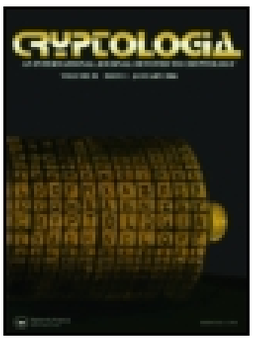

CRYPTOLOGIA

ISSN: 0161-1194 (Print) 1558-1586 (Online) Journal homepage: https://www.tandfonline.com/loi/ucry20

A SIMPLIFIED DATA ENCRYPTION STANDARD ALGORITHM

Edward F. Schaefer

A SIMPLIFIED DATA ENCRYPTION STANDARD ALGORITHM

Edward F. Schaefer

ADDRESS: Department of Mathematics, Santa Clara University, Santa Clara CA 95053 USA. eschaefer@scuacc.scu.edu.

ABSTRACT: In this paper we describe a method of teaching the Data Encryption Standard algorithm in an undergraduate cryptology course. We present a simplified version of the Data Encryption Standard algorithm with all parameters reduced as much as possible. This makes the inner workings of the algorithm accessible to undergraduates. Once the simplified algorithm has been explained to a class, it is easier to explain the real one. We suggest class discussions and homework based on this simplified algorithm.

KEYWORDS: Data Encryption Standard, cryptology course

1. INTRODUCTION

In 1973 and 1974, the National Bureau of Standards solicited data security systems from business and academia. International Business Machines Corporation submitted the Data Encryption Standard algorithm (DES) and it was subsequently accepted and published in 1975 [4]. DES is widely used in business in the United States; PIN numbers, phone conversations and many other types of data are encrypted with DES.

Cryptology is becoming a popular undergraduate course in mathematics departments. Current textbooks appropriate for an undergraduate course on cryptology do not discuss DES [see 1, 3, 5]). There are two possible reasons for this. The security of many popular cryptosystems is based on the difficulty of solving certain number theoretic problems. The problem which is most apparently involved in breaking DES is not viewed to be as elegant as those behind other cryptosystems like RSA, discrete log for finite fields and for elliptic curves, and knapsack. The obvious way of trying to break DES requires finding simultaneous solutions to many non-linear equations in many unknowns over $\mathbf{F}_2$. The other reason must be that the DES algorithm itself is very complicated and can not be explained as neatly as the above-named cryptosystems. This is the problem we will address in this paper.

Below we will give a simplified version of the DES algorithm. The author prepared this algorithm for an upper-divison undergraduate cryptology course taken mostly by mathematics and computer science majors. This version is meant to be educational as opposed to being a secure algorithm. It has similar properties to DES with much smaller parameters. After describing this simplified DES, we will explain how to expand the parameters to see how the the real DES algorithm works.

The author has had success in the classroom using this method of explaining DES. The simplified algorithm is presented on a handout as it is in sections 2 and 3. In parallel, we run through the algorithm in the encryption direction with a chosen key and plaintext. The students work through this example in the margins of the handout. For homework, the students are given a key and ciphertext and are asked to find the plaintext by hand, in order for them to get the flavor of the inner workings of DES. There will be more on this homework assignment at the end of section 3.

By the time the class gets to this algorithm, we have already discussed modular arithmetic since there is addition modulo 2 in this algorithm. We have covered elementary cryptosystems involving transposition and substitution ciphers since the DES algorithm alternates these. We have discussed frequency analysis so that the students understand the consequences of using the blocksize that DES operates on. We have also covered running time analysis and have discussed P and NP problems since solving many non-linear equations in many unknowns over $\mathbf{F}_2$ is a problem which is NP-hard. We discuss DES before public key cryptography since public key cryptography can be used to exchange DES keys.

2. KEYS

For the simplified version of DES, two users have a shared, secret 10 bit key that they have agreed upon ahead of time. From that key they make subkeys. Let the agreed upon 10 bit key be $(k_0k_1 \dots k_9)$ where $k_i \in \{0,1\}$. There are two given permutations, $P10$ = (2,4,1,6,3,9,0,8,7,5) and $P8$ = (5,2,6,3,7,4,9,8), and a shifting sequence (1,2).

First apply $P10$ (which is only used once) to the key and get

$$(k_2k_4k_1k_6k_3k_9k_0k_8k_7k_5) = (s_0s_1s_2s_3s_4s_5s_6s_7s_8s_9)$$

where $k_2 = s_0, \dots, k_5 = s_9$. Break this string into two substrings consisting of the first 5 and the last 5 bits and shift each substring to the left by 1, since 1 is the first number in the shifting sequence. So

$$(s_0s_1s_2s_3s_4)(s_5s_6s_7s_8s_9)$$

gets shifted to

$$(s_1s_2s_3s_4s_0s_6s_7s_8s_9s_5) = (t_0t_1t_2t_3t_4t_5t_6t_7t_8t_9).$$

Now apply $P8$ to pick out and permute 8 of the 10 bits

$$(t_5t_2t_6t_3t_7t_4t_9t_8) = (k_{1,0}k_{1,1}k_{1,2}k_{1,3}k_{1,4}k_{1,5}k_{1,6}k_{1,7}).$$

This is key 1.

Break the last 10 bit string into two substrings $(t_0t_1t_2t_3t_4)(t_5t_6t_7t_8t_9)$ and shift each left by 2, since 2 is the second number in the shifting sequence. We get

$$(t_2t_3t_4t_0t_1t_7t_8t_9t_5t_6) = (u_0u_1u_2u_3u_4u_5u_6u_7u_8u_9).$$

Now apply $P8$ to pick out and permute 8 of the 10 bits

$$(u_5u_2u_6u_3u_7u_4u_9u_8) = (k_{2,0}k_{2,1}k_{2,2}k_{2,3}k_{2,4}k_{2,5}k_{2,6}k_{2,7}).$$

This is key 2.

3. THE SIMPLIFIED ENCRYPTION ALGORITHM

With this simplified DES, the plaintext messages are broken into blocks, each an 8 bit string like (10111101). There are then 2^8 possible plaintext blocks. We encrypt one block at a time. Identical blocks are encrypted identically for a given key.

Encryption is by the composition of five maps, $IP^{-1} \circ \Pi_{T_2} \circ \Theta \circ \Pi_{T_1} \circ IP$, which will be described below. All additions are bit-by-bit modulo 2 additions. As an example

$$\begin{array}{r} (10101) \\ +\ (11001) \\ \hline =\ (01100). \end{array}$$

This is the same as using XOR on every corresponding pair of bits.

There are three kinds of maps.

i) The map IP is the initial permutation; it is (1,5,2,0,3,7,4,6). Let $m_i \in \{0,1\}$. Then

$$\begin{aligned} IP(m_0m_1m_2m_3m_4m_5m_6m_7) &= (m_1m_5m_2m_0m_3m_7m_4m_6) \\ &= (n_0n_1n_2n_3n_4n_5n_6n_7). \end{aligned}$$

We have IP^{-1} =(3,0,2,4,6,1,7,5). So

$$\begin{aligned} IP^{-1}(n_0n_1n_2n_3n_4n_5n_6n_7) &= (n_3n_0n_2n_4n_6n_1n_7n_5) \\ &= (m_0m_1m_2m_3m_4m_5m_6m_7). \end{aligned}$$

ii) The map Θ switches the first four bits with the last four bits;

$$\Theta(m_0m_1m_2m_3m_4m_5m_6m_7) = (m_4m_5m_6m_7m_0m_1m_2m_3).$$

Note that Θ^2 is the trivial map, so $\Theta^{-1} = \Theta$. The maps IP and Θ are each called transposition maps since they simply permute the bits.
iii) Let us define Π_T, where T is some arbitrary map (not necessarily one-to-one) from 4 bit strings to 4 bit strings. We let

$$\Pi_T(X, X') = (X + T(X'), X')$$

where X and X' are 4 bit strings. The map Π_T^2 is trivial, because applying it twice is adding $T(X')$ twice to X and modulo 2 that is the same as adding (0000). So $\Pi_T^{-1} = \Pi_T$. As an example, say the string is (10111101) and T is some function for which $T(1101) = (1110)$. Then $\Pi_T(10111101) = (01011101)$ since (1011) + (1110) = (0101).

Let us describe the map T_1. The input is a 4 bit number $(n_4n_5n_6n_7)$ with $n_i \in \{0, 1\}$. We index them 4 through 7 because we apply T_i to bits 4 through 7 when applying Π_{T_i} to a string of 8 bits. Make a diagram

$$\begin{array}{c|cc|c} n_7 & n_4 & n_5 & n_6 \\ n_5 & n_6 & n_7 & n_4 \end{array}$$

and add key 1 in the following way

$$\begin{array}{c|cc|c} n_7 + k_{1,0} & n_4 + k_{1,1} & n_5 + k_{1,2} & n_6 + k_{1,3} \\ n_5 + k_{1,4} & n_6 + k_{1,5} & n_7 + k_{1,6} & n_4 + k_{1,7}. \end{array}$$

We will rename these 8 bits

$$\begin{array}{c|cc|c} p_{0,0} & p_{0,1} & p_{0,2} & p_{0,3} \\ p_{1,0} & p_{1,1} & p_{1,2} & p_{1,3}. \end{array}$$

There are two given S-boxes, $S[0]$ and $S[1]$, shown below. We have labelled the rows and columns 0 to 3.

$$S[0] = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 1 & 0 & 3 & 2 \\ 3 & 2 & 1 & 0 \\ 0 & 2 & 1 & 3 \\ 3 & 1 & 3 & 2 \end{bmatrix} \end{array} \quad S[1] = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 0 & 1 & 2 & 3 \\ 2 & 0 & 1 & 3 \\ 3 & 0 & 1 & 0 \\ 2 & 1 & 0 & 3 \end{bmatrix} \end{array}$$

Consider $(p_{0,0}p_{0,3})$ and $(p_{0,1}p_{0,2})$ as base-2 numbers between 0 and 3. In S-box $S[0]$ look in row $(p_{0,0}p_{0,3})$ and column $(p_{0,1}p_{0,2})$, and find the entry, which is a number between 0 and 3. Write that number as a base 2 number (q_0q_1). If $(p_{0,0}p_{0,3}) = (00)$ and $(p_{0,1}p_{0,2}) = (10)$ then we look in row 0 and column 2 and find the entry 3 so $(q_0q_1) = (11)$.

Similarly in S-box $S[1]$ look in row $(p_{1,0}p_{1,3})$ and column $(p_{1,1}p_{1,2})$ and find the entry between 0 and 3. Write it as a base 2 number (q_2q_3). Now concatenate them and you have $(q_0q_1q_2q_3)$, a 4 bit string. There is one more given permutation $P4 = (1, 3, 2, 0)$. Apply $P4$ and get $(q_1q_3q_2q_0)$. That is the output of the map T_1. So $T_1(n_4n_5n_6n_7) = (q_1q_3q_2q_0)$. Recall that this is just part of doing Π_{T_1}. So if the string, during encryption, and after the initial permutation, is $(n_0n_1n_2n_3n_4n_5n_6n_7)$ then Π_{T_1} will take that to

$$(n_0 + q_1, n_1 + q_3, n_2 + q_2, n_3 + q_0, n_4, n_5, n_6, n_7).$$

T_2 is identical except that we use key 2 instead. The $S[0]$, $S[1]$ and $P4$ are the same. It may seem unsafe to leave the last 4 bits alone, but recall Θ comes between Π_{T_1} and Π_{T_2}.

The map Π_T is called a substitution map since it replaces bits by bits, and does not simply permute them. An encryption algorithm that alternates transposition maps with substitution maps is called a product cipher. A pair of substitution and transposition maps is called a round. This algorithm runs 2 rounds. Decryption is by $(IP^{-1} \circ \Pi_{T_2} \circ \Theta \circ \Pi_{T_1} \circ IP)^{-1}$. Now recall that $(f \circ g)^{-1} = g^{-1} \circ f^{-1}$. So decryption is by $IP^{-1} \circ \Pi_{T_1} \circ \Theta \circ \Pi_{T_2} \circ IP$.

For homework, the students are told to decrypt the string (10100010) using the key (0111111101) by hand. Then they are told to decode the first four bits of the plaintext string to a letter and the second four bits to another letter where we encode $A - P$ in base 2, i.e. $A = 0000$, $B = 0001$, ...,$P = 1111$. The resulting plaintext message is 'OK'. As a midway check, they are told that after the application of Θ, that the string should be (00010011). Some of the students implemented this algorithm as a computer program for both encryption and decryption on their own.

4. ANALYSIS OF SIMPLIFIED DES

Let us call the plaintext $(p_0 \dots p_7)$, and the ciphertext output $(c_0 \dots c_7)$, that one would get using the key $(k_0 \dots k_9)$. Then each c_i is a polynomial function f_i of the p_j's and the k_j's modulo 2. If the enemy has some pair of corresponding plaintext and ciphertext, then he can try to solve for the key. The only unknowns are the k_i's. With the key he could decrypt future ciphertext. There may be several possible solutions, but cryptanalysis is often a process of narrowing down the possibilities. In this case, we have 8 non-linear equations in 10 unknowns. All of the permutations and additions are linear maps. The non-linearity comes from the S-boxes. Let us consider how they operate. For clarity, let us rename $(p_{0,0}, p_{0,1}, p_{0,2}, p_{0,3}) = (a, b, c, d)$ and $(p_{1,0}, p_{1,1}, p_{1,2}, p_{1,3}) = (w, x, y, z)$. Then the operation of the S-boxes can be computed with the following equations

$$
\begin{aligned}
q_0 &= abcd + ab + ac + b + d \\
q_1 &= abcd + abd + ab + ac + ad + a + c + 1 \\
q_2 &= wxyz + wxy + wyz + wy + wz + yz + w + x + z \\
q_3 &= wxz + wyz + wz + xz + yz + w + y
\end{aligned}
$$

where all additions are modulo 2. Alternating the linear maps with these non-linear maps leads to very complicated polynomial expressions for the ciphertext bits.

A polynomial equation in 10 unknowns over $\mathbf{F}_2$ can have 2^{10} possible terms, each with a coefficient of 0 or 1. On average then, one might expect the 8 equations to have 2^9 terms each. The interested student might be encouraged to try to find these equations with a symbolic processor for a given plaintext/ciphertext pair. He would undoubtedly give up at some time during the process if the symbolic processor does not give up first.

At this point, one could ask the class to suggest and discuss various methods of breaking this simplified DES. Here are a few suggestions in case the students do not come up with them themselves.

1. If the enemy has several plaintext/ciphertext pairs, he could try to solve the equations for the key.

2. If the enemy has several plaintext/ciphertext pairs, he could try all keys and see which ones work.

3. If the enemy has several ciphertexts, he could decrypt using all possible keys to see which give sensible plaintexts.

4. If the enemy has several ciphertexts, he can do a frequency analysis.

The class can discuss such methods again after the presentation of the parameters of the real DES algorithm.

5. THE REAL DES

The actual DES encryption algorithm operates on blocks of plaintext that are 64 bits long and so there are 2^{64} possible plaintext blocks. The encryption is by

$$IP^{-1} \circ \Pi_{T_{16}} \circ \Theta \circ \Pi_{T_{15}} \circ \Theta \circ \ldots \circ \Theta \circ \Pi_{T_1} \circ IP.$$

So DES runs 16 rounds. The key is 56 bits long and comes with 8 extra parity-check bits. The subkeys have 48 bits. So instead of permutations of length 10 and 8 for the keys they are of length 56 and 48. There are 16 subkeys since there are 16 T_i's. The permutation of length 56 is used initially and the permutation of length 48 is used 16 times to pick out the subkeys. The shifting sequence is $(1,1,2,2,2,2,2,2,1,2,2,2,2,2,2,1)$. The initial permutation IP is a permutation of the 64 bits. Now instead of T_i acting on $(n_4 n_5 n_6 n_7)$ it acts on $(n_{32} \ldots n_{63})$. The diagram that you put those in actually looks like

n_{63}	n_{32}	n_{33}	n_{34}	n_{35}	n_{36}
n_{35}	n_{36}	n_{37}	n_{38}	n_{39}	n_{40}
$\vdots$			$\vdots$		$\vdots$
n_{59}	n_{60}	n_{61}	n_{62}	n_{63}	n_{32}

which has 8 rows and 6 columns, hence the 48 bit subkeys. Since there are 8 rows in the diagram, there are 8 S-boxes $S[0], \ldots, S[7]$. Each S-box has 4 rows and 16 columns, since $(n_{63}n_{36})$ can represent 4 numbers and $(n_{32}n_{33}n_{34}n_{35})$ can represent 16 numbers. The last part of the map T_i is a permutation of length 32 instead of 4, half the block length. All of the permutations and S-boxes are available to the users and can also be found in [3].

There are four modes of operation on a DES chip. We have just described the standard encryption mode or *Electronic Code Book* mode. It is the mode that the other three are based on. For a description of all four modes see

REFERENCES

1. Beutelspacher, A. 1994. *Cryptology.* Washington : The Mathematical Association of America.

2. Brassard, G. 1988. *Modern Cryptology: A tutorial.* Lecture Notes in Computer Science **325**, New York: Springer-Verlag.

3. Koblitz, N. 1987. *A course in Number Theory and Cryptography.* New York: Springer-Verlag.

4. Konheim, A. G. 1981. *Cryptography: A primer.* New York: John Wiley & Sons, Inc.

5. Rosen, K. 1993. *Elementary Number Theory and its Applications.* Reading Massachusetts: Addison Wesley.

BIOGRAPHICAL SKETCH

Edward Schaefer received BS degrees in mathematics and psychology at U. C. Davis in 1984. He spent 1984-1986 as a research assistant in the Stanford University department of psychology. He received his PhD in mathematics at U. C. Berkeley in 1992. Currently, he is an assistant professor of mathematics at Santa Clara University where he teaches a cryptology course. His main research interest is arithmetic geometry. He is a member of the AMS, the MAA and ΠME and has recently received a National Security Agency Young Investigators Grant.