http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 4 (4): 307-325, 2005
ISSN 1812-5638

Theory and Implementation of Data Encryption Standard: A Review

Kefa Rabah
Department of Physics, Eastern Mediterranean University, Gazimagusa,
North Cyprus, via Mersin 10, Turkey

Abstract: Over the last decade the world has seen an astounding growth of information technology that has resulted in significant advances in cryptography to protect the integrity and confidentiality of data. The most affected is the financial sector and the e-commerce over the Internet, which requires secure data transfer, or handling during any transaction of business. Secrecy is the heart of cryptography. Encryption is a practical means to achieve information secrecy. In this aspect DES (Data Encryption Standard)- A symmetric key cryptography and its variant triple DES, has over the last three decades played major role in securing data in this sector of the economy and within other governmental and private sector security agencies. In this study the theory and implementation of DES and its suitability in securing data in future will be described. Comparison with other symmetric key crypto-algorithm will also be considered.

Key words: Cryptography, data security, secure communication, DES, 3DES, AES, RSA, EVM, ATM

INTRODUCTION

Today it is widely acknowledged that security issues which are already of highest priority will continue to play a central role in the design of future IT systems. The range and areas of applications with security needs appear to be almost endless: Internet communications; constrained wireless devices (e.g., cell phones and wireless LAN), communication between cars, e-commerce, e-Banking, copy right protection of digital media (e.g., music and videos), electronic bar codes on consumer products (RFID), electronic stamps, etc.[1-6]. Therefore, design and implementations of security solutions in future will increasingly be quite a challenging undertaking since an attacker will attempt to attack the weakest link in the security systems. This could be via many available options, e.g. breaking the underlying crypto-algorithm and which in most cases is harder option to be attempted only as a last resort. Instead the eavesdropper might opt to exploit other weaknesses such as: recovering a key by observing the power consumption or electromagnetic radiation of the crypto-devices; finding vulnerabilities in the security protocols or simply revert to stealing the key: through clever interactive social engineering with those trusted to safeguard the keys. In most cases, however, the crypto-algorithms are always the most important core tool in security applications. Hence, the cryptographic algorithms have to be carefully designed, selected and implemented in order to avoid the core cryptography becoming the weakest link in any security solution[7-10].

Modern crypto-techniques are mathematical transformations (algorithms), which treat messages as numbers or algebraic elements in a space and transform them between a region of meaningful messages or cleartext and a region of unintelligible messages or ciphertext. In order to restore information, an encryption transformation must be reversible and the reversing transformation is called decryption. Conventional, encryption and decryption algorithms are parameterized by cryptographic keys. An encryption algorithm and a decryption algorithm plus the description on the format of messages and keys, form a cryptographic systems or a cryptosystem[11].

Currently, there are two popular kinds of cryptographic protocols: symmetric-key and asymmetric-key protocols. In the symmetric-key protocols, a common key (the secret-key or private key) is used by both communicating partners to encrypt and decrypt messages to be communicated. Symmetric key crypto-algorithm is quite old, with early forms known since 1900 BC: these include crypto-codes and cipher techniques such as the Caesar Cipher, Hill Cipher, Vernam cipher etc. This ciphers were mostly either manually coded (Caesar Cipher) or the early 20th century crypto-schemes that were electro-mechanical in nature (including the WWI and WWII ciphers e.g., the German Enigma) that are not secure against state-of-the-art cryptanalytical attacks[5,12]. However, it wasn't until the early 1970s, with the advent of electro-digital circuits, that truly strong ciphers with more sophisticated private-key cryptosystems were

developed[4]. The development of cheap digital hardware freed the crypto-designers from the design limitations of mechanical electro-computing and brought the cost of high grade cryptographic devices down to where they could be used in such commercial applications as remote cash dispenser (ATMs) and computer terminals. Among these are: DES, RC5, IDEA and AES[11]. The term private-key is used because this technique implies that both the sender and the receiver of the message have to agree on common session key that must be kept private prior to the commencement of any communication between them. Private-key cryptography makes use of the same key on both the sending and the receiving end and is therefore also referred to as symmetric cryptography. The main functions of these algorithms are: encryption, authentication of data through Message Authentication Codes (MACs) and hashing. There are two types of symmetric key algorithms which are commonly distinguished: block ciphers and stream ciphers[4]. Stream ciphers, in contrast to block ciphers, operate on a single bit of plaintext at a time. They consist of transformation which changes for every incoming plaintext bit. Stream ciphers tend to have a higher data throughput (i.e., they tend to be faster) and require less resources than block ciphers and are thus sometimes preferred in securing low level resource constrained wireless devices like cell phone.

In secrete-key cryptography it is assumed that Bob and Alice both use some piece of information in their encrypting and decrypting algorithms that is not known to eavesdropper, Eve. Here, Bob and Alice could be military jets, e-business or just friends trying to have a private conversation. They can't stop Eve listening to their radio signals (or tapping their phone line, or whatever). One solution is for Alice and Bob to exchange a digital key, so they both know it, but it's otherwise secret. The symmetric-key cryptosystems is known to provide high-speed key and communication but have the drawback that a common (or session) key must be established for each pair of participants. One such symmetric key crypto-algorithm that can be used for such task is DES (Data Encryption Standard)[13-15]. It is one of the most widely used modern conventional cryptosystem today and; the one-time pad which till today is the most secure private-key cryptosystem[2]. Although numerous other conventional encryption algorithms have been developed since the introduction of DES, it still remains the most important such algorithm because it is well understood and easy to use.

In symmetric cryptography, for both secrecy and authentication, the sender must transmit the key to the recipient via some secure and tamperproof channel, otherwise the recipient won't be able to decrypt the ciphertext. Hence, the major security issue in secret-key cryptography involves keeping the encryption-key secure and these requirements lead to the major shortcoming of symmetric-key cryptography. The process of exchanging the cryptographic key is referred to as key distribution and can be very difficult[7]. It is not difficult to see the impracticality of this approach among a large collection of people. The key is the secret to breaking the ciphertext; if there exists a really secure method of communicating the key, why isn't that method used to communicate the message in the first place?

To this effect, all classical symmetric encryption methods, suffer from the key distribution problem. The problem is that before a private communication can begin, another private transaction is necessary to distribute corresponding encryption and decryption keys to the sender and receiver, respectively. Typically, a trusted private courier is used to carry a key from the sender to the receiver. Such a practice is not feasible if an electronic mail system or secure data transfer on-the-fly is to be rapid and inexpensive.

The problem of key distribution was further complicated with the development of computer controlled communication networks (Internet) which promises effortless and inexpensive contact between people or computers on opposite side of the globe, replacing most of mail and many excursions with telecommunications. A further complication is in the business world, a private conversation between two parties with no prior acquaintance is a common occurrence, however, and it is unrealistic to expect initial business contacts to be postponed long enough for keys to be transmitted by some physical means. The cost and delay imposed by this key distribution problem is a major barrier to the transfer of business communications to large teleprocessing networks. This difficulty with symmetric ciphers and its associated key distribution, led to the development of public-key cryptosystems, which needs no private couriers; the keys can be distributed over the insecure communication network[16].

In the public-key protocols, there are two associated keys; one is kept private by the owner and used either for decryption (confidentiality) or encryption (signature) of messages. The other key is published in the public domain (public-key server) to be used for the reverse operation or decryption. Different public-key cryptographic systems are used to provide public-key security. Among these we can mention the RSA[17], Diffie-Hellman (DH)[18], Digital Signature Algorithm (DSA)[19,20], ElGamal cryptosystem[21,22] and recent years the Elliptic Curve Cryptography (ECC)[1,23]. These systems

provide these services by relying on the difficulty of different classical mathematical problems, hence provide the services in different ways. It is designed to be computationally intractable to calculate a private-key from its associated public-key; that is, it is believed that any attempt to compute it will fail even when up-to-date technology and equipment are used. Because of this feature, these cryptosystems are considered to be indispensable for secure communication and authentication over open (insecure) networks. Hence, are suitable for providing all the functionality needed in modern security protocols such as secure socket layer: SSL/TLS, which is the backbone Internet security allowing secure e-commerce transactions over the insecure network systems.

The public-key cryptosystems when used for data encryption are extremely computationally demanding and thereby slower than the symmetric ones but provide arbitrary high levels of security and do not require an initial private-key exchange between two communicating parties. However, many symmetric key schemes can encrypt 1000 times faster than public key cryptosystems but are, however, poor at providing non-repudiation and key establishment functionality[19].

In real applications, however, most practical protocols are hybrid protocols which incorporate both symmetric and public key schemes. The public-key algorithm is first used for establishing a common symmetric-key over insecure channel[18,24]. Then the symmetric system is used for secure communication with high throughput. Due to comparative slowness of the public-key algorithms, dedicated hardware is desirable for efficient implementation and operation of the cryptographic systems. This study will concern with symmetric key cryptosystems, DES, however, the possibility of integrating DES/3DES and AES with existing public key cryptosystem like RSA or ECC will also be considered.

CONVENTIONAL SYMMETRIC KEY CRYPTOGRAPHY

The backbone of symmetric-key crypto-algorithm relies on the mathematical abstraction involving permutation-substitution, diffusion and confusion and their integrated use to convert plaintext into ciphertext. According to Shannon a cryptosystem can be characterized with the following properties: which states that the ciphertext message space is the space of all possible messages while the cleartext (human readable) message space is a sparse region inside the message space, in which messages have a certain fairly simple statistical structure, i.e., they are meaningful; a (good) encryption algorithms is a mixing-transformation which distributes the meaningful messages from the sparse and meaningful region fairly uniformly over the entire messages space[25-27]. While, Kerkhoff[28] listed the requirements to design cryptosystems today referred to as Kerkhoff's principle. This principle states that the security of the system has to be based on the assumption that the enemy has full knowledge of the design and implementation details of the crypto-algorithm. The only missing information for the enemy is a short easily exchangeable random number sequence, the secret-key[2]. Without this secret-key, the enemy should not have the chance to even suspect that on an observed communication channel, hidden communication is taking place[3]. Since the adversary may obtain this information eventually, it is preferable not to rely on its secrecy when assessing cryptographic strength. Most of the modern crypto-software in use today meets this principle. Combining these two thinking gives rise to a good cryptosystems design and implementation capable of securing data in any given time-space.

To date computation has not been successful in relating two pieces of information if one of them merely looks random while in fact they are completely dependent on one another (for example, plaintext, ciphertext messages in many cryptosystems). As a result, modern cryptography has its security based on a so-called complexity-theoretic model. Security of such cryptosystems is conditional on various assumptions that certain problems are intractable. Here, intractable means that the widely available computational methods today cannot effectively handle these problems in a reasonable time frame[29].

If random variable follows the distribution and is independent, from any given information, then there is no way to relate a uniformly random variable to any other information by any means of computation[30]. This is exactly the security behind the only unconditionally (or information-theoretically) secure encryption scheme: one-time pad, that is, mixing a uniform random string (called key string) with a message string in a bit-by-bit fashion[2]. The need for independence between the key string and the message string requires the two strings to have the same length. Unfortunately, this poses an almost un-passable limitation for a popular and practical use of the one-time pad encryption scheme.

THE MECHANIC OF SYMMETRIC KEY CRYPTOSYSTEMS

There are two basic working principles of the classical ciphers: substitution and transposition are still

the most important kernel and technique in the construction of the modern symmetric encryption algorithms. The combination of substitution and transposition ciphers are implemented in one of the most important modern symmetric encryption algorithms, DES, which is the subject of this study and of course the same implementation principle also applies to other symmetric ciphers like RC5, AES etc[31-33]. Consider character-based substitution ciphers. Because the plaintext message space coincides with the alphabet, each message is a character in the alphabet and encryption is to substitute character-by-character of each plaintext character with a ciphertext character and the substitution is according to a secrete key. If a key is fixed for encrypting a long string then the same character in the plaintext message will be encrypted to a fixed character in the ciphertext messages. A further strengthening of the ciphertext message can be done via the use of polyalphabetic algorithm[4]. Polyalphabetic ciphers and transposition ciphers are stronger than simple substitution ciphers. However, if the key is short and the message is long, then various cryptanalysis techniques can be applied to break such ciphers. But one may note that, classical ciphers, even simple substitution ciphers can be secure in a very strong sense if the use of cryptographic keys follows certain conditions. With the proper key usages, simple substitution ciphers are today still widely used in cryptographic systems and protocols.

The concept of block cipher encryption originally developed by Feistel[34], is very commonly used for this purpose; it involves the use of a block or a group of bytes for encryption process instead of a single byte or character. Each block can be operated on by any combination of several processes. Typical block size of 64 bits is used. Security is obtained by having a one-to-one mapping between blocks of characters of plaintext and blocks of ciphertext of the same size-but the relationship between them is not easy to figure out. A block cipher operates on a plaintext block of n-bits to produce a ciphertext block of n-bits. There are 2^n possible different plaintext blocks and, for the encryption to be reversible (i.e., for decryption to be possible), each must produce a unique ciphertext block. Such a transformation is called reversible, or nonsingular. A 4-bit input produces one of 16 possible input states (from 0000 to 11111), which is mapped by the substitution cipher into a unique one of 16 possible output states, each of which is represented by 4-bit ciphertext.

An alternative and more popular method of implementing a substitution function is to use a construct referred to as a Symmetric (Substitution) Box, or an S-box. The S-box design is one of the most intense area of research in the field of symmetric block ciphers cryptography. The S-box function takes some bit or set of bits as input and provides some other bit or set of bits as output. It makes use of a replacement table to perform the conversion. These reference tables can map more than one input to the same output. In essence, change to the input vector to an S-box to result in random-looking changes to the output. The relationship should be nonlinear and difficult to approximate with linear functions. As a result of this truth, a hacker cannot take the output from an S-box and figure out which of the many inputs may have been used to generate the output. One of the obvious characteristics of S-box is its size. An 6x4 S-box has n input bits and m output bits. DES has 6x4 S-boxes, blowfish and CAST have 8x32 S-boxes[11]. The larger the S-boxes, the more difficult differential and linear cryptanalysis will be. On the other hand, the larger the dimension n, the (exponentially) larger the lookup table. Thus, for practical reasons, a limit of n equal to about 8 to 10 is usually imposed. An 6x4 S-box typically consists of 2^n rows and of m-bits each. The n-bits of input select one of the rows of the S-box and the m-bits in that row are the output. This type of substitution is not necessarily secure enough; the German Enigma is a complex substitution algorithm that was broken before the advent of computers[5,35,36].

A further secure implementation of the ciphertext can be achieved via use of permutation-transposition technique. Transposition involves exchange of two elements of an ordered list with all others staying the same. A transposition is therefore a permutation of two elements. For example, the swapping of 2 and 5 to take the list 123456 to 153426 is a transposition. Permutation technique on the other hand, involves rearrangement of the characters of a plaintext message to convert the message into an anagram that looks like a message with random characters. In real-practical application, for example, most messages consist of 7-bit ASCII characters or 8-bit in hexadecimal which we can represent with integer numbering (01234567) according to Table 1, to give permutation representation e.g., {7,5,4,6,3,0,2,1}. By scrambling the bits to create a random set of bits (using inverse of our permutation i.e., {5,7,6,4,2,1,3,0} you can get the desired encryption. For example, an 8-bit character, the byte/character 'G', which have position 71 into ASCII character set and have the binary representation '0100 0111' can be encrypted/decrypted as shown in Table 1. Permutation techniques are usually used in conjunction with other techniques such as substitution and encryption-functions like XOR. Substitution techniques usually use some simple strategy such as a lookup table or the XOR-function. Other functions such as binary

Table 1: An example of permutation algorithm for encryption/decryption

	byte/character 'G'	Permutation	Inverse Permutation
		{7,5,4,6,3,0,2,1}	{5,7,6,4,2,1,3,0}
Binary	0100 0111	1110 0100	0100 0111
Decimal	71	228	71
Hex	47	E4	47

addition, multiplication and modular arithmetic functions are also common.

Alternatively, the permutation process can include expansion technique. The expansion-permutation approach takes a block of data and expands it into a set of overlapping groups; each group may be small/large compared to the original block. Suppose a block of 24-bits; we can perform expansion-permutation to convert it into a block of 36 bits as follows: (I) Break the 24-bits into six groups of four bits each and (ii) To each group, add the bit that precedes it and the bit that follows it. Now we have six groups with six bits each for a total of 36 bits.

The techniques in the preceding paragraphs are just some of the commonly used methods in symmetric cryptographic algorithms. You can mix and match them to obtain alternative encryption algorithms, which become more complex and secure by using different encryption techniques one after the other. Popular encryption algorithms make use of 8 or 16 different rounds of encryption techniques. Feistel and co-workers while working at IBM in the early 60s first suggested this systematic approach[34]. Feistel proposed that we could approximate the simple substitution cipher by utilizing the concept of a product cipher, which is obtained by combining two or more basic ciphers in sequence in such a way that the final result or product is cryptographically stronger than any of the component ciphers. In particular, Feistel proposed the use of a cipher that alternates substitutions and permutations. In fact, this is a practical application of a proposal by Claude Shannon to develop a product cipher that alternates confusion and diffusion functions.

The terms confusion and diffusion were introduced by Claude Shannon to capture the two basic building blocks for any cryptographic system. Shannon's concern was to thwart cryptanalysis based on statistical analysis. On the other hand, confusion seeks to make the relationship between the statistics of the ciphertext and the value of the encryption key as complex as possible, again to thwart cryptanalyst attempts to discover the key. The reasoning is as follows: Assume the attacker has some knowledge of the statistical characteristics of the plaintext. For example, in a human-readable message in some language, the frequency distribution of the various letters may be known, e.g. English as shown in Fig. 1. Or there may be words or phrases likely to appear in the message (again with the English alphabet: is, th, the etc.).

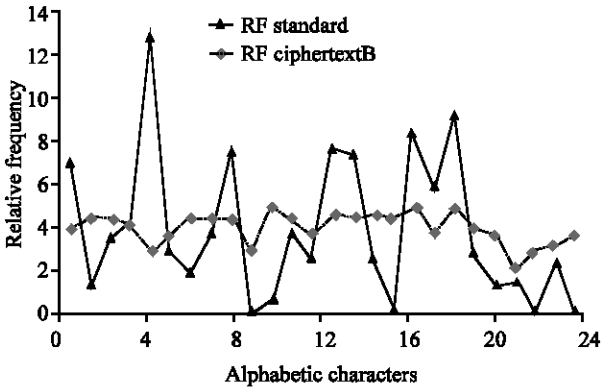

Fig. 1: Comparison of relative frequency distribution between Ciphertext (RF CiphertextB) and English language (RF Standard). Note the absence of similarity in frequency distribution which is caused by combination of several multiple cipher algorithms to result in a strong cipher algorithm to thwart brute-force cryptanalysis

If these statistics are in any way reflected in the ciphertext, the cryptanalyst may be able to deduce the encryption key, or part of the key, or at least a set of keys likely to contain the exact key.

The confusion process is usually implemented via a design function F, which is the heart of the Feistel block cipher and which is the basic building block used in DES. This function relies on the use of S-boxes. This is also the case for most other symmetric block ciphers. The function F provides the element of confusion in a Feistel cipher. Thus, it must be difficult to unscramble the substitution performed by F. One obvious criterion is that F be nonlinear. The more nonlinear F, the more difficult any type of cryptanalysis will be. In rough terms, the more difficult it is to approximate F by a set of linear equations, the more nonlinear F is. Several other criteria should be considered in designing F. The algorithm should have good avalanche properties. It means that a change in one bit of the input should produce a change in many bits of the output. Another version of this is Strict Avalanche Criterion (SAC), which states that any output bit j of an S-box should change with probability ½ when any single input bit I is inverted for all I, j.

In diffusion approach, the statistical structure of the plaintext is dissipated into long-range statistics of the ciphertext. This is achieved by having each plaintext digit affect the value of many ciphertext digits, which is equivalent to saying that each ciphertext digit is affected by many plaintext digits. An example of diffusion is to encrypt a message, $M = m_1, m_2, m_3, \ldots$, of characters with an averaging operation:

$$y_n = \sum_{i=1}^{k} m_{n+i} \pmod{26} \quad (1)$$

adding k successive letters to get a ciphertext letter y_n. One can show that the statistical structure of the plaintext has been dissipated, as is the case in Fig. 1. Thus, even if the attacker can get some handle on the statistics of the ciphertext, the way in which the key was used to produce that ciphertext is so complex as to make it difficult to deduce the key. So, successful are diffusion and confusion in capturing the essence of the desired attributes of a block cipher that they have become the cornerstone of modern cryptographic systems design and implementation in most symmetric crypto-algorithms. The goal of this study is to show how DES crypto-algorithm is accomplished by a combination of these techniques.

THE DATA ENCRYPTION STANDARD (DES)

The Data Encryption Standard (DES) was jointly developed in 1974 by IBM and the US government (US patent 3,962,539) to set a standard that everyone could use to securely communicate with each other. In 1972, the National Institute of Standards and Technology (called the National Bureau of Standards at the time) decided that a strong cryptographic algorithm was needed to protect non-classified information. The algorithm was required to be cheap, widely available and very secure. NIST envisioned something that would be available to the general public and could be used in a wide variety of applications. So they asked for public proposals for such an algorithm. In 1974 IBM submitted the Lucifer algorithm, which appeared to meet most of NIST's design requirements[37].

DES started out as the "Lucifer" algorithm developed by IBM. NIST enlisted the help of the National Security Agency to evaluate the security of Lucifer[37]. At the time many people distrusted the NSA due to their extremely secretive activities, so there was initially a certain degree of skepticism regarding the analysis of Lucifer. One of the greatest worries was that the keylength, originally 128 bits, was reduced to just 56 bits, weakening it significantly. The NSA was also accused of changing the algorithm to plant a back door in it that would allow agents to decrypt any information without having to know the encryption key. But these fears proved unjustified and no such back door has ever been found.

The modified Lucifer algorithm was adopted by NIST as a federal standard on November 23, 1976. Its name was changed to the Data Encryption Standard (DES). The algorithm specification was published in January 1977 and with the official backing of the government it became a very widely employed algorithm in a short amount of time. Since then, the Data Encryption Standard (DES) has proved to be the most popular encryption algorithm of the past twenty-five years. After becoming a U.S. government standard, DES was quickly adopted by other standards bodies worldwide, including ANSI and ISO. The terms of the standard stipulated that it would be reviewed and recertified every five years. NBS recertified DES for the first time in 1987. NIST (NBS after the name change) recertified DES in 1993. DES was quickly adopted for non-digital media, such as voice-grade public telephone lines. Within a couple of years, for example, International Flavors and Fragrances were using DES to protect its valuable formulas transmitted over the phone[38]. Meanwhile, the banking industry, which is the largest user of encryption outside government, adopted DES as a wholesale banking standard. It was also quickly adopted in securing satellite TV broadcasting and communication where it is still playing a major role to-date and is currently used in many different applications around the world. The US National Security Agency (NSA) made several modifications, after which it was adopted as Federal Information Processing Standard (FIPS) standard 46-3 and ANSI standard X3.92. It is officially described in FIPS PUB 46[12].

Over the past three decades DES has proved to be the most important crypto-algorithm ever made. Furthermore, because it had an US National Security Agency (NSA) pedigree, it was widely believed to be secure. However, concerns about its short keylength have mostly dogged the algorithm since the beginning. In the late 1990s, it became widely believed that the NSA was able to break DES by trying every possible key, something called brute force (cryptanalysis) machine capable of breaking DES. This ability was graphically demonstrated by the Electronic Frontier Foundation (EFF) in July 1998, when John Gilmore built a machine for $250,000 that could brute-force a DES key in a few days. Due to these revelations, NIST initiated a program in 1997 to replace DES with a more secure symmetric key crypto-algorithm-the Advanced Encryption Standard (AES)[33].

Years before this, those with interest in more secure applications had already converted to an encryption algorithm called triple-DES (also referred to as 3DES). Triple-DES is the repeated application of three DES encryptions, using two or three different keys[4]. This algorithm leverages all the security of DES while effectively lengthening the key and is in wide use today to protect all kinds of personal, business and financial secrets. To-date DES has proved to be the most studied encryption algorithm ever invented and many

cryptographers went to school on DES. Almost all of the newer encryption algorithms in use today can trace their roots back to DES and research papers analyzing different aspects of DES are still being published today. Note that mainly due to its short keylength, DES expired as a US standard in 1998, with AES effectively replacing it as the crypto of choice.

THEORY AND WORKING OF DES

The Data Encryption Standard (DES) is a symmetric key crypto scheme that uses a 56-bit key, which is today considered by many to be insufficient as it can with moderate effort be cracked by brute force. It operates on blocks of 64 bits using a secret key that is 56 bits long. The original proposal used a secret key that was 64 bits long. It is widely believed that the removal of these 8 bits from the key was done to make it possible for US government agencies to secretly crack messages.

Data Security Standard (DES) was originally designed to be implemented only in hardware systems and is therefore extremely slow in software applications. The algorithm, although complicated, is pretty straightforward. It uses only simple logical operations on small groups of bits and could be implemented fairly efficiently in the mid-1970s hardware of the time[39,40]. DES is not very efficient in software, especially the 32-bit architectures that are common today. Its overall structure was something called a Feistel network, also used in another IBM design called Lucifer, which is often considered to be a precursor to DES[37]. DES is a block cipher, meaning that it encrypts and decrypts data in blocks: 64-bit blocks. DES is an iterated cipher; meaning that it contains 16 iterations (called rounds) of a simpler cipher, (Fig. 2). The algorithm's primary strength came from S-box design, a non-linear table-lookup operation.

The specific utilization and the implementation of the DES is usually based on many factors particularly to the computer system and its associated components. For security reasons and hence cryptanalysis threat defense, the cryptographic algorithm specified in this standard transforms a 64-bit binary value into a unique 64-bit binary value based on a 56-bit variable. If the complete 64-bit input is used (i.e., none of the input bits should be predetermined from block to block) and if the 56-bit variable is randomly chosen, no technique other than trying all possible keys using known input and output for the DES will guarantee finding the chosen key[41-43]. As there are over 70×10^{15} (seventy quadrillion) possible keys of 56-bits, the feasibility of deriving a particular key in this way is extremely unlikely in typical threat environments. Moreover, if the key is changed frequently, the risk of this

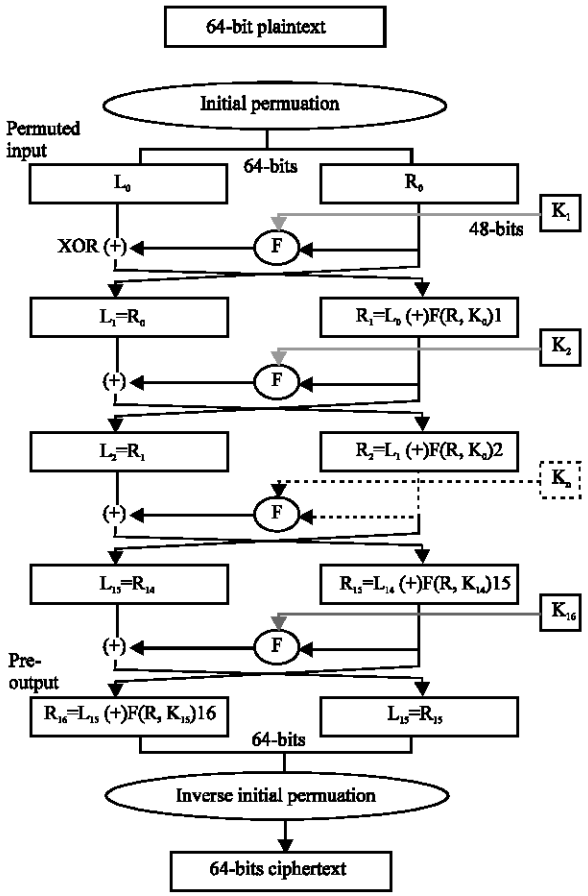

Fig. 2: Schematic DES algorithm

event is greatly diminished. However, users should be aware that it is theoretically possible to derive the key in fewer trials (with a correspondingly lower probability of success depending on the number of keys tried) and should be cautioned to change the key as often as practical. Users must also provide the cryptographic key a high level of protection in order to minimize the potential risks of its unauthorized computation or acquisition. The feasibility of computing the correct key may change with advances in cryptographic technology. However, when correctly implemented and properly used, this standard will provide a high level of cryptographic protection to electronic and digital data.

Detailed design and implementation of DES: Recall that DES is a block cipher that operates on plaintext blocks of a given size (64-bits) and returns ciphertext blocks of the same size (Fig. 2). Thus, DES results in a permutation among the 2^{64} possible arrangements of 64-bits, each of which may be either 0 or 1. Each block of 64 bits is divided into two blocks of 32-bits each, a left half block L and a

right half R. (This division is only used in certain operations).

Example: Let M be the plain text message M=0123456789ABCDEF, where, M is in hexadecimal (base 16) format. Rewriting M in binary format, we get the 64-bit block of text:

M = 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
L = 0000 0001 0010 0011 0100 0101 0110 0111
R = 1000 1001 1010 1011 1100 1101 1110 1111

The first bit of M is "0". The last bit is "1". read from left to right.

DES operates on the 64-bit blocks using key sizes of 56- bits. The keys are actually stored as being 64 bits long, but every 8th bit in the key is not used (i.e., bits numbered 8, 16, 24, 32, 40, 48, 56 and 64). However, we number the bits from 1 to 64, going left to right, in the calculations below. But, as you will see, the eight bits just mentioned get eliminated when we create subkeys.

Example: Let K be the hexadecimal key K=133457799BBCDFF1. This gives us as the binary key (setting 1 = 0001, 3 = 0011, etc. and grouping together every eight bits, of which the last one in each group will be unused):

K = 00010011 00110100 01010111 01111001 10011011 10111100 11011111 11110001

The DES algorithm uses the following steps:

Step 1: Create 16 subkeys, each of which is 48-bits long
The 64-bit key is permuted according to table: Permutation PC-1. Since the first entry in the table is 57, this means that the 57th bit of the original key K becomes the first bit of the permuted key K+. The 49th bit of the original key becomes the second bit of the permuted key. The 4th bit of the original key is the last bit of the permuted key. Note only 56 bits of the original key appear in the permuted key.

Permutation PC-1

Bit	0	1	2	3	4	5	6
1	57	49	41	33	25	17	9
8	1	58	50	42	3	26	18
15	10	2	59	51	43	35	27
22	19	11	3	60	52	44	36
29	63	55	47	39	31	23	15
36	7	62	54	46	3	30	22
43	14	6	61	53	45	37	29
50	21	13	5	28	20	12	4

For example, PC-1 table can be used to figure out how bit-30 of the original 64-bit key transforms to a bit in the new 56-bit key. Find the number 60 in the table and notice that it belongs to the column labeled 3 and the row labeled 22. Add up the value of the row and column to find the new position of the bit within the key. For bit-60, 22+3=25, so bit-60 becomes bit-25 of the new 56-bit key. Note that bits-8, 16, 24, 32, 40, 48, 56 and 64 of the original key are not in the table. These are the unused parity bits that are discarded when the final 56-bit key is created.

Example: From the original 64-bit key:

K = 00010011 00110100 01010111 01111001 10011011 10111100 11011111 11110001

we get the 56-bit permutation:
K+ = 1111000 0110011 0010101 0101111 0101010 1011001 1001111 0001111

Next, split this key into left and right halves, C_0 and D_0, where each half has 28 bits.

Example: From the permuted key K+

C_0 = 1111000 0110011 0010101 0101111
D_0 = 0101010 1011001 1001111 0001111

With C_0 and D_0 defined, create sixteen blocks C_n and D_n, 1<=n<=16. Each pair of blocks C_n and D_n is formed from the previous pair C_{n-1} and D_{n-1}, respectively, for n = 1, 2, ..., 16, using the following schedule of "left shifts" of the previous block. To do a left shift, move each bit one place to the left, except for the first bit, which is cycled to the end of the block.

Iteration number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Number of left shift	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

This means, for example, C_3 and D_3 are obtained from C_2 and D_2, respectively, by two left shifts and C_{16} and D_{16} are obtained from C_{15} and D_{15}, respectively, by one left shift. In all cases, by a single left shift is meant a rotation of the bits one place to the left, so that after one left shift the bits in the 28 positions are the bits that were previously in positions 2, 3,..., 28, 1.

Example: From original pair C_0 and D_0

C_0 = 1111000011001100101010101111
D_0 = 0101010101100110011110001111

C_1 = 11100001100110010101011111
D_1 = 10101010110011001111100011110

C_2 = 11000011001100101010111111
D_2 = 01010101100110011110001111101

C_3 = 00001100110010101011111111
D_3 = 01010110011001111100011110101

C_4 = 00110011001010101011111111100
D_4 = 01011001100111100011110101

C_5 = 11001100101010101111111110000
D_5 = 01100110011110001111010101

C_6 = 00110010101010111111111000011
D_6 = 10011001111000111101010101

C_7 = 11001010101011111111100001100
D_7 = 01100111100011110101010110

C_8 = 00101010101111111110000110011
D_8 = 10011110001111010101011001

C_9 = 01010101011111111100001100110
D_9 = 00111100011110101010110011

C_{10} = 01010101111111110000110011001
D_{10} = 11110001111010101011001100

C_{11} = 01010111111110000110011001 01
D_{11} = 11000111101010101100110011

C_{12} = 01011111111000011001100101 01
D_{12} = 00011110101010101100110011 11

C_{13} = 01111111100001100110010101 01
D_{13} = 01111010101010110011001111 00

C_{14} = 11111110000110011001010101 01
D_{14} = 11101010101011001100111100 01

C_{15} = 11111000011001100101010101 11
D_{15} = 10101010101100110011110000 111

C_{16} = 11110000110011001010101011 11
D_{16} = 01010101011001100111100011 11

We now form the keys K_n, for 1<= n<=16, by applying the following permutation table to each of the concatenated pairs C_nD_n. Each pair has 56 bits, but the table, Permutation PC-2, only uses 48 of these.

Permutation PC-2

Bit	0	1	2	3	4	5
1	14	17	11	24	1	57
7	3	28	15	6	21	10
13	23	19	12	4	26	8
19	16	7	27	20	13	2
25	41	52	31	37	47	55
31	30	40	51	45	33	48
37	44	49	39	56	34	53
43	46	42	50	36	29	32

Therefore, the first bit of K_n is the 14th bit of C_nD_n, the second bit the 17th and so on, ending with the 48th bit of K_n being the 32th bit of C_nD_n.

Example: For the first key

C_1D_1 = 11100001100110010101010111111 1010101 0110011 0011110 0011110

which, after applying the Permutation PC-2, becomes:

K_1 = 000110 110000 001011 101111 111111 000111 000001 110010

For the other keys we have:

K_2 = 011110 011010 111011 011001 110110 111100 100111 100101

K_3 = 010101 011111 110010 001010 010000 101100 111110 011001

K_4 = 011100 101010 110111 010110 110110 110011 010100 011101

K_5 = 011111 001110 110000 000111 111010 110101 001110 101000

K_6 = 011000 111010 010100 111110 010100 000111 101100 101111

K_7 = 11011 001000 010010 110111 111101 100001 100010 111100

K_8 = 111101 111000 101000 111010 110000 010011 101111 111011

K_9 = 111000 001101 101111 101011 111011 011110 011110 000001

K_{10} = 101100 011111 001101 000111 101110 100100 011001 001111

K_{11} = 001000 010101 111111 010011 110111 101101 001110 000110

K_{12} = 011101 010111 000111 110101 100101 000110 011111 101001

K_{13} = 100101 111100 010111 010001 111110 101011 101001 000001

K_{14} = 010111 110100 001110 110111 111100 101110 011100 111010

K_{15} = 101111 111001 000110 001101 001111 010011 111100 001010

K_{16} = 110010 110011 110110 001011 000011 100001 011111 110101

End of subkeys manipulation. Next look at the plaintext message itself.

Step 2: Encode each 64-bit block of data

The first procedure is to perform an initial permutation IP of the 64 bits of the plaintext message data M. This rearranges the bits according to the table, Initial Permutation IP, where the entries in the table show the new arrangement of the bits from their initial order. The

58th bit of M becomes the first bit of IP. The 50th bit of M becomes the second bit of IP. The 7th bit of M is the last bit of IP.

Initial Permutation IP

Bit	0	1	2	3	4	5	6	7
1	58	50	42	34	26	18	10	2
9	60	52	44	36	28	20	12	4
17	62	54	46	38	30	22	14	6
25	64	56	48	40	32	24	16	8
33	57	49	41	33	25	17	9	1
41	59	51	43	35	27	19	11	3
49	63	55	47	39	31	23	25	7

Example: Applying the initial permutation IP to the block of text M, given previously, we have:

M = 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111

IP = 1100 1100 0000 0000 1100 1100 1111 1111 1111 0000 1010 1010 1111 0000 1010 1010

Here the 58th bit of M is "1", which becomes the first bit of IP. The 50th bit of M is "1", which becomes the second bit of IP. The 7th bit of M is "0", which becomes the last bit of IP.

Next divide the permuted block IP into a left half L_0 of 32 bits and a right half R_0 of 32 bits.

Example: From IP, we get L_0 and R_0:

L_0 = 1100 1100 0000 0000 1100 1100 1111 1111

R_0 = 1111 0000 1010 1010 1111 0000 1010 1010

We now proceed through 16 iterations, for 1<=n<=16, using a function f which operates on two blocks--a data block of 32-bits and a key K_n of 48-bits--to produce a block of 32-bits. Let + denote XOR addition, (i.e., bit-by-bit addition modulo 2). Then for n going from 1 to 16 we calculate:

$$L_n = R_{n-1}$$
$$R_n = L_{n-1} + f(R_{n-1}, K_n)$$

This results in a final block, for n = 16, of $L_{16}R_{16}$. That is, for each iteration we take the right 32-bits of the previous result and make them the left 32-bits of the current step. For the right 32-bits in the current step, we XOR the left 32-bits of the previous step with the calculation f.

Example: For n = 1, we have:

K_1 = 000110 110000 001011 101111 111111 000111 000001 110010

$L_1 = R_0$ = 1111 0000 1010 1010 1111 0000 1010 1010

$R_1 = L_0 + f(R_0, K_1)$

It remains to explain how the function f works. To calculate f, we first expand each block R_{n-1} from 32-bits to 48-bits. This is done by using a selection table, E-Bit-Selection Table, that repeats some of the bits in R_{n-1}. Call the use of this selection table the function E. Thus $E(R_{n-1})$ has a 32-bit input block and a 48-bit output block.

Let E be such that the 48-bits of its output, written as 8 blocks of 6-bits each, are obtained by selecting the bits in its inputs in order according to the, E-Bit-Selection Table, shown below:

E-Bit-Selection Table

Bit	0	1	2	3	4	5
1	32	1	2	3	4	5
7	4	5	6	7	8	9
13	8	9	10	11	12	13
19	12	13	14	15	16	17
25	16	17	18	19	20	21
31	20	21	22	23	24	25
37	24	25	26	27	28	29
43	28	29	30	31	32	1

Thus the first three bits of $E(R_{n-1})$ are the bits in positions 32, 1 and 2 of R_{n-1} while the last 2 bits of $E(R_{n-1})$ are the bits in positions 32 and 1.

Example: We calculate $E(R_0)$ from R_0 as follows:

R_0 = 1111 0000 1010 1010 1111 0000 1010 1010

$E(R_0)$ = 011110 100001 010101 010101 011110 100001 010101 010101

(Note that each block of 4 original bits has been expanded to a block of 6 output bits.)

Next in the f calculation, we XOR the output $E(R_{n-1})$ with the key K_n: $K_n + E(R_{n-1})$.

	S1															
	Column Number															
Row No	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

Example: For K_1, $E(R_0)$, we have

K_1 = 000110 110000 001011 101111 111111 000111 000001 110010

$E(R_0)$ = 011110 100001 010101 010101 011110 100001 010101 010101

$K_1+E(R_0)$ = 011000 010001 011110 111010 100001 100110 010100 100111.

Calculating the function f is not finished. To this point we have expanded R_{n-1} from 32-bits to 48-bits, using the selection table and XORed the result with the key K_n. Now 48-bits available, or eight groups of 6-bits. Use them as addresses in tables called "S-boxes". Each group of six bits will give us an address in a different S-box. Located at that address will be a 4-bit number. This 4-bit number will replace the original 6-bits. The net result is that the eight groups of 6-bits are transformed into eight groups of 4 bits (the 4-bit outputs from the S-boxes) for 32-bits total.

We write the previous result, which is 48 bits, in the form:

$$K_1 + E(R_{n-1}) = B_1B_2B_3B_4B_5B_6B_7B_8,$$

where each B_i is a group of six bits. Next calculate:

$$S_1(B_1)S_2(B_2)S_3(B_3)S_4(B_4)S_5(B_5)S_6(B_6)S_7(B_7)S_8(B_8)$$

where, $S_i(B_i)$ refers to the output of the I-th S-box.

Note again that each of the functions S1, S2,..., S8, takes a 6-bit block as input and yields a 4-bit block as output. The table to determine S_1 is shown and explained below:

If S_1 is the function defined in this table and B is a block of 6 bits, then $S_1(B)$ is determined as follows: The first and last bits of B represent in base 2 a number in the decimal range 0 to 3 (or binary 00 to 11). Let that number be I. The middle 4 bits of B represent in base 2 a number in the decimal range 0 to 15 (binary 0000 to 1111). Let that number be j. Look up in the table the number in the I-th row and j-th column. It is a number in the range 0 to 15 and is uniquely represented by a 4 bit block. That block is the output $S_1(B)$ of S_1 for the input B. For example, for input block B = 011011 the first bit is "0" and the last bit "1" giving "01" as the row. This is row 1. The middle four bits are "1101". This is the binary equivalent of decimal 13, so the column is column number 13. In row 1, column 13 appears 5. This determines the output; 5 is binary 0101, so that the output is 0101. Hence $S_1(011011) = 0101$.

The tables defining the functions $S_1,...,S_8$ are the following:

S1															
14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S2															
15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S3															
10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S4															
7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S5															
2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S6															
12	1	10	15	9	2	6	8	0	03	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S7															
4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	15

S8															
13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Example: For the first round i.e.,

$K_1 + E(R_0) =$ 011000 010001 011110 111010 100001 100110 010100 100111.

Note again the eight numbers are extracted from the S-boxes - one from each box as shown below. In each case of S[n][row][column], the first and last bits of the current B[n] are used as the row index and he middle four bits as the column index (as explained above). Taking the sequence of 6-bits at a time from,
$K_1 + E(R_0)$ and performing the 8-S-box operations, we have:

S[1]B[1] = S[1](00, 1100) = S[1][0][12] = 5 = 0101
S[2]B[2] = S[2](01, 1000) = S[2][1][8] = 12 = 1100
S[3]B[3] = S[3](00, 1111) = S[3][0][15] = 8 = 1000
S[4]B[4] = S[4](10, 1101) = S[4][2][13] = 2 = 0010
S[5]B[5] = S[5](11, 0000) = S[5][3][0] = 11 = 1011
S[6]B[6] = S[6](10, 0011) = S[6][2][3] = 5 = 0101
S[7]B[7] = S[7](00, 1010) = S[7][0][10] = 9 = 1001
S[8]B[8] = S[8](11, 1010) = S[8][3][10] = 7 = 0111

Putting all together we get:

$S_1(B_1)S_2(B_2)S_3(B_3)S_4(B_4)S_5(B_5)S_6(B_6)S_7(B_7)S_8(B_8) =$
0101 1100 1000 0010 1011 0101 1001 0111

The final stage in the calculation of f is to do a permutation using the Permutation P look up table on the S-box output above to obtain the final value of f:

$$f = P(S_1(B_1)S_2(B_2)...S_8(B_8))$$

The permutation P operation is defined using the Permutation P table below. The permutation P yields a 32-bit output from a 32-bit input by permuting the bits of the input block.

Permutation P

Bit	0	1	2	3
1	16	7	20	21
5	29	12	28	17
9	1	15	23	26
13	5	18	31	10
17	2	8	24	14
21	32	27	3	9
25	19	13	30	6
29	22	11	4	25

Example: From the output of the eight S boxes:

$S_1(B_1)S_2(B_2)S_3(B_3)S_4(B_4)S_5(B_5)S_6(B_6)S_7(B_7)S_8(B_8) =$
0101 1100 1000 0010 1011 0101 1001 0111

Perform Permutation P operation as:

$f = P(S_1(B_1)S_2(B_2)...S_8(B_8)) =$ 0010 0011 0100 1010 1010 1001 1011 1011

$R_1 = L_0 + f(R_0, K_1) =$ 100 1100 0000 0000 1100 1100 1111 1111+ 0010 0011 0100 1010 1010 1001 1011 1011
$=$ 1110 1111 0100 1010 0110 0101 0100 0100

In the next round, $L_2 = R_1$, which is the block we just calculated and then, $R_2 = L_1 + f(R_1, K_2)$, is calculated and so on for 16 rounds. At the end of the sixteenth round we have the blocks L_{16} and R_{16}. Then reverse the order of the two blocks into the 64-bit block

$$R_{16}L_{16}$$

and apply a Final Permutation IP^{-1} as defined by the lookup table below:

Final Permutation IP^{-1}

Bit	0	1	2	3	4	5	6	7
1	40	8	48	16	56	24	64	32
9	39	7	47	15	55	23	63	31
17	38	6	46	14	54	22	62	30
25	37	5	45	13	53	21	61	29
33	36	4	44	12	52	20	60	28
41	35	3	43	11	51	19	59	27
49	34	2	42	10	50	18	58	26
57	33	1	41	9	49	17	57	25

The IP and IP^{-1} lookup tables are used just like PC-1 and PC-2 were for the key scheduling. That is, the output of the algorithm has bit-40 of the preoutput block as its first bit, bit-8 as its second bit and so on, until bit-25 of the preoutput block is the last bit of the output. By looking at the IP^{-1} table it becomes apparent why one permutation is called the inverse of the other. For example, let's examine how bit-35 is transformed under IP. In the IP table, bit-35 is located at the intersection of the column labeled 3 and the row labeled 41. So this bit becomes bit 44 of the original 64-bit block after the permutation. Now let's apply IP^{-1}. From the IP^{-1} lookup table, bit-44 is located at the intersection of the column labeled 2 and the row labeled 33. So bit-44 becomes bit-35 after the final permutation, IP^{-1}. And this is the bit position that we started with before the first permutation. So IP^{-1} really is the inverse of IP. It does the exact opposite of IP. If you run a block of plaintext through IP and then pass the resulting block through IP^{-1} you'll end up with the original block.

Example: If we process all 16 blocks using the method defined previously we get on the 16th round,

L_{16} = 0100 0011 0100 0010 0011 0010 0011 0100
R_{16} = 0000 1010 0100 1100 1101 1001 1001 0101

reverse the order of these two blocks and apply the final permutation to:

$R_{16}L_{16}$ = 00001010 01001100 11011001 10010101 01000011
01000010 00110010 00110100
IP^{-1} = 10000101 11101000 00010011 01010100 00001111
00001010 10110100 00000101

Which in hexadecimal format is: 85E813540F0AB405.

This is the encrypted form of M = 0123456789ABCDEF: namely, C = 85E813540F0AB405.

Decryption is simply the inverse of encryption, following the same steps as above, but reversing the order in which the subkeys are applied.

Confidentiality modes of DES operation: A block cipher processes (encrypts or decrypts) messages as data blocks. Usually, the size of a bulk message (i.e., a message string) is larger than the size of the message block cipher, the long message is divided into series of sequential listed message blocks and the cipher processes these blocks one at a time. A number of different modes of operation have been devised on top of an underlying block cipher algorithm. These modes of operation (except a trivial case of them) provide several desirable properties to the ciphertext blocks, such as adding nondeterminism (randomness) to a block cipher algorithm, for example, padding plaintext messages to an arbitrary length (so that the length of a ciphertext needn't be related to that of the corresponding plaintext), control of error propagation, generation of key stream for a stream cipher, etc.

The DES algorithm as we have seen turns a 64-bit message block M into a 64-bit cipher block C. If each 64-bit block is encrypted individually, then the mode of encryption is called Electronic Code Book (ECB) mode. There are two other modes of DES encryption, namely Chain Block Coding (CBC) and Cipher Feedback (CFB), which make each cipher block dependent on all the previous messages blocks through an initial XOR operation and; Counter (CTR) mode. Our description above follows the most recent NIST recommendation[44,45].

Electronic Code Book (ECB): This is the regular DES algorithm, exactly as described above. Data is divided into 64-bit blocks and each block is encrypted one at a time. Separate encryptions with different blocks are totally independent of each other. This means that if data is transmitted over a network or phone line, transmission errors will only affect the block containing the error. It also means, however, that the blocks can be rearranged, thus scrambling a file beyond recognition and this action would go undetected. ECB is the weakest of the various modes because no additional security measures are implemented besides the basic DES algorithm. However, ECB is the fastest and easiest to implement, making it the most common mode of DES seen in commercial applications. This is the mode of operation used by Private Encryptor.

Cipher Block Chaining (CBC): In this mode of operation, each block of ECB encrypted ciphertext is XORed with the next plaintext block to be encrypted, thus making all the blocks dependent on all the previous blocks. This means that in order to find the plaintext of a particular block, you need to know the ciphertext, the key and the ciphertext for the previous block. The first block to be encrypted has no previous ciphertext, so the plaintext is XORed with a 64-bit number called the Initialization Vector, or IV for short. So if data is transmitted over a network or phone line and there is a transmission error (adding or deleting bits), the error will be carried forward to all subsequent blocks since each block is dependent upon the last. If the bits are just modified in transit (as is the more common case) the error will only affect all of the bits in the changed block and the corresponding bits in the following block. The error doesn't propagate any further. This mode of operation is more secure than ECB because the extra XOR step adds one more layer to the encryption process.

Cipher Feedback (CFB): In this mode, blocks of plaintext that are less than 64-bits long can be encrypted. Normally, special processing has to be used to handle files whose size is not a perfect multiple of 8 bytes, but this mode removes that necessity. (Private Encryptor handles this case by adding several dummy bytes to the end of a file before encrypting it, also known as padding.) The plaintext itself is not actually passed through the DES algorithm, but merely XORed with an output block from it, in the following manner: A 64-bit block called the Shift Register is used as the input plaintext to DES. This is initially set to some arbitrary value and encrypted with the DES algorithm. The ciphertext is then passed through an extra component called the M-box, which simply selects the left-most M bits of the ciphertext, where M is the number of bits in the block we wish to encrypt. This value

is XORed with the real plaintext and the output of that is the final ciphertext. Finally, the ciphertext is fed back into the Shift Register and used as the plaintext seed for the next block to be encrypted. As with CBC mode, an error in one block affects all subsequent blocks during data transmission. This mode of operation is similar to CBC and is very secure, but it is slower than ECB due to the added complexity.

Output Feedback (OFB): This is similar to CFB mode, except that the ciphertext output of DES is fed back into the Shift Register, rather than the actual final ciphertext. The Shift Register is set to an arbitrary initial value and passed through the DES algorithm. The output from DES is passed through the M-box and then fed back into the Shift Register to prepare for the next block. This value is then XORed with the real plaintext (which may be less than 64-bits in length, like CFB mode) and the result is the final ciphertext. Note that unlike CFB and CBC, a transmission error in one block will not affect subsequent blocks because once the recipient has the initial Shift Register value, it will continue to generate new Shift Register plaintext inputs without any further data input. However, this mode of operation is less secure than CFB mode because only the real ciphertext and DES ciphertext output is needed to find the plaintext of the most recent block. Knowledge of the key is not required.

The DES variant: triple-DES (3DES): This algorithm is a variation of DES; it uses the DES encryption algorithm three times with two different keys. Given a plaintext message, the first key is used to DES-encrypt the message. The second key is used to DES-decrypt the encrypted message. (Since the second key is not the right key, this decryption just scrambles the data further.) The twice-scrambled message is then encrypted again with the first key to yield the final ciphertext. This three-step procedure is called triple-DES. (Triple-DES can also be done with three separate keys instead of only two. In either case the resultant key space is about 2^{112}.) This process creates an encrypted datastream that is unbreakable with today's code-breaking techniques and available computing power, while being compatible with DES. Financial institutions currently use this technique.

Other modern encryption techniques in this category are:

RC codes: Rivest codes are named after MIT professor Ronald Rivest who is also the co-inventor of the RSA public key encryption algorithm. These methods are proprietary algorithms that are distributed by RSA Data Security. The two most popular codes are RC2 (which is a block cipher method like DES) and RC4 (which is a stream cipher that produces a stream of pseudo-random numbers that are XORed with the information). These codes can be used with keys from 1 to 1,024 bits in length. There is no estimate of how secure these codes really are because they are proprietary[4,11].

IDEA: In 1990 James L. Massey and Xuejia Lai developed and published the International Data Encryption Algorithm (IDEA) in Zurich, Switzerland. This technique uses a 128-bit key and seems to be very strong (although the exact nature of the security it provides is not known)[4,11].

Skipjack: This is a (another alternative block-cipher encrypter) secret algorithm that was developed by the National Security Agency for civilian purposes. It uses an 80-bit key. It is at the heart of the Clipper chip used by law enforcement agencies to perform legal wiretaps. The Clipper chip is not as secure as Skipjack[4,11].

DES SECURITY ISSUES-CRACKING DES

Recall that DES uses a key that is 56 bits, or seven characters long. At the time of its development, it was believed that trying out all 72,057,594,037,927,936 possible keys (a seven with 16 zeros) would be impossible because computers could not possibly ever become fast enough[46].

Before DES was adopted as a national standard, during the period NBS (forerunner to NIST) was soliciting comments on the proposed algorithm, the creators of public key cryptography, Martin Hellman and Whitfield Diffie, registered some objections to the use of DES as an encryption algorithm. Hellman wrote: Whit Diffie and I have become concerned that the proposed data encryption standard, while probably secure against commercial assault, may be extremely vulnerable to attack by an intelligence organization (letter to NBS, October 22, 1975).

Diffie and Hellman then outlined a brute force attack on DES. (By "brute force" is meant that you try as many of the 2^56 possible keys as you have to before decrypting the ciphertext into a sensible plaintext message.) They proposed a special purpose "parallel computer using one million chips to try one million keys each" per second and estimated the cost of such a machine at $20 million.

Fast forward to 1998. Under the direction of John Gilmore of the EFF, a team spent $220,000 and built a special-purpose machine that could decrypt a message by trying out all possible keys i.e., the 56-bit DES key space

in an average of 4.5 days. The machine cost less than $250,000 and searched over 88 billion keys per second. On July 17, 1998; they further announced they had cracked a 56-bit key in 56 h. The computer, called Deep Crack, uses 27 boards each containing 64 chips and is capable of testing 90 billion keys a second.

Despite this, just around this time of Deep Crack success in cracking DES, Robert Litt, the then principal associate deputy attorney general at the Department of Justice, denied it was possible for the FBI to crack DES and he commented: "Let me put the technical problem in context: It took 14,000 Pentium computers working for four months to decrypt a single message . . . . We are not just talking FBI and NSA [needing massive computing power], we are talking about every police department.

Responded cryptography expert Bruce Schneier: the FBI is either incompetent or lying, or both. Schneier went on to say: The only solution here is to pick an algorithm with a longer key; there isn't enough silicon in the galaxy or enough time before the sun burns out to brute-force triple-DES (Crypto-Gram, Counterpane Systems, August 15, 1998).

Some relevant sums: In cracking DES, the DES cracker searches a 2^{56} key space (72,058,000,000,000,000 keys) at a speed of 33.333 MHZ (i.e., 33.333 million keys/second). To search the entire key space would therefore take 68.50 years.

In real practice the DES cracker is actually searching for up to 16384 keys in parallel. If the whole key space were searched it would find keys at an average rate of one per 68.50/16384 years, which is one every 36.65 h.

To calculate the expected time until the first key is found, one treats the system as having a Poisson distribution. Since the linear feedback shiftback register is walking the keyspace pretty much at random this is a reasonable assumption.

So we have 2^{56} possible keys in the keyspace and 2^{14} possible matches. Let the mean rate of matching be: $m=2^{14}/2^{56}=2^{-42}$. The probability that any particular cell is empty is: $(m_0/0!)e^{-m}$. Simplifying this, (m_0=1 and 0!=1), we get the probability that a cell is empty is given by: e^{-m}.

Let the probability that the first r cells are empty be p.

Then $p=(e^{-m})^r$ [independent events].

So $p=^{-mr}$ Taking the natural log of both sides and solving for r we have: r=-In(p)/m

So for a probability, p, equal to 0.5 (i.e., average luck) for our value of m we need to search r keys, where: $r=6931472/(2.2727734)^{-13}\simeq 3,048,494,000,000$ keys. At 33.333MHz this will take 25.40 h.

Note carefully that if your luck is worse, then one time in a thousand cracking attempts (i.e., p=0.001) then a lot more keys will need to be searched. In fact it will take about 10.5 days at 33.333MHz. In practice cracks have been found that ran in times between 5 and 37 h.

Key lengths and possible combination of breaking the key:

- 40-bit, 2^{40} = 1,099,511,627,776
- 64-bit, 2^{64} = 18,446,744,073,709,551,616
- 128-bit, 2^{128} = 3.4028E+36
- 192-bit, 2^{192} = ??? Give it a try!

PRACTICAL APPLICATION OF DES AND ITS VARIANTS

Maximizing the benefits of a security upgrade-triple DES encryption: Increased competition, new mandates and technological advances, are all contributing to the increased pace of change in a modern banking environment where secure banking is paramount in this day and age of global economy. Today bank customer's wants to spent and/or get access to their money on the fly and in a secure manner - a la personalized smartcard (secure moneycard)[47,48]. And this without any doubt is contributing to a turbulent time for European banking industry and financial institutions. Of these, one of the most high profile and high impact changes is undoubtedly the widespread adoption of EMV (Europay-MasterCard-Visa) microchip-based smartcard. For this reason, banks throughout the European region are being encouraged by the global card schemes (MasterCard and Visa) and national payment associations (such as the UK's APACS and the French Groupement des Cartes Bancaires), to make some fundamental changes to the way their card payment systems work. At the same time as they are updating their systems to support EMV, many banks must also manage the transition from DES to Triple-DES encryption due to security issues concerning DES. These significant migration projects need not be considered in isolation. Banks can minimize the impact and cost of changes by identifying system components for which a single update can accommodate both requirements.

The necessity to move to Triple DES Encryption has resulted from the widespread acceptance of the fact that Single DES is no longer secure enough to protect financial transactions. It has been demonstrated that DES keys can be cracked using relatively low cost hardware in less than 24 h today. To counter this threat, MasterCard and Visa

are introducing sets of mandates and recommendations, which lay out a timetable for the deployment of Triple-DES for the various components of a bank's card processing systems. Under this circumstances the back office within the banks, which host systems and critical devices such as the Host Security Modules (HSMs), are usually among the first to be upgraded.

In parallel to their Triple DES Encryption plans, European banks must also consider how to introduce support for the acquiring and issuance of EMV smart cards. One of the main drivers for this is the knowledge that their exposure to fraud losses may dramatically increase starting 2005, when the liability shift rules from Visa and MasterCard come into effect. If banks do not support EMV by this time and the other party in a transaction does, then they may automatically become liable for any fraud. A frightening scenario, when fraud losses in the UK alone during 2002 totaled nearly £425 million.

Like Triple-DES, the migration to EMV will affect many components of a bank's systems. Ensuring that the Host Security Module in the back office has the necessary functionality is one of the first steps. In order to clarify the requirements for Triple-DES and EMV capable Host Security Modules, Europay (now MasterCard Europe) has developed the Europay Security Platform (ESP) specifications. The legacy Europay Security Module (ESM) devices that ESP replaces are used by many of Europay's Member banks and are neither Triple-DES capable, or upgradeable. They must therefore all be replaced. For these reasons the ESP specifications have been designed to allow banks to buy off-the-shelf HSM products that can be integrated into the MasterCard Europe payment card infrastructure.

System changes for Triple-DES and EMV migration require considerable investment. However, many banks are taking the opportunity to tackle some other related issues that will simplify their key management and provide tangible benefits to their business. For example, MasterCard's ESP specifications includes features to simplify the transferal of sets of encryption keys used for magnetic stripe and EMV transaction processing, between a bank and the MasterCard key management centre. This enables EMV card issuing banks to utilize MasterCard Europe's stand in services, for all of their card products-both magnetic stripe and chip.

Another significant development, this time in the ATM world, which can be piggybacked onto the Triple-DES changes, is the provision of support for secure and automated methods to initialize ATM encryption keys. In order to upgrade most ATMs to Triple-DES, the integral Encrypting PIN Pads must be replaced. ATM manufacturers have recognized that this presents an opportunity to move away from the traditional, manual methods of loading private keys required for symmetric cryptosystems when ATMs are commissioned.

The new Triple-DES capable PIN Pad products from vendors such as NCR and Diebold are pre-loaded with RSA keys and certificates during manufacture. Alternatively they could be re-loaded with newest public-key crypto - the Elliptic Curve Cryptosystems (ECC) - that uses smaller keylength but provide same crypto-strength as RSA[1]. The ability to load the RSA crypto-algorithm on-board the HSM removes the need to install the private-keys required by the symmetric ciphers. A Host Security Module equipped with a complementary set of keys and certificates can use RSA techniques to encrypt and sign packages containing an ATM's unique Triple-DES initialization keys. These can be sent to an ATM over the communication lines used for normal transaction messages on-the-fly, thereby eliminating the need for teams of trusted engineers to manually enter sensitive key components at each ATM. (Recall that trusted engineers are amenable to key-nicking via social engineering!) This can represent a significant cost saving and improved security for the banks owning the ATMs and this would enable them to maximize the benefits from their investment in Triple-DES migration. The process can easily be automated, enabling the ATM's initialization keys to be regularly replaced for increased security.

It is important to note that there are clear benefits, in terms of cost savings and increased efficiency, if a number of security upgrades are undertaken at the same time rather than looking at each in isolation. The technology is there for the taking but are banks looking at the big picture - time will tell!?

HARDWARE BASED DATA ENCRYPTION AND COMPRESSION

For commercial, banking and financial institutions wishing to link remote networks together, secure network Application Specific Integrated Circuit (sn-ASIC) crypto-network cards can offer high levels of security within the router-based security system. Since all wide area traffic passes through a router, this is an ideal place to locate the complex hardware required to provide secure data encryption. In addition, integrating the complex key management procedures into the router management system reduces the cost of supporting an encrypted network. A further advantage would be to implement an sn-ASIC public-key capable system allowing for initialization of symmetric session keys. In such circumstances high-level classified information from

security sensitive applications can be fully encrypted using the latest cryptographic techniques before transmission across the insecure wide area network, ensuring the integrity and confidentiality of the data. Secure ASIC-cards can also be fitted directly into routers to provide integrated high-performance hardware-based compression and DES/3DES encryption on all validated PPP and Frame Relay links. Dedicated 32-bit encryption and compression processors can offer high speed; secure encryption and compression without affecting the overall performance of the router. In combination, an extra level of security is added - encrypted data is even harder to crack, if it is initially randomized by the compression process. ASIC crypto-based smartcards may be implemented with DES/3DES in CBC mode. CBC introduces a dependency between data blocks, which protects against fraudulent data insertion and replay attacks. In addition, CBC ensures that consecutive repetitive blocks of data do not yield identical cipher text. In short, sn-ASIC-based encryption provides the safest data transmission between networks at the lowest connection cost.

THE ADVANCED ENCRYPTION STANDARD (AES)

On January 2, 1997, the United States' National Institute of Standards and Technology (NIST) announced the initiation of a new symmetric-key block cipher algorithm as the new encryption standard to replace the DES. The new algorithm would be named the Advanced Encryption Standard (AES). Unlike the closed process for the DES, an open call for the AES algorithm was formally made on September 12, 1997. The call stipulated that the AES would specify an unclassified, publicly disclosed symmetric-key encryption algorithm(s); the algorithm(s) must support (at a minimum) block sizes of 128-bits, key sizes of 128, 192 and 256-bits and should have a strength at the level of the triple DES, but should be more efficient than the triple DES. In addition, the algorithm(s), if selected, must be available royalty-free, worldwide.

On August 20, 1998, NIST announced a group of fifteen AES candidate algorithms. The AES is expected to supersede DES (and 3DES) as the standard symmetric crypto-algorithm of the future. The AES crypto-algorithm uses highly advanced data encryption technique developed by Belgian cryptographers Joan Daemen and Vincent Rijmen. After a rigorous multi-year evaluation process, in 2001 the US Government chose AES as the new government standard (FIPS-197) replacing the older DES encryption standard. AES provides far greater security through much larger key size and an improved encryption algorithm. AES also known as Rijndael, is a block cipher with a variable block size and variable key size. The key size and the block size can be independently specified to 128, 192 or 256-bits. This allows AES to encrypt/decrypt 128-bit blocks of data with 3 standard key lengths:

- 128-bit key length that corresponds to approx. 3.4×10^{38} keys
- 192-bit key length corresponding to approx 6.2×10^{57} keys
- 256-bit key length corresponding to approx. 1.1×10^{77} keys

By comparison, DES has approx. 7.2×10^{16} keys. To try and put this into perspective, if we assumed a super-computer could break the DES code in one second, it would take the same super computer 149 thousand billion years to decode an AES key with a 128-bit key length. AES encryption is also particularly well suited for electronic devices such as PCS, IP and mobile phones, PDAs, firewalls and wireless standards, such as the high-speed 802.11g standard.

Positive impact of the AES on applied cryptography: The introduction of the AES is expected to introduce a few positive changes in applied cryptography. First, multiple encryption, such as triple-DES will become unnecessary with the AES: the enlarged and variable key and data-block sizes of 128, 192 and 256 can accommodate a wide spectrum of security strengths for various application needs. Since multiple encryption uses a plural number of keys, the avoidance of using multiple encryption will mean a reduction on the number of cryptographic keys that an application has to manage and hence will simplify the design of security protocols and systems.

Secondly, wide use of the AES will lead to the emergence of new hash functions of compatible security strengths[19,49-51]. In several ways, block cipher encryption algorithms are closely related to hash functions. It has been standard practices that block cipher encryption algorithms are often used to play the role of one-way hash functions. The logging-in authentication protocol of the UNIX operating systems is a well-known example; of a one-way transformation usage of the DES function in the realization of the UNIX password scheme. In practice, hash functions are also commonly used as pseudo-random number functions for generating keys for block cipher algorithms[47]. With the AES's variable and enlarged key and data-block sizes, hash functions of compatible sizes will be needed. However, due to the square-root attack (the birthday attack etc.), a hash

function should have a size, which doubles the size of a block cipher's key or data-block size. Thus, matching the AES's sizes of 128, 192 and 256, new hash functions of output sizes of 256, 384 and 512 are needed. The ISO/IEC are currently in the process of standardizing hash function SHA-256, SHA0384 and SHA-512[4,9,11]. A further advantage of reverting to AES is that its keylength size is of similar size to ECC's thereby giving the two a greater compatibility when used with wireless constrained devices[1].

Finally, as in the case that the DES's standard position had attracted much cryptanalysis attention trying to break the algorithm that these efforts have contributed to the advance of knowledge in block cipher cryptanalysis, the AES as the new block cipher standard will also give rise to a new resurgence of high research interest in block cipher cryptanalysis which will certainly further advance the knowledge in the area.

CONCLUSIONS

In this study various aspect have been explained that are of paramount importance in designing and implementing a secure symmetric cipher crypto-algorithm: We have also effectively shown how to implement DES -- a symmetric key cipher including its security weaknesses. In general, it can be conclude that crypto-algorithms are always the most important core tool in security applications. Hence, the cryptographic algorithms have to be carefully designed, selected and implemented in order to avoid the core cryptography becoming the weakest link in a security solution. DES has been shown to have weaker cryptographic capability and hence care must be taken if one needs to implement DES, otherwise it is advisable to revert to 3DES or to a more secure symmetric cipher like AES. It has been also shown that the introduction of AES and its use will not only greatly enhance the symmetric key cryptosystems but for the first time this ciphers can be integrated into constrained wireless devices like PDAs, cell phones etc. which should greatly enhance the future introduction of 3G cell-phones. The 3G with its advance network technological features of Internet ready allowing for enhanced e-commerce and online banking will greatly place big demand in the design and implementation of secure wireless communication and transactions on-the-fly.

REFERENCES

1. Rabah, K., 2005. Theory and implementation of elliptic curve cryptography. J. Applied Sci., 5: 604-633.
2. Rabah, K., 2005. Implementation of one-time pad cryptography. Inform. Technol. J., 4: 87-95.
3. Rabah, K., 2004. Steganography-The Art of Hiding Data. Inform. Technol. J., 3: 245-269
4. Menezes, A., P. Van Oorschot and S. Vanstone, 1997. Handbook of Applied Cryptography. CRC Press.
5. Kahn, D., 1983. The Codebreakers: The Story of Secret Writing. New York, Macmillan.
6. Feistel, H., 1973. Cryptography and computer privacy. Scientific America, 228: 15-23.
7. Blakley, G.R., 1979. Safeguarding Cryptographic Keys. Proceedings of the National Computer Conference, 1979. Am. Federation of Inform. Processing Soc., 48: 242-268.
8. Bihan, E., 1992. New types of cryptanalytic attacks using related keys. Technical Report 753, Computer Science Department. Technion-Israel Institute of Technology.
9. Christoffersson, P., S.A. Ekahll, V. Fak, S. Herda, P. Mattila, W. Price and H.O. Widman, 1988. Crypto Users Handbook. A Guide for Implementers of Cryptographic Protection in Computer Systems. North Holland: Elsevier Science Publishers.
10. Campbell, C.M., 1979. Design and specification of cryptographic capabilities. IEEE Computer Soc. Magazine, 16: 15-19.
11. Bruce, S., 1993. Applied Cryptography. John Wiley and Sons New York.
12. Rabah, K., 2004. Data security and cryptographic techniques: A review. Inform. Technol. J., 3:106-132.
13. Data Encryption Standard, 1997. Federal Information Processing Standard (FIPS) Publication 46, National Bureau of Standards, US Department of Commerce, Washington D.C.
14. Smid, M.E. and D.K. Branstead, 1988. The Data encryption standard: Past and future. Proceedings of the IEEE, 76: 550-559.
15. Davis, R.M., 1978. The data security standard in perspective. Computer security and the data encryption standard. National Bureau of Standards Special Publication pp: 500-27.
16. Merkle, R.C., 1978. Secure communication over insecure channels. Communications of the ACM., 21: 294-299.
17. Rivest, R., A. Shamir and L.M. Adleman, 1983. Cryptographic communications systems and method. US Patent 4,404,829,20 Sept. 1983.
18. Diffie, E. and M.E. Hellman, 1966. New directions in cryptography. IEEE Transaction on Information Theory, 22: 644-654.
19. Rabah, K., 2005. Secure Implementing message digest. Authentication and digital signature (In Press).

20. Rabin, M.O., 1979. Digital Signature and public-key functions as intractable as factorization. MIT Laboratory of Computer Science, Technical Report, MIT/LCS/TR-212.
21. ElGamal, T., 1985. A public-key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans. Inform. Theory, 31: 469-472.
22. Rabah, K., 2005. Elliptic Curve elgamal signature and encryption schemes (In Press).
23. Koblitz, N., 1987. Elliptic curve cryptosystems. Mathematics of Computation, 48: 203-209.
24. Bellovin, S.M. and M. Merritt, 1992. Encrypted key exchange: Password-based protocols secure against dictionary attacks. Proceedings of the 1992 IEEE Computer Society Conference on Research in Security and Privacy, pp: 72-84.
25 Shannon, C.E., 1993. Collected Papers: Claude Elmwood Shannon. N.J.A. Sloane and A.D. Wyner, Eds, New York: IEEE Press.
26. Shannon, C.E., 1948. A mathematical theory of communication system. Bell Technical J., 27: 379-423, 623-656.
27. Shannon, C.E., 1949. Communication theory of secrete systems. Bell Technical J., 28: 656-715.
28. Kerkhoff's Principle: (1935-1903). Assume the (Unauthorized) User Knows All Ciphering Procedures, Kerkhoff.
29. Koblitz, N., 1987. A: Course in Number Theory and Cryptography. Springer-Verlag.
30. Maurer U.M. and J.L. Massey, 1990. Perfect local randomness in pseudo-random sequences. Advances in cryptology-CRYPTO'89 proceedings, Berlin: Springer-Verlag, pp: 110-112.
31. Rivest, R.L., 1995. The RC5 Encryption Algorithm. RSA Laboratories, CryptoBytes, 1: 1
32. Brown, L. and J. Seberry, 1990. On the design of permutation in DES type cryptosystems. Advances in Cryptology: Proceedings of EUROCRYPT 89, Berlin, Pringer-Velag, pp: 696-705.
33. Department of Commerce/National Institute of Standards and Technology (US)., 1997. Announcing request for candidate algorithm nominations for the Advanced Encryption Standard (AES). World Wide Web. http://csrc.nist.gov/encryption/aes/pre-round1/aes_9709.htm.
34. Feistel, H., 1974. Block cipher cryptographic system. U.S. Patent #3,798,359,19, Mar 1974.
35. Hodges, A., 1983. Alan Turing: The Enigma. Simon and Schuster.
36. Welchman, G., 1982. The Hut Six Story: Breaking the Enigma Codes. McGraw-Hill.
37. Smith, J.L., 1971. The design of lucifer. A cryptographic device for data communications. IBM Research Report, RC3326.
38. With Data Encryption, Scents Are Safe at IFF, 1980. Computerworld 14, No. 21, 95.
39. Kroph, T., J. Frosl, W. Beller and T. Giesler, 1990. A hardware implementation of modified DES algorithm. Microprocessing and Microprogramming, 30: 59-66.
40. Davio, M., Y. Desmedt, J. Goubert, F. Hoornaert and J.J. Quisquater, 1985. Efficient hardware and software implementation of DES. Advances in Cryptology-EUROCRYPT'84 Proceedings, Berlin: Springer-Verlag, pp: 62-73.
41. Biham, E. and A. Shamir, 1993. Differential Cryptanalysis of the Data Encryption Standard. Springer-Verlag, New York.
42. Matsui, M., 1994. The First Experimental Cryptanalysis of the Data Encryption Standard. In: Y.G. Demesdt, Ed., Advances in Cryptology-crypto. Springer-Verlag, New York, 94:1-11.
43. Dvio, M., Y. Desmedt, M. Fosseprez, R. Govaerts, J. Hulsbrosch, P. Neutjens, P. Piret, J.J. Quisquater, J. Vandewalle and S. Wouters, 1984. Analystical characteristics of the data encryption standard. Advances in cryptology: Proceedings of Crypto 83, Plenum Press, pp: 171-202.
44. Mahir, B., R. Guérin and P. Rogaway, 1995. XOR MACs: New methods of message authentication using block cipher. Accepted to Crpto '95.
45. Mihi, B., J. Killian and P. Rogaway, 1990. The Security of Cipher Block Chaining. In: Yvo, G.D. (Ed.) Advances in Cryptology-Crypto Springer-Verlag, New York, pp: 341-358.
46. Lexar Corporation, 1976. An evaluation of the DES.
47. Cryptographic Algorithms for Protection of Computer Data During Transmission and Dormant Storage, 1973. Federal Register 38, No. 93.
48. Davies, D.W. and W.L. Price, 1989. Security for Computer Networks: An Introduction to Data Security in Teleprocessing and Electronics Funds Transfer, 2nd Edn., John Wiley and Sons, New York.
49. Merkle, R., 1990. One Way Hash Function and DES. In: Brassard, G., (Ed.), Advances in Cryptology: Proceedings of Crypto'89, Springer-Verlag, New York, pp: 428-446.
50. Gene, T., 1992. Message authentication with one-way hash function. ACM Computer Communications Review, 22: 2938.
51. Kaliski, B. and M. Robshaw, 1995. Message Authentication with MD5. Cryptobytes, The Technical Newsletter of RSA Laboratories, Spring.