertext randomness. This paper introduces two encryption algorithms that enhance the original DES named DDES and HDES. DDES is mainly based on a secure selection of both S-boxes and P-box arrangements during each encryption round, it has also extended the key length by adding two more keys beside the original one to the encryption process. HDES, on the other hand, uses a hash function to generate a random fingerprint for each plaintext block. This fingerprint is used to generate the seed to produce round seeds that are used to select secure S-boxes only for each round in the encryption process. These two variants meet with certain demands that are imposed by the user applications context. DDES provides a higher ciphertext randomness with some added processing time, while HDES provides a relatively secure variant with lower processing time. These two variants can provide alternatives depending on the targeted applications that require different levels of security and processing time. DDES and HDES have been evaluated and compared against DES, DESX and 3DES, using a number of metrics including chi-square test, cipher data difference, hamming distance and processing time.

Keywords: DES, S-Boxes, P-Box, Encryption Algorithm, Chi-Square Test

Introduction

In 1975 the National Institute of Standards and Technology (NIST) has released the Data Encryption standard (DES) with a free license for its use. The first official version of the encryption standard FIPS-46 was released in 1977. The standard was revised three times later: FIPS-46-1 in 1988, FIPS-46-2 in 1993 and FIPS-46-3 in 1999 (De Cannière, 2011). Since the release of DES, its mysterious S-boxes and its 56-bit secret key resulted in controversy and some distrust among the research community (Van Tilborg and Jajodia, 2011). DES is a block cipher with 64-bit block size, it uses only 56 bits during the encryption process while the rest are reserved for error detection and correction.

Studies in the literature (Biham and Shamir, 1991; Kumar *et al*., 2006; Zodpe *et al*., 2012; Lee, 2013) claim that the DES key length would make the algorithm vulnerable for many kinds of attacks like brute force attacks and more advanced attack such as linear cryptanalysis and differential cryptanalysis. For example, in 1990, Biham and Shamir (1991) proposed a differential cryptanalysis method that could be used to attack DES. The method would be more efficient than exhaustively searching all possible keys if the algorithm used at most 15 rounds instead of 16 rounds. However, a few years later, IBM released some details about DES design criteria, which showed that indeed the 16 rounds of the standard had constructed the system to be resistant to differential cryptanalysis. In 1998, the Electronic Frontier Foundation (EFF) built deep crack machine in

response to DES challenge II. The deep crack decrypted a DES encrypted message in only 56 hours. Six months later, in response to DES Challenge III and in collaboration with distributed.net, the EFF used deep crack to decrypt another DES-encrypted message, for which the operation took 22 h and 15 min (Kumar *et al*., 2006).

The design of data encryption algorithms should be immune and robust against various types of attacks that may threat the encryption algorithm including differential power analysis (Kocher *et al*., 2011), differential cryptanalysis (Biham and Shamir, 1991) and linear cryptanalysis (Matsui and Yamagishi, 1993) attacks. The Differential Power Analysis (DPA) is applied on the crypto data and employs statistical methods to reveal the encryption key. Attackers analyze the power singles of the crypto-circuit and analyze the part of crypto data during execution process to reveal the key. Hiding and masking techniques can be used to protect DES circuits from DPA attacks. However, the hiding technique does not provide a full protection against advanced DPA attacks such as second-order attack.The differential cryptanalysis attack depends on the obtainability of ciphertext and plaintext sets. The key is derived by selecting pairs of plaintext related by a particular difference, then differences of corresponding ciphertexts are computed. The attacker analyses the statistical patterns of the resulting pairs of differences in order to discover the inner entries of the S-boxes used in the encryption process. On the other hand, linear cryptanalysis is based on the probability of occurrences in linear expressions relating plaintext, ciphertext and sub-key bits. It is known as plaintext attack, where the attacker has information on plaintext sets and their corresponding ciphertexts. In many scenarios and applications it is rational to assume that the attacker has knowledge of plaintext sets and their corresponding ciphertexts (Matsui and Yamagishi, 1993; Matsui, 1993). It should be noted that linear cryptanalysis and differential cryptanalysis attacks rely on the order and the content of the DES S-boxes (Schneier, 1996).

The correlation between the plaintext and the ciphertext has been investigated in (Yun-peng *et al*., 2009; Matsui, 1994). The higher level of correlation enable the attackers to infer the encryption key or directly the original plaintext. Different modification can be performed in general for the encryption algorithms to de-coupling the high level of correlations between the plaintext and the ciphertext. These recommendations focus on the dynamic design of the substitution process and the permutation function. Besides, some relevant principles can be used to decrease the correlation between the ciphertext and the plaintext and also the correlation between the encrypted symbols. Verdult (2015) stated that using parallel computing the amount of chosen plaintext that is required to break the DES algorithm is now considered very small with the high processing power. Besides, the simple XOR and short key length function are responsible for weaknesses in DES.

A number of studies (Biryukov and Wagner, 2000; Zhuang *et al*., 2014; Biham and Biryukov; 1995, Verma and Prasad, 2009; Sison *et al*., 2012; Kilian and Rogaway, 2001) have been proposed to improve the original DES against the attacks described above. Diffie and Hellman proposed triple DES (Biryukov and Wagner, 2000). The idea was to multiple encrypt the block using DES with three different keys. However, triple DES adds more constraints on the key sharing and management protocols, besides triple DES is slower than the original DES that was never designed to be used in this way (NIST, 2012).

Zhuang *et al*. (2014) proposed an improvement on DES circuit against DPA attack. The improvements consist of two components which are secured S-boxes and rotating masks that are implemented at hardware level. To protect the linear parts of DES algorithm, the masks are rotated after each encryption round and then the hiding method is used by adding two extra inputs to the S-boxes which are count and position. The count input is used as a counter for the current encryption round and the position is used as a pointer for the entries in the S-boxes to perform changes on the S-boxes position value based on the count value

Biham and Biryukov (1995) proposed a DES variant that perform a key-dependent transformation to the S-boxes to rearrange the original eight S-boxes in different ways using the last 5 bits of the key. They showed that not all the 8! Possible ways to arrange the eight boxes are valid ones. In fact some of the S-boxes combinations can make DES weaker against the linear and differential cryptanalysis. The limitation of this approach is the high probability of repeating the S-boxes arrangement in different rounds.

Verma and Prasad (2009) proposed modifications to DES by first dividing the expanded right part of 48 bits into two parts each of 24 bits, then two different functions are applied to each of these two part. The key length was also increased to 112 bits by using tow keys. An analysis of these modifications has shown that modifying the F-function enhances the DES diffusion, which means each input bit affects many output bits. Also, an analytical proof has shown that the modifications proposed on the key enhances confusion in the difference. However, the proposed approach requires 50% more time than the original DES.

The advancement of internet applications and new technologies allow easier and quicker access to users' sensitive data. The unauthorized disclosure, alteration or destruction of the sensitive data could cause a significant level of risk to the affiliates (Friedewald *et al*., 2010). Sensitive data may include individual's ethnic origin,

political opinions, religious beliefs physical or mental health. Usually, the sensitive data are protected by laws, regulations, or policies that require the adoption of proper data encryption algorithm. For example, Health Insurance Portability and Accountability Act (HIPAA) security rule, requires in §164.308(a)(4) that health care organizations should "Implement technical policies and procedures for electronic information systems that maintain electronic protected health information to allow access only to those persons or software programs that have been granted access rights". Other regulations such as Payment Card Industry (PCI) that regulate the process of smart cards mentioned that "Sensitive information must be encrypted during transmission over networks that are easily accessed by malicious individuals. Misconfigured wireless networks and vulnerabilities in legacy encryption and authentication protocols continue to be targets of malicious individuals who exploit these vulnerabilities to gain privileged access to cardholder data environments". This requires fast and efficient encryption process that is able to protect the individual privacy. Any delay caused by the selected encryption algorithm will disturb the associated verification processes such as identity verification and authentication process.

The selection of the encryption algorithm for the sensitive data in an organization is influenced by many issues beside of the application type. Dong *et al*. (2015) mentioned that efficient and sometime fast encryption algorithm is required in a big data platform especially when the data are transmitted from a data owner's local server over multiple transmission component, the selection of the encryption algorithm should be made based on the criticality and classification of the data. For this reason, Sison *et al*. (2012) proposed an improved DES variant that can be used to secure the smart card data, the improvement made by odd-even substitution component to DES. The average running time through different attempts with the modified DES was 365.2 millisecond while 355.8 millisecond with the typical DES. The authors conclude that such improvement could be useful for smart cards applications since the modified DES enhanced the typical DES with almost equivalent average running time. Other research works on cloud computing propose classification frameworks to classify the data based on their level of sensitivity as to select the suitable encryption algorithm for securing the data before transmission (Tawalbeh *et al*., 2015; Shaikh and Sasikumar, 2015).

This paper analysis the randomness of two DES variants that are designed to support the security of different applications that mandates variable characteristics of data encryption requirements such as processing speed and level of sensitivity. The first variant is called the Dynamic Data Encryption Standard (or DDES for) that is mainly based on a secure selection of both S-boxes and P-box arrangements during each encryption round (Alnoury *et al*., 2016). The S-boxes and P-box arrangements selection depends on a random seed, which results in dynamic selection of the secured combination of S-boxes and P-box that differs each round. In DDES we have extended the key length by adding two more keys beside the original one. The second variant is called Hashed Data Encryption Standard (or HDES for short) that uses a hashing component to produce a random fingerprint for each plaintext block. This fingerprint is used along with the encryption key to produce the seed to generate round seeds that are used to select secure arrangements of S-boxes for each round in the encryption process (Al-Qassas *et al*., 2016). It is worth noting that although some of the components of DDES and HDES hold the same name, their internal processes are different.

The rest of this paper is organized as follows. Sections 2 and 3 illustrate the operation of DDES and explain its components. Sections 4 and 5 describe the operation of HDES and explain its components. Section 6 studies the performance of the proposed methods against well-known DES variants. Finally, Section 7 concludes the work and provides potential future directions.

DDES Overview

The enhancements of DDES over the original DES (Kim, 1991) can be described as follows. DDES dynamically generates different permutations of the S-boxes and P-boxes for each encryption round. This approach overcomes the original DES where it selects only one arrangement of the S-boxes for the sixteen rounds. This is based on an optimization technique that uses a random seed to organize the relationship between the S-boxes and the generated P-box arrangements during each round of the encryption process. The seed is generated by mapping the plaintext using three keys through the seed generator component. DDES then selects the arrangements from a secure pool of S-boxes and P-box in every encryption round, this pool has been verified in (Biham and Biryukov, 1995; Brown and Seberry, 1990). For the decryption process, the seed used to select the S-boxes and P-box is embedded within the ciphertext so that the receiver can use it to build the boxes in the reverse order.

DDES uses secured combinations of S-boxes and P-boxes to provide robustness against different types of attacks such as differential attacks. Having S-boxes and the P-box arrangements change every single round along with key whitening, would make the attacks very difficult, specially that even related texts which are needed in differential attacks are not encrypted in the same way. DDES has included extra strength against brute force attacks as it has longer key (Rogaway, 1996).

The components of DDES are described as follows: Pseudorandom generator (PRG), seed generator, boxes generator, seed filter and seed distributor. The seed generator creates a seed using the given three encryption keys. The PRG uses this seed to produce random numbers known as subseeds, which are fed into the boxes generator to allow dynamic generation of S-boxes and P-box arrangements for each round. After completing all encryption rounds, the seed distributer will insert the seed within the resulted ciphertext. At the receiver side, in order to perform the decryption process, the seed is extracted from the cihpertext using the seed filter component. This seed is used to produce the same arrangements of S-boxes and P-box used in the encryption process, but in reverse order. Section 3 provides detailed description for each of these components.

To illustrate the operation of DDES, the encryption process is shown in Fig. 1 and its pseudocode is presented in Fig. 2. An overview of the encryption process in DDES is described in the following steps:

1. The seed generator generates a unique seed, based on the plaintext and three encryption keys K_1, K_2 and K_3. The seed then used for the process of generating the S-boxes and P-box arrangements. This seed is fed to PRG to produce a series of subseeds. The PRG produces two subseeds for each round, which are fed into the boxes generator to select secured S-boxes and P-box arrangements
2. The plaintext is XORed with K_1
3. Split the text from step 2 into two parts, left and right, each of size 32 bits
4. The key scheduler generates a round key from K_2. The round key then XORed with the right part after it is expanded to 48 bit. This step proceeds with substituting the result to the S-boxes acquired from the boxes generator, the resulting text then permuted by the P-box and then XORed with the left part
5. The right 32 bits in step three will become the left part of the new text and the 32 bit obtained from step four will become the right part of the new text. The new text will be XORed with K_3 after the 16 rounds of the encryption process to generate the ciphertext
6. Finally, in order to deliver the seed to receiver side, the seed will be embedded within the ciphertext using the seed distributer

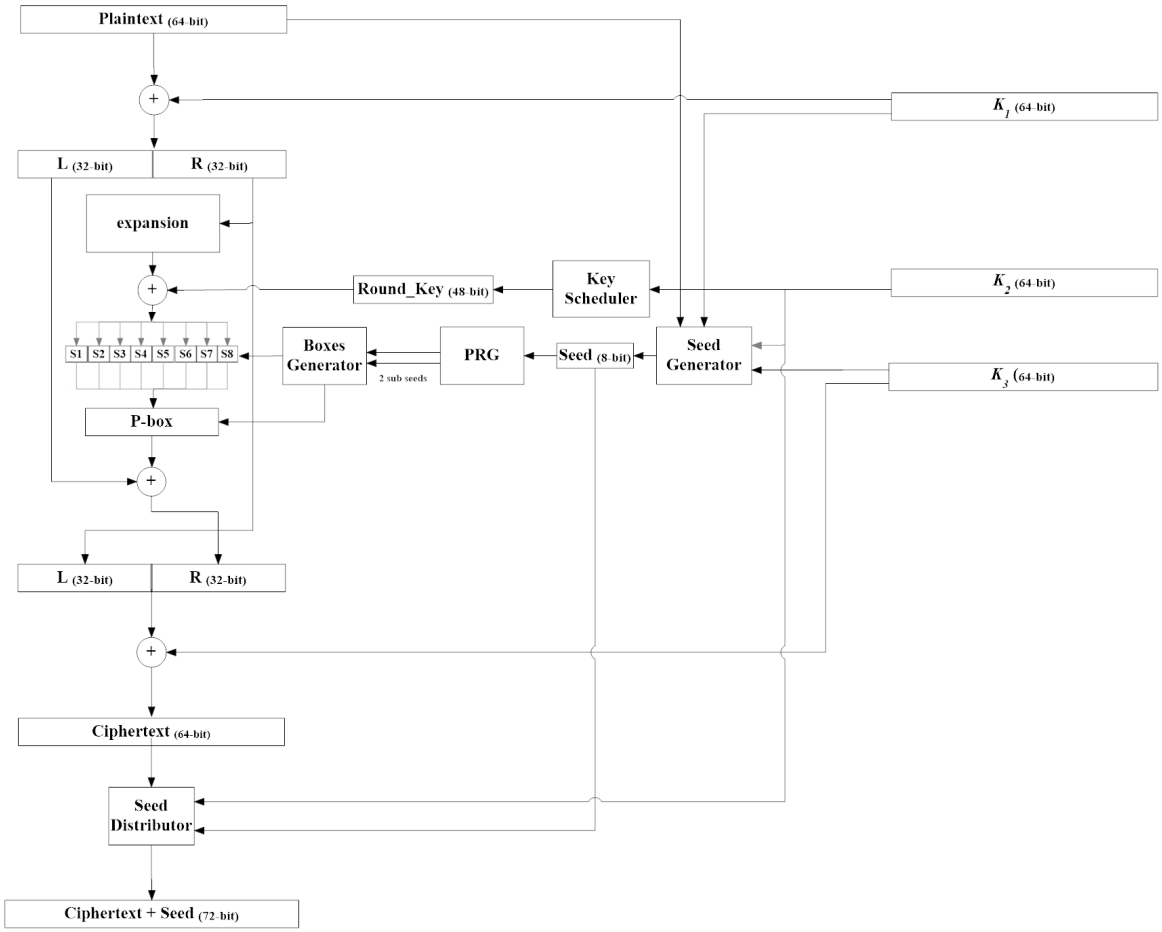

Fig. 1: Overview of the DDES encryption process

```
DDES encryption process
{
Input: keys K1, K2, K3, and plaintext (P), each of size 64 bits
Output: Ciphertext (C) of 72 bit
   seed = seedGenerator(P, K1, K2, K3)
   for ( round =1 to 16) {
         P ← P ⊕ K1
         Split P into two halves left and right, each of size 32 bits.
         rk ← keySchedular(K2)
         subseeds[2]← PRG(seed)
         temp ← right
         expanded_right ← Expand(right)
         expanded_right ← expanded_right ⊕ rk
         Generate S-boxes using subseed1 and P-box using subseed2
         right ← Permute(S-boxes(expanded_right)) ⊕ left
         left ← temp
         P ← join(left, right)
    }
    C ← P ⊕ K3
    C ← SeedDistributer(C, seed, K2)
}
```

Fig. 2: DDES encryption process

For the decryption process, the receiver feds the ciphertext into the seed filter, which extracts the seed from the ciphertext. The seed is used later to generate the S-boxes and P-box arrangements in reverse order. The right part of the text goes through the expansion algorithm and XORed with the reversed order of round keys. When the ciphertext reaches the substitution of S-boxes and P-box in permutations phase, the receiver will use the seed to generate the same boxes used for the encryption process.

DDES Components

Seed Generator

The seed generator takes the three keys (K_1, K_2, K_3) and the plaintext each of size 64 bits. The aim of this component is to obtain a seed of 8 bits that maps the plaintext and the three keys. The plaintext is first XORed with K_1 and K_3, then the resulted string is split into two parts and each part is fed to the original DES P-box. The reason for using the P-boxes is to create an avalanche effect, where changing few bits in the plaintext or in one of the keys, will result in a different seed and thus a different boxes arrangements during the encryption process. After that the resulting string is XORed with K_2 resulting in a string of 8 bytes binary string named result. To reduce this to one byte only to generate the needed seed, K_2 is used as follows: Divide K_2 into 8 groups from left right, each contains 6 bits. The decimal value of each group will determine the location of the corresponding bit in the seed. For example, let the first group of bits contains the decimal value 10, then the first bit in the seed is the 10th bit in the output. The pseudocode for the seed generator is illustrated in Fig. 3.

Pseudorandom Generator

The pseudorandom generator in DDES receives a seed of 8 bits as an input to generate two subseeds. The subseeds are fed into the boxes generator to obtain a secure combination from the S-boxes and the P-box arrangements for each round.

Boxes Generator

The boxes generator takes as input two subseeds of 8 bits each. One of the subseeds is used for the purpose of generating different permutations of secured S-boxes in each round. The P-box is generated using the other subseed and the S-boxes arrangement. Having different combinations of S-boxes and P-box arrangements for each round is to prevent linear cryptanalysis attacks as Biham and Biryukov (1995) shown. This enhancement is based on the boxes generator component that provides a secure S-boxes and P-box arrangements in every encryption round as illustrated in in Fig. 4. For each round, the boxes generator receives two new subseeds generated from the PRG, which will determine the selection of the S-boxes and P-box secure arrangements. The secure arrangements are stored in a lookup table. This secure arrangement is linked to another lookup table that contains the secure compatible P-box arrangements. These arrangements are proven to be secure based on the work Brown and Seberry (1990) and the work of Biham and Biryukov (1995). Afterwards,

based on the arrangements of the S-boxes and the P box originated from the secured tables, the boxes constructor part, will generate the S-boxes and P-box.

Seed Distributer

The seed distributer is used to insert the seed inside the ciphertext. It takes as input the ciphertext, the second key and the seed, to generate a ciphertext with the seed embedded within it. The embedded seed will be used in the decryption process since the receiver needs the same seed used by the sender. This seed will be given to the PRG at the receiver side. Figure 5 illustrates the pseudocode for the seed distributer. In order for the decryption to be possible, the sender and the receiver must have the same seed to be able to generate the same combinations of the S-boxes and P-box. In order to make the process harder for the attacker, we embed the seed generated using the three keys within the ciphertext which is mapped using the second key. On the receiver side, the seed filter will extract the seed from the ciphertext.

Seed Filter

The aim of this component is to extract the seed from the ciphertext, in order to perform the decryption process. In fact, this component performs the reverse operations that have been accomplished by the seed distributor. It takes as an input ciphertext with the seed embedded within it and the second key. The output will be the ciphertext and the seed which is fed into PRG to generate the sequence of sub-seeds in a reverse order.

```
SeedGenerator(K1, K2, K3, P)
{
Input: keys K1, K2, K3, and plaintext (P), each of size 64 bit
Output : Seed (8 bits )

        str1 ← P ⊕ K1 ⊕ K3
        str2 ← permute(str1)
        result ← str2 ⊕ K2
        seed ← reduce(result, K2) // reduces the result from 64 bits to 8 bits
}
```

Fig. 3: Pseudocode for the seed generator

```
BoxGenerator(subseeds)
{
Input: two subseeds
Output: index to the secure S-boxes arrangement and the compatible P-box arrangement
        S-boxes arrangement ← lookup(S-boxes table, subseed 1)
        P-box arrangement← lookup(P-box compatable table, subseed 2, S-boxes arrangement)
        Generate S-boxes and P-box using boxesConstructor
}
```

Fig. 4: Pseudocode for the boxes generator

```
SeedDistributor(C, seed, K2)
{
Input: ciphertext(C) of 64 bits, key K2 of 64 bits and seed of 8 bits
Output: ciphertext of size 72 bits with seed embedded in it

    divide K2 into 8 groups (g1,g2, ... , gn) each of size 8. Number these groups from 1 to 8
    for ( i =1 to 8) {
        li ← convert the value of gi into decimal
        insert the i'th bit of the seed in the location li within the ciphertext
    }
}
```

Fig. 5: Pseudocode for the seed distributer

HDES Overview

HDES uses a random seed derived from the plaintext. For each round in the encryption process, the seed is used to produce arrangements of secured S-boxes that are selected from a secured pool, which has been verified in (Biham and Biryukov, 1995; Brown and Seberry, 1990). In order to emphasis on the dynamic generation of the S-boxes, the plaintext is fed to a hashing component to produce a unique fingerprint that is used to generate the required seed. For each round in the encryption process a round seed is produced based on the generated seed. After completing the encryption process rounds, the seed will be embedded within the generated ciphertext, which will be used in decryption on the other side.

The main differences between DDES and HDES can be summarized as follows. DDES generates secure combinations of both S-boxes and P-box arrangements during each round, while HDES generates only secure arrangements of S-boxes. Furthermore, although some of the components of DDES and HDES hold the same title, their internal processes are different. For example the seed in DDES is generated based on the three keys and the plaintext. In HDES the seed is generated based on the key and the hashed plaintext.

The HDES components are described as follows: Hashing component, pseudorandom generator, seed generator, S-boxes generator, seed insertion and seed filter. The hashing component is used to produce a fingerprint of the plaintext. The seed generator takes the generated text from the hashing component and XOR it with the encryption key to produce the seed. The Pseudorandom Generator (PRG) in its turn produces round seeds for each round in the encryption process. The S-boxes generator takes the round seed to generate S-boxes arrangements for each round. The seed insertion component is used to produce a ciphertext with the seed embedded within it. The filter component is used during the decryption process to extract the seed from the ciphertext in order to perform the decryption. A detailed description of these component is provided in section 5.

The following steps provide illustration of the encryption process in HDES in addition to the illustration depicted in Fig. 6. A pseudocode that helps in understanding the operation of HDES is given in Fig. 7:

1. The plaintext is fed into the hashing component where it goes through the hash function. The output from the hash functions is XORed with the encryption key
2. The text resulted from the hashing component is used by the seed generator to generate the seed
3. The PRG uses the seed to produce a series of round seeds, which are used in producing the secured arrangements of the S-boxes generated through the S-boxes generator
4. The plaintext is split into two parts (L and R), each of size 32 bits
5. The key scheduler generates a round key, which will be XORed with the expanded R of size 48 bits. The resulting text is fed into the S-boxes and then permuted. The resulting 32 bits text is then XORed with L
6. A new text is produced of size 64 bits by joining the text from step 5 with R. However, R will become the left part
7. The generated text after completing the 16 rounds represents the produced ciphertext
8. Finally, the seed insertion component embeds the seed within the ciphertext

HDES Components

Hashing Component

The idea of using the hashing component is to create a fingerprint of the plaintext. The main idea behind this is to produce a unique string corresponding to every plaintext at the initial round. The hash function takes as input 64 bits representing the plaintext. The selection of the hash function is governed by a number of factors and constraints. For instance, the hash value produced by the hash function should not be the same for different plaintexts and the hashing result should provide low risk of collision. However, hash functions with low degree of collision can be considered acceptable. Another factor that may affect the selection of the hashing algorithm is the processing time, which is an essential factor that may influence the performance of the proposed HDES. For the design purpose of HDES, the selected hash function should be fast and simple to reduce the overhead resulted from using the hash function. Studies from the literature (Szydlo and Yin, 2006; Ilya, 2005) show that short hash functions are usually faster than long ones.

The processing speed of various cryptographic hash functions such as the SHA family, the MD family and many others has been investigated and analysed in (Knopf, 2007; Gauravaram, 2007; Preneel, 2003). Many factors were considered in the analysis including compiler platform, hardware environment and round functions, which may affect the processing time of the hash function. The analysis has shown that SHA-1 is better than MD5 in terms of processing speed and similar to MD4. It has also revealed that SHA-1 is better than both SHA-2 and SHA-256.

In this study, HDES selects SHA-1 as a hash function. This selection considers the trade-offs between the hashing speed and collision resistance of the hash function (Szydlo and Yin, 2006). For the design purposes of HDES, the speed of the hash function is considered more important than the collision resistance, since the hashing component is used as an inner component and that is used only to generate an input for the seed generator, which in turn will XOR the hashed text with the

key to produce the seed. HDES will then perform the 16 rounds of the encryption process, which means that the frequency of using the hash function is very low and hence the possibility of the collision to occur is low.

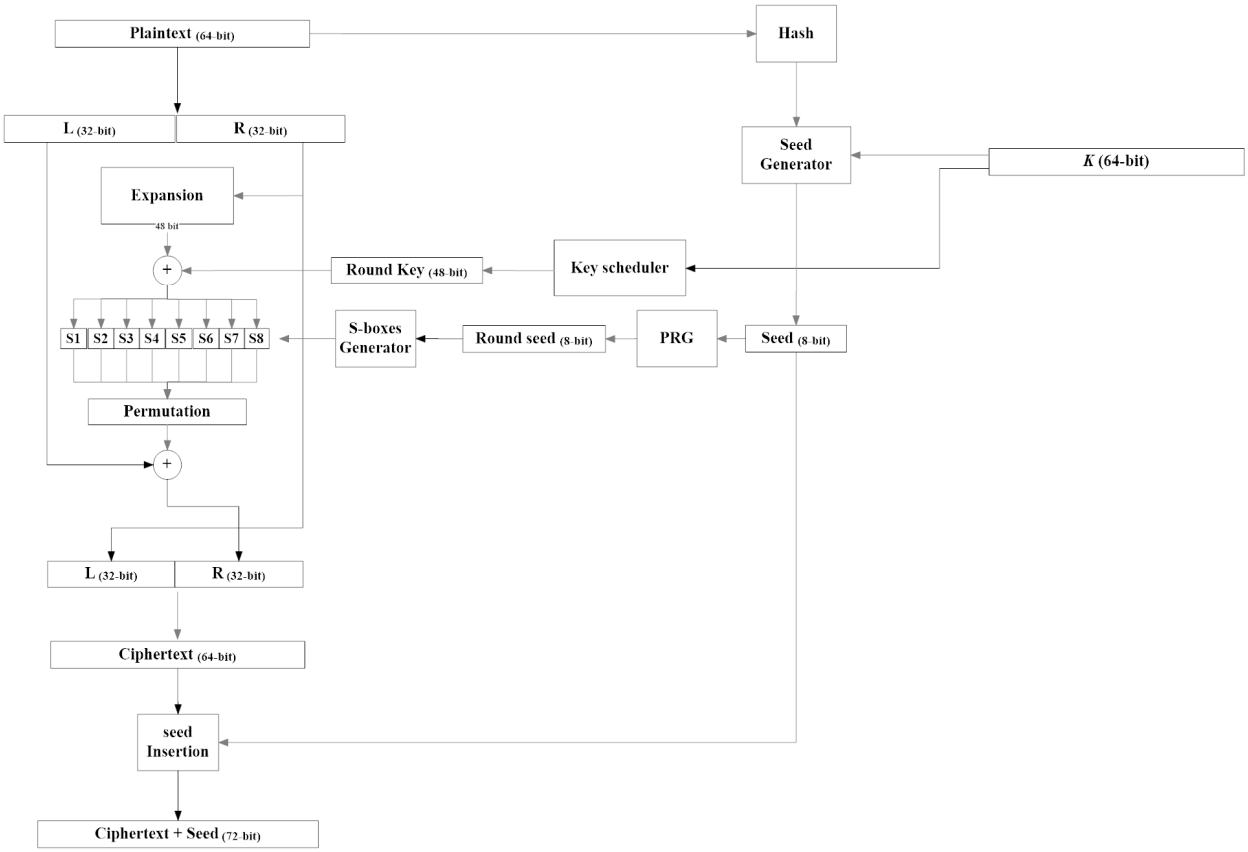

Fig. 6: Proposed HDES structure

```
HDES encryption process
{
Input: key K and plaintext (P), each of size 64 bits
Output: Ciphertext (C) of 72 bit
   HT ← Hash(P)
   seed = Seedgenerator(HT, K)
   for ( round =1 to 16) {
         Split P into two halves left and right, each of size 32 bits.
         temp ← right
         expanded_right ← Expand(right)
         rk ← keySchedular(K)
         round_seed← PRG(seed)
         expanded_right ← expanded_right ⊕ rk
         Generate S-boxes using the round_seed
         right ← Permute(S-boxes(expanded_right)) ⊕ left
         left ← temp
         P ← join(left, right)
    }
    C ← P
    C ← SeedInsertion(C, seed)
}
```

Fig. 7: HDES encryption process

Pseudorandom Generator

The pseudorandom generator is used to produce a series of random numbers based on a seed input of size 8 bits. These random numbers are fed into the S-boxes generator to generate different arrangements of S-boxes for each round.

Seed Generator

The seed generator takes the 64 bits hashed plaintext and XOR it with the encryption key the resulted text will be used to create a seed of 8 bits. The main idea behind this is to get a random and unique string that has one to many relationship with the plaintext and the key so we can produce a unique seed to be fed into the PRG. Due to the characteristics of the hashing function, any alteration in the plaintext would produce a totally different seed that would be random due to the XOR operation. This comes from the interesting characteristics of the XOR operation, when a fixed distribution string is XORed with a uniform distribution string (i.e., random string), the resulting string would follow the uniform distribution. In order to reduce the resulted 64 bits into 8 bits, to create the needed seed, the 64 bits are divided into 8 groups from left to right. To allow simplicity in the reduction process while keeping the randomness, the initial bit of each group will correspond to a bit in the seed.

S-Boxes Generator

The S-boxes generator generates dynamically, for each round, secured permutations of S-boxes. The dynamic nature comes from the fact that the S-boxes generator takes, for encryption round, a round seed obtained from the PRG, which is used to select the secure arrangement of S-boxes. The secure arrangements are stored in a lookup table where each entry represent a secure arrangement of S-boxes. These arrangements are proven to be secure based on the work Brown and Seberry (1990) and the work of Biham and Biryukov (1995).

Seed Insertion

The seed insertion plays an important role in the decryption process. Its role is to embed the seed in the generated ciphertext in order to allow its decryption. The seed is used to generate the same S-boxes used by the sender in encryption process, so that the receiver can decrypt the received ciphertexts. The seed bits are distributed within the ciphertext. To allow unpredictable distribution of the seed bits, the encryption key bits are used to insert the seed within the ciphertext. The insertion process is based on dividing the encryption key into eight groups, each of size bits. The insertion location within the ciphertext for a given bit in the seed is associated with the decimal value of the corresponding group in the key.

Filter

The filter component is used to extract the seed embedded within the ciphertext in the decryption process. It takes as input the key and the ciphertext to extract the seed bits. The seed will be given to the PRG to produce the same sequence of sub-seeds but in reverse order.

Evaluation

Encryption algorithms should be able to produce unpredictable random string of ciphertext. However, in many cases, the randomness of the ciphertext has been analysed to break the system. For example, the poor randomness quality of the digital signature algorithm in the third version of the Sony PlayStation allowed the attackers to recover the private key used in the signing process (Lee, 2013). In best case scenarios, the ciphertext should be unpredictable and the probability of zeros and ones are equal to 50%. Therefore, we have examined the quality of ciphertext randomness generated form DDES, HDES, DESX, 3DES and DES. The evaluation is conducted using four measures that designed based on guidance of the NIST test statistical suite of cryptographic applications (Rukhin *et al*., 2010): Chi-square test, cipher data difference, hamming distance and encryption time. The five algorithms have been challenged to identical scenarios to encrypt various files with sizes from 45 KB to 1 MB that have data entropies ranging from 0.80 to 0.99. The platform used in the evaluation is Xeon E3-1225 running Windows server 2008, with processor speed of 3.2GHz and 8 GB of RAM.

Chi-Square Test

The chi-square test is a statistical test that is used to compare an observed data with what is expected to obtain from a random generation system, based on pre-defined hypothesis. The hypothesis usually initiated early in the experiment, based on the evaluator understanding and beliefs about the expected statistical outcome of a specific experiment. The chi-square is used to assess the likelihood that the hypothesis is true based on the Equation 1 (Lee, 2013). Where O is the observed data and E is the expected data. The chi-square is usually useful to see if there is a difference between two or more groups of data:

$$x^2 = \sum_i \frac{(O_i - E_i)^2}{E_i} \tag{1}$$

In this measures, the chi-square of the ciphertext generated from the DDES, HDES, DESX, 3DES and DES has been computed to understand the randomness of the ciphertext based on expected outcome using the null hypothesis. In the null hypothesis there will be no

difference between the observed and the expected data. The cipher data is expected to have 50% of zeroes and 50% of ones, because in a truly random function the ratio between zeros and ones are equal. Table 1 illustrates the average chi-square test probability based on the chi-square values and their corresponding p-values with degree of freedom of one. The chi-square test probability has been computed after encrypting files with data entropies ranging from 80% to 99%. Based on (Fahmy *et al*., 2005; Lee, 2013) if the probability is greater than 99% or less than 1%, the ciphertext is almost certainly not random. If the probability is between 99% and 95% the ciphertext is randomly suspect. If the probability between 90% and 95% indicate the ciphertext is almost random. As Table 1 illustrates, DDES shows the best level of randomness compared to its counterparts with an average probability of 93.7%, which indicates that the ciphertext obtained from DDES is almost random. Whereas HDES average probability is 97.7% which indicates that its randomness fall under the randomly suspect level, which is also the case of DES, DESX and 3DES, although 3DES shows better performance than that of HDES.

Hamming Distance

The hamming distance is an important measure to capture the diffusion effect for the encryption algorithms. The higher degree of diffusion is always preferable for the encryption algorithm, which indicates that each plaintext bit or key bit change, affects many bits of the ciphertext bits. The hamming distance is the number of bits which needs to be changed to turn one string into the other. The hamming distance between two ciphertexts is calculated using Equation 2, where X is the first ciphertext which consists of several bits say $x1$, $x2$, …, xn, Y is the second ciphertext which consists of several bits say $y1$, $y2$, …, yn.The hamming distance is calculated by flipping a number of bits in the plaintext:

$$HD(X,Y) = \left|\{i|x_i \neq y_i\}\right| \quad (2)$$

The hamming distance is computed between the generated ciphertexts corresponding to the plaintext with flipped bits. This is done by flipping 1 bit, 2 bits, 3 bits and 4 bits as shown in Fig. 8. This process is performed on each of DES, DESX, 3DES, HDES and DDES. As the figure illustrates, DDES has higher hamming distance than its counterparts in most cases. Besides, the HDES hamming distances are also higher than the hamming distances of DES, DESX and 3DES.

This means that the diffusion effect for DDES and HDES is higher than that of DES, DESX and 3DES.However, the reason for this is the ability of DDES and HDES to change the internal permutations each round.

Table 1: The chi-square probability values for the encryption algorithms

Encryption algorithm	Chi-square test (%)
DES	98.3
DESX	98.1
3DES	95.9
DDES	93.7
HDES	97.7

Cipher Data Difference

The cipher data difference is proposed to find the randomness of the data encryption algorithms. It is computed by finding the absolute difference of ones and zeros in the plaintext and the absolute difference of ones and zeros in the ciphertext. The lower the value of this metric indicates higher level of randomness. Figure 9 shows the cipher difference of DDES, HDES, DES, DESX and 3DES. Different data entropies are used to evaluate the cipher difference of the selected encryption algorithms. The data entropies are represented in the x-axis and the cipher data difference is represented the y-axis.

The points in Fig. 9 present an average of 30 runs that has confidence interval of 95%. As the figure shows, DDES demonstrates lower cipher difference in most of the cases followed by HDES. This indicates that the randomness obtained from DDES and HDES outperforms their counterparts. The data encryption algorithms show close performance for files with data entropies between 0.80 and 0.84, where 3DES shows the lowest cipher data difference in this interval. However, as the data entropy increase, DDES outperforms the other algorithms. It is also noted that HDES show better randomness results than DES, DESX and 3DES in the interval of 0.90 and 0.99 of data entropies. This means, in the case of low data entropy the algorithms have analogous outcome of randomness, however DDES and HDES exhibit better randomness for the higher values of data entropies.

Encryption Time

Figure 10 shows the time required for the encryption process for DDES, HDES, DES, DESX and 3DES. As the figure illustrates, 3DES has the highest encryption processing time, this is due to the fact that 3DES repeats the encryption process of the original DES but using three different encryption keys. Moreover, the cipher data difference achieved by 3DES was higher than DDES and HDES, which means that 3DES requires more time of processing while providing less randomness to the cipher. On the other hand, the encryption time for DDES is higher than DES, DESX and HDES. This is due to the extra processing to enable DDES to use secure combinations of the S-boxes and the P-box on each round of the encryption process that results on a lower cipher data difference than its counterparts as shown in Fig. 9.

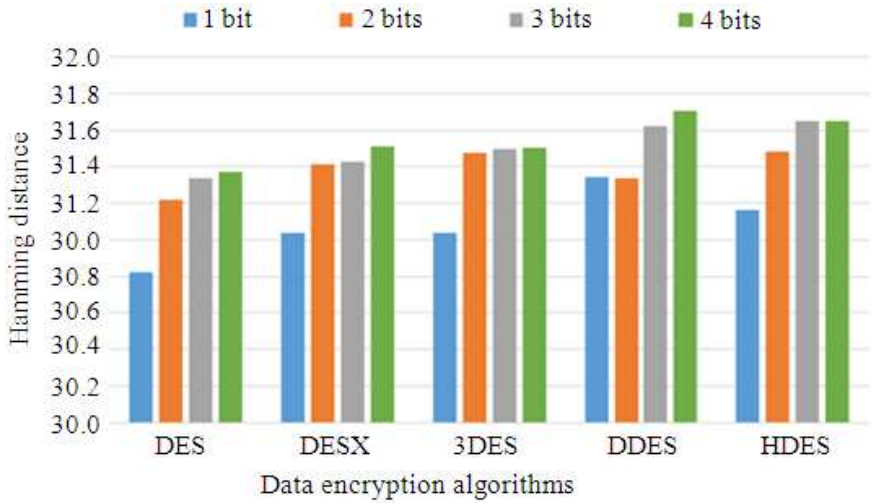

Fig. 8: Hamming distance of the data encryption algorithms

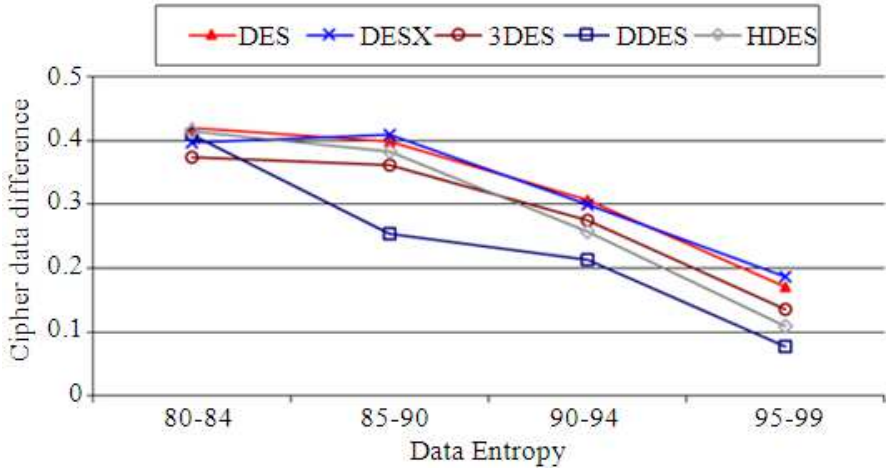

Fig. 9: Cipher data difference

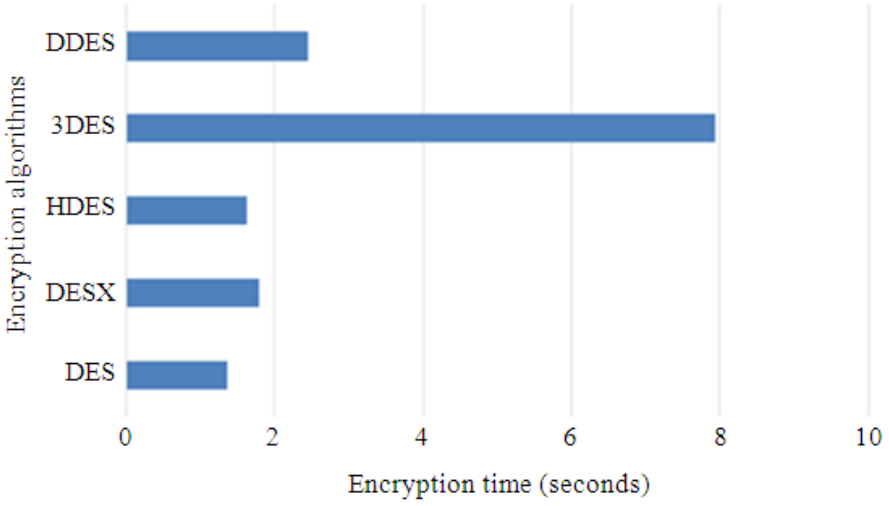

Fig. 10: Encryption processing time

Conclusion

This paper proposes two variants of DES named DDES and HDES. The objective of DDES and HDES is to overcome the flaws in the original DES, by redesigning and combining several techniques and components to enhance the original DES. The main characteristics of DDES is its larger key size and its internal components based on secure combinations of S-boxes and P-box arrangements. HDES, on the other hand, uses a hash function to obtain a unique fingerprint for each plaintext block that is used to select secure S-boxes only for each round in the encryption process. The rationale behind the two variants is to meet the trade-off between robustness of the ciphertext and processing speed. The two variants have been evaluated extensively using a number of metrics: Chi-square test, cipher data difference, hamming distance and processing time and their performance has been compared against DES, DESX and 3DES. DDES and HDES provide options for the security professionals to deploy the required security level and processing time based on the user application context. This have been confirmed by the conducted experiments that show DDES with higher degree of randomness in terms of chi-square test, cipher data difference and hamming distance while having a relatively acceptable encryption time. HDES, on the other hand, shows acceptable randomness levels. Although it has lower levels of randomness than that of DDES, its encryption time outperforms DDES. As part of our future work, we plan to investigate other randomness measures including linear Complexity test, discrete Fourier transform (spectral) test and approximate entropy test of DDES and HDES against AES and blowfish.

Author's Contributions

Malik Qasaimeh: Contributed to research design, algorithms implementation, randomness analysis, result discussion and analysis. In addition to writing, editing and reviewing of the manuscript.

Raad S. Al-Qassas: Contributed to algorithms design, implementation, discussion and analysis of the results. In addition to writing, editing and reviewing of the manuscript.

Ethics

The corresponding author confirms that all of the other authors have read and approved the the manuscript associated with no ethical issues involved.

References

Alnoury, H.A., M. Qasaimeh and R.S. Al-Qassas, 2016. Improved DES with dynamic S-boxes and P-box arrangements. J. Next Generat. Inform. Technol., 7: 14-26.

Al-Qassas, R.S., M. Qasaimeh and H. Al-Nouri. 2016. A fingerprint featured data encryption algorithm. Proceedings of the 7th International Conference on Information and Communication Systems, Apr. 5-7, IEEE Xplore Press, Jordan, pp: 227-232. DOI: 10.1109/IACS.2016.7476116

Biham, E. and A. Biryukov. 1995. How to strengthen DES using existing hardware. In: Advances in Cryptology — ASIACRYPT'94, Pieprzyk, J. and R. Safavi-Naini (Eds.), Springer, Berlin, Heidelberg, ISBN-10: 978-3-540-59339-3, pp: 398-412.

Biham, E. and A. Shamir, 1991. Differential cryptanalysis of DES-like cryptosystems. J. Cryptol., 4: 3-72. DOI: 10.1007/BF00630563

Biryukov, A. and D. Wagner, 2000. Advanced Slide Attacks. In: Advances in Cryptology — EUROCRYPT 2000, Preneel, B. (Ed.), Springer, Berlin, Heidelberg, ISBN-10: 978-3-540-67517-4, pp: 589-606.

Brown, L. and J. Seberry, 1990. On the Design of Permutation P in des Type Cryptosystems. In: Advances in Cryptology — EUROCRYPT '89, Quisquater, J.J. and J. Vandewalle (Eds.), Springer, Berlin, Heidelberg, ISBN-10: 978-3-540-53433-4, pp: 696-705.

De Cannière, C., 2011. Triple DES. In: Encyclopedia of Cryptography and Security, Van Tilborg, H.C.A. and S. Jajodia (Eds.), Springer US, Boston, MA.

Dong, X., R. Li, H. He, W. Zhou and Z. Xue *et al*., 2015. Secure sensitive data sharing on a big data platform. Tsinghua Sci. Technol., 20: 72-80. DOI: 10.1109/TST.2015.7040516

Fahmy, A., M. Shaarawy, K. El-Hadad, G. Salama and K. Hassanain, 2005. A proposal for a key-dependent AES. Proceedings of the 3rd International Conference on Sciences of Electronic, Technologies of Information and Telecommunications, Mar. 27-31, IEEE Xplore Press, Tunisia.

Friedewald, M., D. Wright, S. Gutwirth and E. Mordini, 2010. Privacy, data protection and emerging sciences and technologies: Towards a common framework. Innovation Eur. J Soc. Sci. Res., 23: 61-67. DOI: 10.1080/13511611003791182

Gauravaram, P., 2007. Cryptographic hash functions: Cryptanalysis, design and applications. Published dissertation in partial fulfillment of the requirements for the degree of Doctor of Philosophy Thesis, Queensland University of Technology, Queensland, Australia.

Ilya, M., 2005. Hash functions: Theory, attacks and applications. Proceedings of Microsoft Research, Silicon Valley Campus, (SVC'05), pp: 1-22.

Kilian, J. and P. Rogaway, 2001. How to protect DES against exhaustive key search (an analysis of DESX). J. Cryptol., 14: 17-35. DOI: 10.1007/s001450010015

Kim, K., 1991. Construction of DES-like S-boxes Based on Boolean Functions Satisfying the SAC. In: Advances in Cryptology — ASIACRYPT '91, Imai, H., R.L. Rivest and T. Matsumoto (Eds.), Springer, Berlin, Heidelberg, ISBN-10: 978-3-540-57332-6, pp: 59-72.

Knopf, C., 2007. Cryptographic Hash Functions. Leibniz Universität Hannover, Germany.

Kocher, P., J. Jaffe, B. Jun and P. Rohatgi, 2011. Introduction to differential power analysis. J. Cryptographic Eng., 1: 5-27. DOI: 10.1007/s13389-011-0006-y

Kumar, S., C. Paar, J. Pelzl, G. Pfeiffer and A. Rupp *et al*., 2006. How to break DES for euro 8,980. Proceedings of the 2nd Workshop on Special-Purpose Hardware for Attacking Cryptographic Systems, Apr. 3-4, Ruhr University Bochum, Germany.

Lee, M., 2013. Investigating modern cryptography. Published dissertation in partial fulfillment of the requirements for the degree of Master of Science Thesis, Universty of Windsor, Ontario, Canada.

Matsui, M. and A. Yamagishi, 1993. A New Method for Known Plaintext Attack of FEAL Cipher. In: Advances in Cryptology—EUROCRYPT'92, Rueppel, R.A. (Ed.), Berlin, Heidelberg, ISBN-10: 978-3-540-56413-3, pp: 81-91.

Matsui, M., 1993. Linear Cryptanalysis Method for DES Cipher. In: Advances in Cryptology—EUROCRYPT'93, Helleseth, T. (Ed.), Springer, Berlin, Heidelberg, ISBN-10: 978-3-540-57600-6, pp: 386-397.

Matsui, M., 1994. On Correlation Between the Order of S-Boxes and the Strength of DES. In: Advances in Cryptology—EUROCRYPT'94, De Santis, A. (Ed.), Springer, ISBN-10: 978-3-540-60176-0, pp: 366-375.

NIST, 2012. Recommendation for the Triple Data Encryption Algorithm (TDEA) block cipher. NIST Special Publication 800-67 Revision 1, National Institute of Standards and Technology, United States.

Preneel, B., 2003. Analysis and design of cryptographic hash functions. Published dissertation in partial fulfillment of the requirements for the degree of Doctor of Philosophy Thesis, Katholieke Universiteit Leuven, Belgium.

Rogaway, P., 1996. The security of DESX. RSA Laboratories Cryptobytes, 2: 8-11.

Rukhin, A., J. Soto, J. Nechvatal, M. Smid and E. Barker *et al*., 2010. Statistical test suite for random and pseudorandom number generators for cryptographic applications. Special Publication 800-22: Revision 1a. National Institute of Standards and Technology, United States.

Schneier, B., 1996. Applied Cryptography: Protocols, Algorithms and Source Code in C. 2nd Edn., John Wiley & Sons, Inc., Indianapolis, IN, ISBN-10: 0471128457, pp: 758.

Shaikh, R. and M. Sasikumar, 2015. Data classification for achieving security in cloud computing. Procedia Comput. Sci, 45: 493-498. DOI: 10.1016/j.procs.2015.03.087

Sison, A.M., B.T. Tanguilig III, B.D. Gerardo and Y. Byun, 2012. Implementation of Improved DES Algorithm in Securing Smart Card Data. In: Computer Applications for Software Engineering, Disaster Recovery and Business Continuity, Communications in Computer and Information Science, Kim, T., C. Ramos, H. Kim, A. Kiumi and S. Mohammed *et al*. (Eds.), Springer, Berlin, Heidelberg, ISBN-10: 978-3-642-35266-9, pp: 252-263.

Szydlo, M. and Y.L. Yin, 2006. Collision-Resistant Usage of MD5 and SHA-1 Via Message Preprocessing. In: Topics in Cryptology–CT-RSA 2006, Pointcheval D. (Ed), Springer, Berlin, Heidelberg, ISBN-10: 978-3-540-31033-4, pp: 99-114.

Tawalbeh, L.A., N.S. Darwazeh, R.S. Al-Qassas and F. Aldosari, 2015. A secure cloud computing model based on data classification. Procedia Comput. Sci., 52: 1153-1158. DOI: 10.1016/j.procs.2015.05.150

Van Tilborg, H.C.A. and S. Jajodia, 2011. Encyclopedia of Cryptography and Security. 2nd Edn., Springer US, ISBN-10: 978-1-4419-5907-2, pp: 1416.

Verdult, R., 2015. The (in) security of proprietary cryptography. Published dissertation in partial fulfillment of the requirements for the degree of Doctor of Philosophy Thesis, Radboud University Nijmegen, Netherland.

Verma, J. and S. Prasad, 2009. Security Enhancement in Data Encryption Standard. In: Information Systems, Technology and Management, Prasad, S.K., S. Routray, R. Khurana and S. Sahni (Eds.), Springer, Berlin, Heidelberg, ISBN-10: 978-3-642-00405-6, pp: 325-334.

Yun-peng, Z., L. Wei, C. Shui-ping, Z. Zheng-jun and N. Xuan *et al*., 2009. Digital image encryption algorithm based on chaos and improved DES. Proceedings of the International Conference on Systems, Man and Cybernetics, Oct. 11-14, San Antonio, TX, pp: 474-479. DOI: 10.1109/ICSMC.2009.5346839

Zhuang, Z., J. Chen and H. Zhang, 2014. A countermeasure for DES with both rotating masks and secured S-boxes. Proceedings of the 10th International Conference on Computational Intelligence and Security, Nov. 15-16, IEEE Computer Society, Washington, pp: 410-414. DOI: 10.1109/CIS.2014.43

Zodpe, H.D., P.W. Wani and R.R. Mehta, 2012. Design and implementation of algorithm for des cryptanalysis. Proceedings of the 12th International Conference on Hybrid Intelligent Systems, Dec. 4-7, IEEE Xplore Press, Pune, pp: 278-282. DOI: 10.1109/HIS.2012.6421347