Dynamic Symmetric Cipher (DySC)

Manimozhi K., Kalaichelvi V., Meenakshi P., Poorinima M., & Sumathi A.
Asst. Professor, Department of CSE, SASTRA University, Kumbakonam
kmanimozhi77@src.sastra.edu

Abstract : This paper proposes a Dynamic Symmetric block Cipher for encryption and decryption. It uses varying number of keys in each round with varying key length. Depends on the application the key length can be chosen. To increase the security, it can be enhanced into n number of levels (Rounds). It is a simple algorithm parameterized by the block size, the number of rounds, and key length. These parameters can be adjusted to meet different goals for security and performance.

Key words – Encryption; Decryption; Substitution; Flip; and Transposition..

1.0 Introduction

Security is the primary concern of all those people who deal with activities, which involve protection of risk. The branch of science "cryptography" is concerned with the security of information, developed in the hands of military people and it was nurtured by them for quite long time as their private property. For this reason many algorithms are developed for encryption and decryption which provide high security. All these algorithms are kept open to the public and the secrecy of the algorithm lies entirely in the key. This paper stands different that the development of our own algorithm addresses the user needs in specific, there by more flexibility.

2.0 Proposed DySC Algorithm

2.1 Encryption **Dynamic Parameters**

Fig 1. shows the overall structure of DySC encryption. The inputs to the encryption algorithm are N_b, K_i, K_{sw}, r, K_{max} and Plain text. Encryption of data is done in "rounds". The general setup of each round is the same, except for the number of subkeys. DySC encryption uses r number of rounds.

N_b	Key length	Block length	Rounds	K_{max}
2^1 - 2^6	2^5 - 2^{10}	2^2 - 2^7	Minimum 2	$\leq N_b$

2.1.1. Operations in the round function

a. Substitution

Plain text is the combination of the ASCII characters. The ASCII characters are divided into N_b blocks, starting from 0 to N_b-1. Each block contains $128/N_b$ characters. The division of blocks is represented in Table 1. The steps are given below to create the substitution table[1].

Sample parameters

N_b=32 K_{sw} = 1423451235

K_i =	00	30	28	26	24	22	20	18	16	14	12	10
	08	06	04	02	01	03	05	07	09	11	13	15
	17	19	21	23	05	27	29	31				

Step 1:

Take the Key K_i.

Step 2:

The key K_i is permuted by using K_{sw}. Thus we get the key K_p. $K_p = K_{sw}(K_i)$

K_p=	00	29	21	13	05	28	20	12	04	25	17	09
	01	24	16	08	02	10	18	26	03	11	19	27
	06	14	22	30	07	15	23	31				

Step 3:

Using the Encryption key, we have to construct the substitution table. Consider the second number in K_p, that is 29 and it refers to the block 29 in Table 1. This block contains the characters {t,u,v,w}, The equivalent

ASCII values {116,117,118,119} are obtained and replace these characters by decimal 0,1,2,3 in ASCII table. This procedure is continued for the remaining Numbers in K_p. Thus the Substitution table is formed.

Table 1. ASCII Table

Block no	Characters	Block no	Characters
0		17	D E F G
1		18	H I J K
2	\n	19	L M N O
3		20	P Q R S
4		21	T U V W
5		22	X Y Z [
6		23	\ ] ^ _
7		24	` a b c
8	Blank ! " #	25	d e f g
9	$ % & '	26	h i j k
10	( ) * +	27	l m n o
11	, - . /	28	p q r s
12	0 1 2 3	29	t u v w
13	4 5 6 7	30	x y z {
14	8 9 : ;	31	| } ~
15	< = > ?		
16	@ A B C		

Table 2. Substitution Table

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	64	65	66	67	80	81	82	83	96	97
1	98	99	112	113	114	115	60	61	62	63
2	44	45	46	47	28	29	30	31	12	13
3	14	15	72	73	74	75	88	89	90	91
4	104	105	106	107	120	121	122	123	52	53
5	54	55	36	37	38	39	20	21	22	23
6	4	5	6	7	68	69	70	71	84	85
7	86	87	100	101	102	103	116	117	118	119
8	56	57	58	59	40	41	42	43	24	25
9	26	27	8	9	10	11	76	77	78	79
10	92	93	94	95	108	109	110	111	124	125
11	126	127	48	49	50	51	32	33	34	35
12	16	17	18	19	0	1	2	3		

b. Flipping

For each bit of the input, if the corresponding bit of the flipping key is 0, the corresponding output bit will be the same as the input bit. If the corresponding bit of the flipping key is 1, the corresponding output bit will be the complement of the input bit. In reconstructing the original input, the output of the flipping operation is flipped against the same flipping key [1,2].

c. Transposition

Transposition techniques systematically transpose the positions of the plain text elements. In DySC to transpose the elements Folding techniques are used.

Folding Techniques

Three types of folding techniques are used to do the transposition. The nearest square root value 'n', for the length of the plain text block is found and a matrix with order n x n is formed with the plain text block. Then, all the three types of folding are applied[2].

1. Horizontal Folding
2. Vertical Folding
3. Diagonal Folding

2.1.2 Encryption Algorithm

```
DySC_Encrypt(r, Nb, Kmax, Ki)
{
    Kp =Permute(Ki)
    construct_substitution_table(Kp)
    b=0
    while(!EOF(pfile)){
    Pb=read(pfile,Nb*2)
    for i=1 to r{
            for each char c of Pi-1
                    substitute(c)
            Key(Kp)
```

```
    for j=1 to NK[i] {
                K_j=generate_sub_key(K_p)
                C_i = flip(P_i-1,j,SK[i][j])
                }
                transpose(C_i)
                P_i=C_i
        }
        b=b+1
        }
    }
    Permute(K_i)
    {
        Read Ksw
        for i=1 to len(K_i)
                K_p[i]=swap(K_sw , K_p[i])
        return K_p
    }
    construct_substitution_table(K_p)
    {
        Divide ASCII table into N_b blocks
        Var sub[13][10],fill=0
        for i=1 to len(K_p)
           {
                B=Block[K_p[i]]
                for j=1 to len(B)
                  {
                        row=first(B[j])
                        col=last(B[j])
                        sub[row][col]=fill
                        fill=fill+1
                }
        }
}
substitute(P)
{
        for i=1 to len(P) {
                row=first(P[i])
                col=last(P[i])
                C[i]=sub[row][col]
        }
        return C
}
Transpose()
{       form_matrix()
        fill_matrix(C_i)
        C_i= H_fold(V_fold(D_fold(C_i)))
}
```

2.2 Decryption

The inputs to the decryption algorithm are N_b, K_i, K_{sw}, r, K_{max} and Cipher text. For decryption the reverse process of encryption is done based on the algorithm given. For substitution the inverse table(Table 3) is used.

2.2.1 Decryption Algorithm

```
DySC_Decrypt(r)
{
        K_p =Permute(K_i)
        construct_rev_substitution_table(K_p)
        b=0
        while(!EOF(cfile))
```

```
        {
          Cb=read(cfile,Nb*2)
          for i=r to 1
        {
                    for each char c of Ci
                              substitute(c)
                    for j=1 to NK[i]  {
                    Kj=generate_sub_key(Kp)
                    Pi = flip(Ci,SK[i][j])
                    }
                    transpose(Pi)
                    Ci-1=Pi
          }
          b=b+1
          }
}
```

2.3. Key Schedule

In cryptography, the so-called product ciphers are a certain kind of ciphers, where the (de-)ciphering of data is done in "rounds". The general setup of each round is the same, except for some hard-coded parameters and a part of the cipher key, called a subkey. Key schedule is an algorithm that, given the key, calculates the subkeys for these rounds. In this algorithm for every i^{th} round the Encryption key is divided into NK_i sub keys.

```
Key(Kp, r, Kmax)
{
          p=nearestprime(r^Kmax)
          for i=1 to r
          {
                    NK[i]=p%(Kmax+1)
                    p=nextprime(p)
          }
}
generate_sub_key(Kp)
{
          for i=1 to r
          for j = 1 to NK[i]
                    {
                    k=shift(flip(slide(Kj),Kj))
                    SK[i][j]=k
                    }
}
```

Table 3. Inverse Substitution Table

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	124	125	126	127	60	61	62	63	92	93
1	94	95	28	29	30	31	120	121	122	123
2	56	57	58	59	88	89	90	91	24	25
3	26	27	116	117	118	119	52	53	54	55
4	84	85	86	87	20	21	22	23	112	113
5	114	115	48	49	50	51	80	81	82	83
6	16	17	18	19	0	1	2	3	64	65
7	66	67	32	33	34	35	96	97	98	99
8	4	5	6	7	68	69	70	71	36	37
9	38	39	100	101	102	103	8	9	10	11
10	72	73	74	75	40	41	42	43	104	105
11	106	107	12	13	14	15	76	77	78	79
12	44	45	46	47	108	109	110	111		

2.3.1 ***Operations used in Key Schedule***

a. Slide

'Slide' is an operation used in the Key scheduling algorithm to generate the sub keys. It takes encryption key and NK_i as inputs.

Length(slider)=Length(binary(Finalbit position))

Table 4. Sub Key

K_p	1	0	0	1	0	1	0	1	0	0	0	1	1	1	0	0
Bit Position	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Derived Key	0	1	1	1	1	1	0	1	1	1	1	0	1	0	1	0

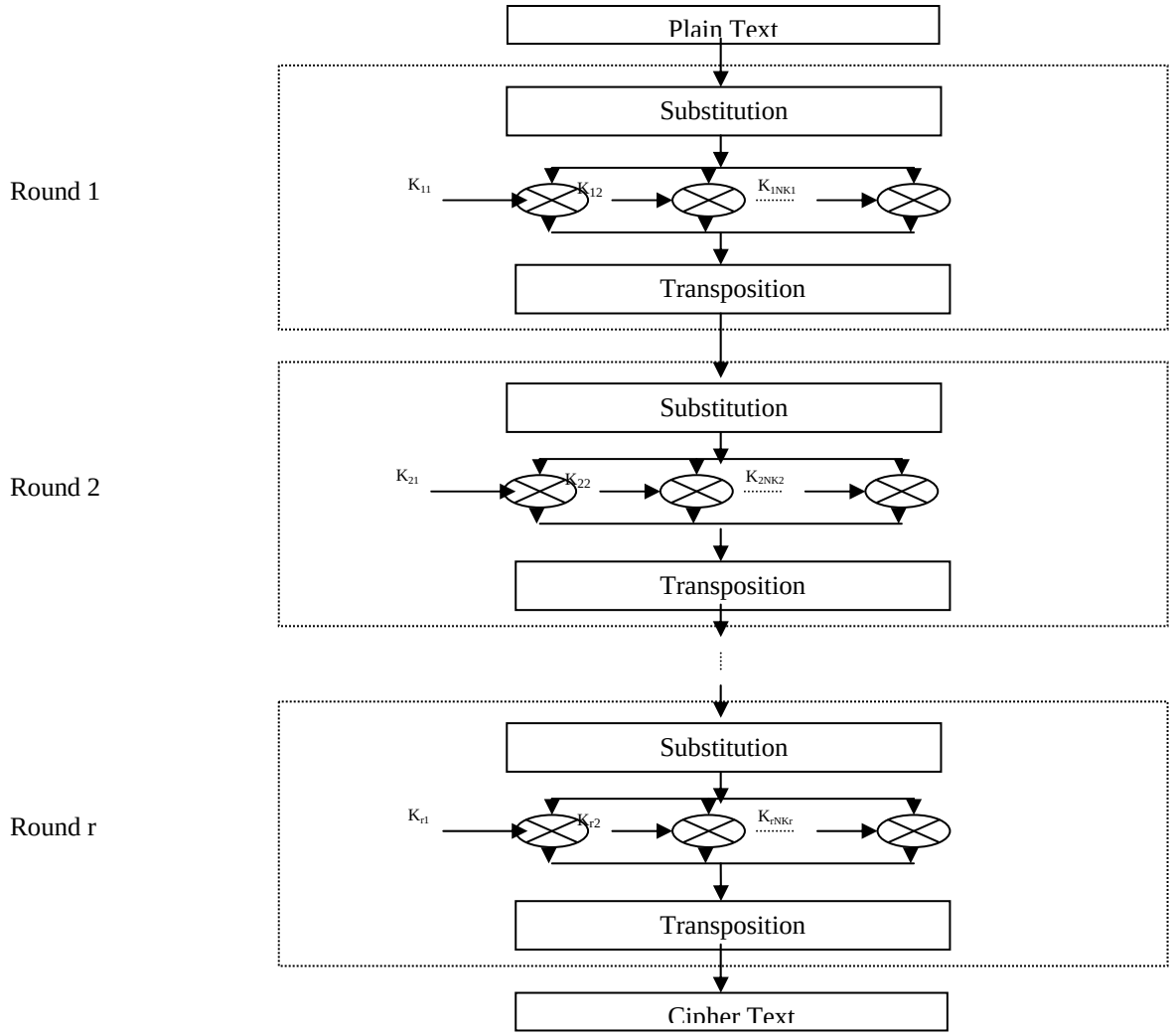


Figure 1. Encryption

Example **:** Binary equivalent of 15 = 1111_2 . The length of slider is four. To derive a sub key of length 16, 16 bits of K_p are taken. The first four bits are taken (1001_2) from K_p. '9' is the equivalent decimal value. In the derived key the 9^{th} bit position is filled with '1'. One bit forwarded circularly and the same procedure is repeated to fill the remaining positions, shown in the table 4 and table 5. The positions which are not filled with 1s are filled with 0s.

b. Flip

$K_f = \text{flip}(K_p , K_s)$

c. Circular Left Shift

Left circular shift is done. The number of bits to be shifted (N_s) is the number of sub keys used in the round (NK_i). $N_s = NK_i$

$$m = \text{Round}(\text{Floor}(K_{pi}/NK_i), 8)$$

Table 5. Slider

Bit position	Bits	Decimal Equivalent
0 1 2 3	1001	9
1 2 3 4	0010	2
2 3 4 5	0101	5
3 4 5 6	1010	10
4 5 6 7	0101	5
5 6 7 8	1010	10
6 7 8 9	0100	4
7 8 9 10	1000	8
8 9 10 11	0001	1
9 10 11 12	0011	3
10 11 12 13	0111	7
11 12 13 14	1110	14
12 13 14 15	1100	12
13 14 15 0	1001	9
14 15 0 1	0010	2
15 0 1 2	0100	4

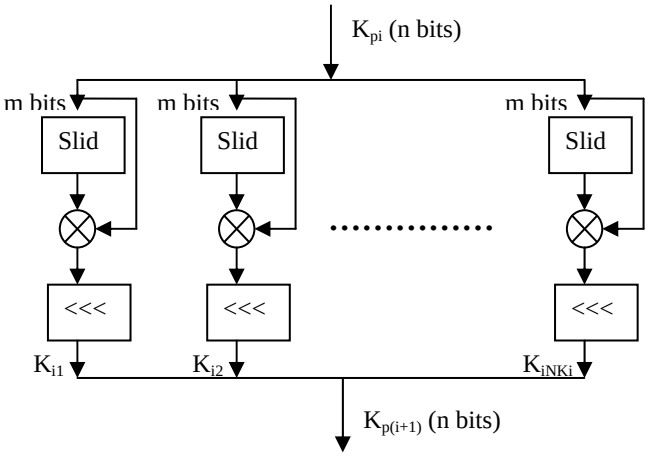


Figure 2 Sub Key Generation

3.0 Analysis of the Proposed Algorithm:

Brute Force Attack

To hack into the DySC encryption / decryption algorithms using the brute force approach, one needs to guess the flipping key, the Substitution function, the permutation function and the encryption level.

Encryption Level

The last piece of the secret information is the encryption level. It is a positive integer. The higher the encryption level is, the more secure the algorithm is. However, we should be cautious with large values of the encryption level since the increasing of the encryption level is proportional to the decreasing of the Encryption / decryption speed.

Key generation:

The main strength of the DySC algorithm is Sub-key generation. DySC uses varying number of keys in each round with varying key length. Depends on the application the key length can be chosen.

4.0. Conclusion

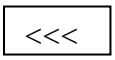
The DySC algorithm takes plain text of length n bits, a key of length n bits and it produces an output of length n bits. This is dynamic, since it uses plain text, key of variable length and different numbers of keys are used in each round. Depends on the level of security needed, number of rounds can be implemented. Since it satisfies the principles of block ciphers and Feistel Cipher structure, it is a symmetric block cipher. If there is a need for the applications to expose its inputs and its encrypted forms of the inputs, then it should use the DySC encryption / decryption algorithm instead. It is also a simple, fast, and fairly secure algorithm.

GLOSSARY

P, PT	-	Plain text
C, CT	-	Cipher text
P_i	-	Plain text in i^{th} round
C_i	-	Cipher text in i^{th} round
K_i	-	Initial key
K_p	-	Permuted key
K_{pi}	-	Key used at i^{th} round
K_{sw}	-	Swapping key
r	-	Number of rounds
n	-	Length of the key

ICT - Intermediate Cipher Text
IPT - Intermediate Plain Text
N_b - Number of Blocks
N_c - Number of Characters
B - Block
K_{max} - Maximum number of keys
N_p - Nearest prime number
NK_i - Number of keys used in i^{th} round
SK[][] - Array of sub keys

 - Flipping



- Left Rotation

REFERENCES

[1] V.Kalaichelvi, N.Subha & M.Venkatesh, “H20 Algorith for Secure Communication”, Recent trends in Computational Mathematics - IJMS, Volume 5, No?:2, Dec’ 2006, pp237 – 247.
[2] Peter Phuong Vo, Chau Maggie Vo “FMS and FMSE encryption/decryption algorithms”, Proceedings of the 3rd annual conference on Information security curriculum development, Kennesaw, Georgia, ACM, New York, USA. 2006, Pages : 142 – 146.
[3] Kalaichelvi. V RM.Chandrasekaran, "Fold Head Mode Cryptography" Narosa Publisher 2005, PP 32 – 36
[4] Kalaichelvi. V RM.Chandrasekaran “AnAlgorithm for Secure Communication" Allied Publishers Pvt. Ltd, 2006, PP3 – 8
[5] Bruce Schenier " Applied Cryptography" John Wiley Sons Inc, 2001
[6] William Stallings "Cryptography and Network Security", Pearson Edn Pvt.Ltd.
[7] Richard Smith "Internet Cryptography", Pearson Edn Pvt.Ltd