Four Way Encryption Method for Secure Message Coding by use of CBC, Merkle-Hellman, Randomizers and Discrete Logarithmics

B. Padhmavathi
Department of CSE
SRM University
Vadapalani Campus
Chennai – 600026,
Tamil Nadu, India

Arghya Ray
Final year-Department of CSE
SRM University
Vadapalani Campus
Chennai-600026,
Tamil Nadu, India

Alisha Anjum
Pre-final year, Dept. of CSE
SRM University
Vadapalani Campus
Chennai-600026,
Tamil Nadu, India

ABSTRACT
The Cipher Block Chaining (CBC) and the Merkle-Hellman algorithm are two different cryptosystems for encrypting messages. But using these cryptosystems separately has many drawbacks. However there are several ways of getting rid of these drawbacks.

This paper demonstrates how to use the combination of both the cryptographic algorithms along with randomized vectors, keys and super-increasing sequence to encrypt messages. The encrypted message is stored in secret buffers and these positions are encrypted by use of discrete logarithmics and sent. This not only strengthens the encrypted message but also makes the transmission more secure, so that only the intended recipient of the message is able to decipher the message. This paper focuses on enhancing the confidentiality of the message transfer.

General Terms
Cryptography; Discrete Logarithmics; Merkle-Hellman Knapsack Cryptosystem; RSA algorithm

Keywords
Security; cryptography; cryptosystem; blocks cipher; cipher block chaining;discrete logarithmics;knapsack problem; randomizers; superincreasing vector.

1. INTRODUCTION
In the Cipher Block Chaining Mode of encrytion, the input to the encryption algorithm is the output from the XOR operation performed on the current plain text block and the preceding cipher text block. A Symmetric key is used for each block. Here the processing of the sequence of plain text blocks is chained together. The input to the encryption function for each plaintext block bears no fixed relationship to the plain text block.

On the other hand, the knapsack problem is a NP complete problem in combinatorial optimization. The knapsack problem selects the most useful items from a number of items given that the knapsack has a certain capacity. Knapsack problems are widely used to design solutions to the industrial problems using Public-key cryptography.

The 0-1 knapsack problem states that there is a knapsack with a given capacity and a certain number of items that need to be filled inside the knapsack. Each item has a value and a weight associated with it. The knapsack problem selects the items that can be put in the knapsack so that the value of all the items maximizes the weight and does not increase the total capacity of the knapsack.

This can be denoted as –

Maximize $\sum^{n}_{i=0} b_i x_i$ (1)

Subject to $\sum^{n}_{i=0} w_i x_i \leq W$ (2)

$$x_i = \begin{cases} 1, \text{ if the item is included in the knapsack} \\ 0, \text{if the item is not included in the knapsack} \end{cases} \quad (3)$$

where,

'b' is the value associated with each item i
'w' is the weight associated with each item i
'W' is the maximum capacity of the knapsack
'n' is the number of items

The subset sum problem is a special case of the knapsack problem [5].This problem finds a group of integers from a list vector V, where V = (v1, v2, v3, … vn), with the subset of elements in the vector V gives a sum of S. It also determines if a vector X = (x1, x2, x3… xn) exists where xi element of {0,1} so that V*X = S [5]. Then A super increasing knapsack vector s is created and the super-increasing property is hidden by creating a second vector M by modular multiplication and permutation. The vector M is the public key of the cryptosystem and is used to decrypt the message [2].

2. EXISTING SYSTEM
The existing system uses either Cipher Block Chaining Mode or Merkle-Hellman Knapsack Cryptosystem to perform encryption and decryption operations on the plain text. The Cipher Block Chaining (CBC) Mode is a type of Block Cipher Modes of operation which takes n-blocks of plain text and performs encrytion operation on all the blocks simultaneuosly whereas, the Merkle-Hellman invented in 1978 is based on the superincreasing subset problem. But both the crypto-systems had several disadvantages.

In the Cipher Block Chaining Mode the original text can be easily got back from the cipher text by just knowing the key or the Initial Vector. Again in case of Merkle-Hellman cryptosystem by knowing the value of the number and the co-prime number, we can easily trace back the plain text.

3. PROPOSED SYSTEM

To overcome the drawbacks, we propose a system where we use the combination of both the cryptosystems and also make use of randomizers and discrete logarithmics in order to encrypt messages so that only the intended recipient of the message is able to decipher the messages.
The structure is prepared such that the plain text is encrypted using Cipher Block Chaining Mode and then further encrypted using the Merkle-Hellman cryptosystem. The key keeps on varying since it's calculated based on the final output of the previous block. This helps to get an encrypted message of higher strength because we use a randomized vector, key as well as a super-increasing subset. Even the encrypted codes are stored in secret buffers and only those positions are encrypted through discrete logarithmics and sent to the receiver.

The original message can be obtained by performing the operations in the reverse order. i.e, perform Merkle-Hellman algorithm and then the Cipher Block Chaining Mode on the cipher text with the help of the varying vector, key and super-increasing sequence which was used during encryption.

We describe the various steps of our proposed cryptosystem in the following paragraphs. The encryption procedure for the message is described in section IV of this paper where we use the string "Go" as input secret message and perform several operations on it to get the message to be transmitted as "000300460131013900310159".

In section V, we describe the steps to decrypt this received message "000300460131013900310159" and get back the original message "Go". Both the encryption and decryption processes take the same initial vector, initial key and super-increasing which is chosen by the use of a random number generating algoritm.

In section VI, we show the outputs for both encryption and decryption techniques. Thus we prove that our proposed cryptosystem is working correctly. We have discussed about the future enhancements to our proposed technique in section VIII of this paper.

4. ENCRYPTING MESSAGES

The proposed cryptosystem performs encryption in three steps. First the plain text is divided into single characters and the characters are converted to their binary equivalent. Each of these represents a block of code. Each of these blocks is encrypted by CBC to form blocks of temporary cipher text. The initialization vector and the key are randomly generated.

Secondly, the temporary cipher text is encrypted through the Merkle-Hellman encryption scheme whose main idea is to create a subset problem which can be solved easily and then to hide the super-increasing nature by modular multiplication and permutation. The super-increasing sequence is generated randomly.

Thirdly the encrypted message is stored in secret buffers and sent to receiver. The postions where this message is stored is hashed, encrypted by use of discrete logarithmic functions and sent to reciever.

The key used to encryt the text of successive blocks in CBC is made to be varying based on the final output of the previous block of code. The transformed vector forms the encrypted message and the original superincreasing vector forms the private key and is used to decipher the message.

The encryption process works as a whole where the blocks are simultaneously encrypted.

Figure 1 demonstrates the encryption technique. The plain text is broken down into blocks of seven bits each. p1, p2,…., pn represents the blocks of plain text. Each block of plain text is encrypted by using a vector and a key, thereby producing a temporary code (TC). This temporary code is further encrypted using the Merkle-Hellman cryptosystem to get the final blocks of cipher text (C1, C2,.. Cn). After all blocks of code have been encrypted, they are combined together to get the cipher text.

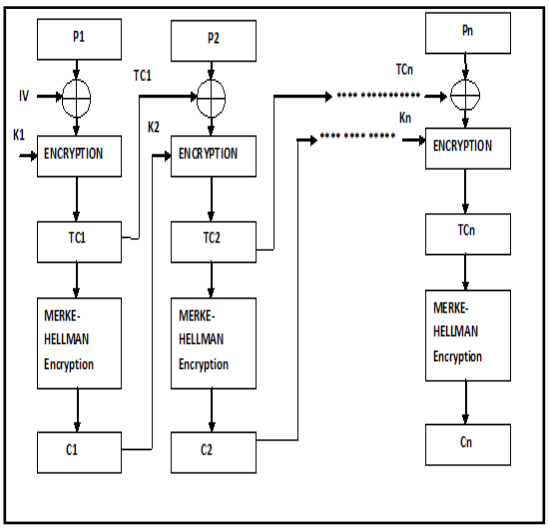

Figure 1: Block diagram of the encryption process

4.1 Mathematical Explanation

The first step is to generate a random initial vector (IV) and a key and a super-increasing sequence.These are used to perform the first encryption process.

The second step is to convert all the characters of the message into binary. The binary sequence is represented by the variable "b".

The third step is to perform CBC to get the temporary cipher text.

The fourth step is to take the super-increasing sequence already generated randomly. A superincreasing sequence is one where every number is greater than the sum of all preceding numbers.

$$s = (s1, s2, s3, \ldots. sn) \quad (4)$$

The fourth step is to choose two secret numbers – an integer (a), which is greater than the sum of all numbers in the sequence's' and its co-prime (r).The sequence 's' and the numbers 'a' and 'r' collectively form the Private key of the cryptosystem.

A sequencing vector (b) is used during the second stage of encryption. Its used to calculate the set of the vector sequence by multiplying the co-prime number 'r' with the corresponding element in the super-increasing sequence set (s) and then taking the modulus of the product with respect to 'a'.

Therefore, $b_i = r * s_i \bmod (a)$. (5)
All elements b1, b2, b3, …. bn of the sequence b are multiplied with the corresponding elements of the binary sequence b. The numbers are then added to create the encrypted message Ci.
This sequence C = (C1, C2, C3… Cn) forms the coded message of the cryptosystem.
These blocks of code are stored in the secret buffer and the index of the stored message are encrypted by use of discrete logarithm and sent to the receiver along with the initial vector, key, hash information and the sequence information.

4.2 Example

For encrypting the string "Go"

Step 1.

The first step is to generate a random IV (Initial Vector) and the random key.

Suppose the randomly generated initial vector (IV) and the key (k1) are: The IV = 0 0 1 1 1 1 1

Key (k1)= 0 1 0 1 1 1 0

Then all the characters in the string are converted into binary

G = 1 0 0 0 1 1 1

o = 1 1 0 1 1 1 1

Here the binary equivalents of each and every character are considered as a separate block.

The super-increasing sequence to be used later is also generated randomly. Here, three randomly genreated numbers are used to generate the required random sequence. Suppose, the three numbers are 1, 5 and 9.

Using this, the random sequence is generated through a certain combination as {1, 5, 10, 20, 40, 80, 160}

Step 2: For 1st block.

The binary value of 'G' is XORed with IV and then the result is again XORed with the key.

p1, p2,…., pn represents the blocks of plain text. Each block of plain text is then encrypted by using a vector and a key, thereby producing a temporary code (TC). TC1,TC2,… TCn represents the temporary codes for the plain text blocks p1,p2,..pn respectively.

Thus,TC1= ((0011111) XOR (100011 1))XOR (010111 0)

= 1 1 1 0 1 1 0

This completes the CBC stage of 1st block.

Now we perform Merkle-Hellman scheme on the temporary output of 1st block.

The first step is to choose a superincreasing sequence. In this case the sequence is s = (1, 5, 10, 20, 40, 80, 160)

The two secret numbers chosen are a = 4439, r = 10.

The sequence vector is b = b1, b2,… bn

Where $b_i = r * s_i \bmod a$.

The message is encrypted by multiplying all the elements of sequence b with the corresponding elements of sequence s and adding the resulting sum.

Therefore, the encrypted message

$$C = \sum^{n}_{i=0} b_i * (TCj)_i \quad (6)$$

Where, j represents the jth block

b1 = 1 * 10 mod 1439 = 10
b2 = 5 * 10 mod 1439 = 50
b3 = 10 * 10 mod 1439 = 100
b4 = 20 * 10 mod 1439 = 200
b5 = 40 * 10 mod 1439 = 400
b6 = 80 * 10 mod 1439 = 800
b7 = 160 * 10 mod 4439 = 161

Encrypting the character G –

b = (10, 50, 100, 200, 400, 800, 161)

and TC1= (1 1 1 0 1 1 0)

CG = 10+50+100+400+800 = 1360

Step 3: For 2nd block

The second character to be encrypted is o.

Here the temporary encrypted value of the previous block 'G' is taken as the Initial vector IV, from previous step and perform XOR operation with the input block.

The previous temporary encrypted value was TC1= 1 1 1 0 1 1 0.

The key is obtained by taking the final output of the 1st block and then calculating the remainder value by dividing it by 128.

i.e. k2= binary equivalent of (CG mod 128)
= binary equivalent of (1360 mod 128)
= binary equivalent of (80)
= 1 0 1 0 0 0 0.

Now, TC2= ((1 1 1 0 1 1 0) XOR (1 1 0 1 1 1 1)) XOR (1 0 1 0 0 0 0)

= 1 0 0 1 0 0 1

b = (10, 50, 100, 200, 400, 800, 161) and TC2= (1 0 0 1 0 0 1)

Therefore, Co = 10 + 200 + 161 = 371.

Therefore, the actual encrypted message value is C= 13600371.

Step 4: Performing the hashing and logarithmic functions.

In this step the encrypted values are hashed onto some positions and those position values are encrypted using discrete logarithmic functions by using the concept of RSA algorithm.

A random number is generated to make the hash more secure. Here the hash value generated is 3.

Here the encrypted codes are hashed onto the positions 2 and 3. These positions are encrypted using the RSA algorithmic concepts where the fixed values considered are p = 17; q = 11; n= p*q = 187; ø (n) = (p-1)*(q-1) =160; e=7;

For the above values d=23

Thus, while encrypting the position values the fomula used is:

Ci = Me mod n

Thus for position2, the encrypted position code is

Cpos2= 27 mod 187 =128

This is added to the hash value to enable more secrecy.

Thus the resulting position value is 128 + 3 = 131.

This hash number is multiplied with itself and the resulting number is stored in the original hash number.Thus the present hash value becomes 9.

For the 3rd position, the encrypted position index becomes,

Cpos3= 37 mod 187 = 130

This is added to the hash value to enable more secrecy.

Thus the resulting position value is 130 + 9 = 139.

The cipher text to be transmitted now becomes 000300460131013900310159 where the first four digits represent the random hashing number generated. The next four digits represent the key. The next eight digits represent the position of the two character codes. The next four digits represent the initial vector and the last four digits contain information to generate the super-increasing sequence.

Thus, the message to be transmitted is 000300460131013900310159.

5. DECRYPTING MESSAGES

During the decryption process, the received message is first broken down into the cipher text part (position information), initial vector,

hash information, key information and the sequence information. These encrypted position codes are decrypted to find out the original positions where the encrypted blocks were hashed onto. These blocks of cypertexts are then decrypted simulataneously. Each block is first decrypted using the Merke-Hellman, and then further decrypted using the vector and the key. This is repeated for all the blocks as the system is a CBC scheme.

Figure 2 demonstrates the decryption technique. The decryption process is opposite to the encryption process. The encrypted cipher text is broken down into blocks of code C1,C2,…, Cn. Each of these blocks is first decrypted by using Merkle-Hellman Decryption algorithm. We get a temporary code (TC) which acts as the vector for XOR operation of the next block of code. The key and vector are used for the decryption process of the temporary code TC by CBC decryption technique. The output of the CBC decryption process gives the original block of plain text. To get the original message back the plain text blocks p1,p2,…,pn are chained together.

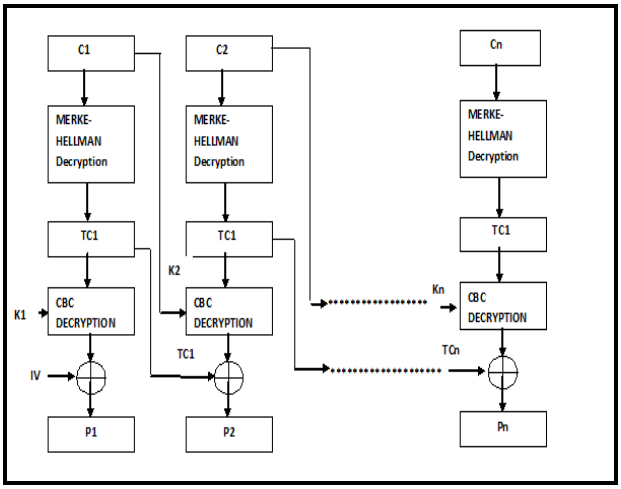

Figure 2: Block diagram of Decryption Process

5.1 Mathematical Explanation

To decrypt the message C, the recipient of the message would have to find the bit stream which satisfies the Equation [1]–

$$M = \sum_{i=0}^{n} p_i * b_i \qquad (7)$$

To solve equation (7), the user would need the private key (s, a, r).

The first step finds out the original hash positions by decrypting the encrypted positions using the RSA concepts to decrypt.

The formula used is P_i = Cid mod n

From the original position values, the encryted message blocks are got.

The second step is to calculate the modular multiplicative inverse of 'r' in r mod a [4].

This is calculated using the Extended Euclidean algorithm. This modular multiplicative inverse of 'r' in r mod a is denoted as r-1.

The third step is to multiply each element of the encrypted message (C) with r-1 mod a.

The largest number in the set which is smaller than the resulting number is subtracted from the number. This continues until the number is reduced to zero[1]. This temporary code is then fed into the CBC cryptosystem where each temporary block is passed through the decryption algorithm. The result is XORed with the preceeding ciphertext block to produce the plaintext block. [7]

The decryption technique involves the same algorithms executed in the reverse order.

5.2 Example

Decrypting the message: 000300460131013900310159.

Step 1.

From the received message the message is broken down into four parts. The information about hash, the randomly generated key, the encryted position indexes, the initial vector and the information about the super-increasing sequence.

The first four digits represent the random hashing number generated. The next four digits represent the key. The next eight digits represent the position of the two character codes. The next four digits represent the initial vector and the last four digits contain information to generate the super-increasing sequence.

Thus the hash number is 3, the key is 46, the encrypted positions are 131 and 139, the initial vector is 31 and the sequence information contains the numbers 1, 5 and 9.

Step 2:

Finding out the original positions.

Using the RSA concepts now the encrypted position indexes are decrypted.

The formula used is Pi = Cid mod n
The values used were p = 17; q = 11; n= p*q = 187;
ø (n) = (p-1)*(q-1) =160; e=7;
From the above values we got d=23
Thus to get the first position, we perform the decryption operation. First the position is reduced by the hash number. Thus pos1=131-3=128. It is then operated for logarithmic decryption.
Pos1= 12823 mod 187 = 2
Thus we come to know that the first block of encypted message is in position 2.
Similarly, for position 2 the hash number is multiplied with itself and subtracted from the encrypted position2 value. Its then operated for logarithmic decryption.

Pos 2 = (139-9)23 mod 187 = 3

Thus the second block of encypted message is in position 3.

In position2, we find out the encrypted block to be 1360

In position3, we find out the encrypted block to be 371.

Therefore, C1= 130, C2 = 371

Here C1, C2 represent the cipher blocks of code.

Step 3: Decrypting 1st block.

Perform Merkle-Hellman scheme on C1=1360
The modular inverse of 10 in 10 mod 1439 is calculated by using the extended Euclidean algorithms and was found to be 144.
The encrypted message is again broken down into blocks of three digits each as C1,C2,..,Cn.

The encrypted message for first block of code C1 is 285 and s = (1, 5, 10, 20, 40, 80, 160)

To decrypt the message we multiply the block of encrypted code with the modular inverse got and then modulus of it is taken with respect to the number chosen initially.

Thus, 1360 * 144 mod 1439 = 136

The largest number in the sequence s, which is smaller than 136 is 80.

136 – 80 = 56
56 – 40 = 16
16 – 10 = 6
6 - 5 = 1
1 - 1 = 0

Therefore, the binary sequence becomes 1 1 1 0 1 1 0.
This is same as the temporary code (TCi) we got during encryption of the same block.

Now we apply CBC on the 1st block. The Cipher text is XORed with the key and then with the IV or C i-1 to get the required blocks of plain text in binary form, which is converted to alphabet format to get the message back.

We XOR it first with the Initial key k1 and then with the Initial Vector (IV) to get the first block of plain text.

Therefore,

P1= ((0 1 0 1 1 1 0) XOR(1 1 1 0 1 1 0))XOR (0 0 1 1 1 1)

= (1 0 0 0 1 1 1)

The character equivalent of this binary value is G.

Step 4: For 2nd block
Similarly, for the second block C2= 371 we perform Merkle-Hellman Decryption technique first.

For C= 371 * 144 mod 1439 = 181

181 – 160 = 21
21 - 20 = 1
1 - 1 = 0

Therefore, TC2= (1 0 0 1 0 0 1).
Now for block 2,
The key is got by taking the cipher code C1 and getting the remainder got by dividing it with 128 and then getting the binary equivalent of the reaminder.
i.e. k2= binary equivalent of (CG mod 128)
= binary equivalent of (1360 mod 128)
= binary equivalent of (80)
= 1 0 1 0 0 0 0.

We XOR the second block of cipher text (C2) first with the key k2, and then with the temporary code of previous step TC1 to get the second block of plain text.

Therefore,

P2= ((1 0 1 0 0 0 0) XOR(1 0 0 1 0 0 1)) XOR (1 1 1 0 1 1 0)

= 1 1 0 1 1 1 1

The character equivalent to it is 'o'.
Thus the original message "Go" is got back.
Thus, by taking the initial vector, key and super-increasing sequence randomly and by use of hash functions and the plain text as input, we encrypt the plain text and get the cipher text as output. In order to get back the original message this cipher text is used along with the same initial vector, key and super-increasing sequence for decryption.

6. EXPERIMENTAL RESULTS

To demonstrate the proposed system we use Java platform and BlueJ version 1.3.5 as the software. The number of lines used in coding for developing the cryptosystem is 763.

We take any string, and a difficulty level as inputs and we get the cipher text as output.

Both the encryption and decryption processes are demonstrated in the figure 3 given below.Here we take the plain text as"Go" as input.The initialization vector, the key and the super-increasing are generated randomly. We get the message to be transmitted as "000300460131013900310159". Here the first four digits represent the random hashing number generated. The next four digits represent the key. The next eight digits represent the position of the two character codes. The next four digits represent the initial vector and the last four digits contain information to generate the super-increasing sequence.

This data helps the receiver to decrypt the data.

For the decryption process, the received message is "000300460131013900310159". From the received message the message is broken down into four parts. The information about hash, the randomly generated key, the encryted position indexes, the initial vector and the information about the super-increasing sequence.

Thus the hash number is 3, the key is 46, the encrypted positions are 131 and 139, the initial vector is 31 and the sequence information contains the numbers 1, 5 and 9. Using this information we can find out the positions where the coded message is hidden and then the message can be decrypted to get the original plain text "Go" back.

Thus, by entering the original message as "Go" we generate a cipher text which is transmitted to the receiver. This transmitted message is fed to the decryption algorithm of the cryptosystem present on the receiver side and we get the original message "Go" back. Thus the cryptosystem works successfully.

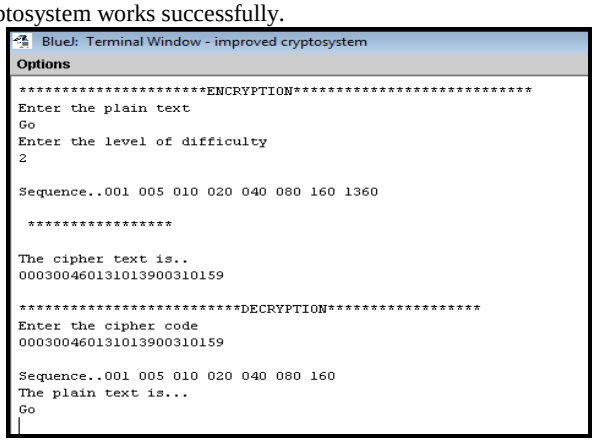

Figure 3: Combined Output of Encryption and Decryption Process

It is to be noted that the randomly generated sequence won't be displayed on the screen. Only the cipher text will be displayed. We display the randomly generated sequence for reference in order to explain our paper better.

7. CONCLUSION

This paper explains how to encrypt and decrypt data by enhancing the work of the Cipher block chaining mode through the use of Merkle-Hellman Knapsack cryptosystem. Furthermore, the randomly generated vector, key and super-increasing sequence strengthens the encryption process. Since the encryption of the successive blocks depends on the temporary codes and outputs of previous blocks, the message to be transmitted gets really difficult to be broken. The whole cryptosystem is mase even more secure by making the transfer

of message in secret buffers which is kept hidden. Only the positions where the encrpted messages are stored are encrypted through discrete logarithmics and displayed. This makes the proposed cryptosystem even more secure. The whole cryptosystem was demonstrated by encrypting a string "Go" and then decrypting it. The decrypted string matched with the original string.

8. FUTURE SCOPE

The algorithms can further be strengthened by the use of various schemes like the use of shifting algorithms where the randomly generated key or vector can be rotated left or right or in a circular way.

9. ACKNOWLEDGEMENTS

We are greatly thankful to Mr. Asish Agarwal for presenting such a wonderful idea of sending secret messages through the use of Merkle-Hellman Knapsack Cryptosystem. His idea is the backbone of our proposed cryptosystem which will help in tranfer of secret messages more securely.

10. REFERENCES

[1]. B.Padhmavathi, Arghya Ray, Alisha Anjum, Santhoshi Bhat,"Improvement of CBC Encryption Technique by using the Merkle - Hellman Knapsack Cryptosystem", IEEE Madras sponsored ISCO 2013

[2]. Ashish Agarwal ,"Encrypting Messages using the Merkle Hellman Knapsack Cryptosystem", IJCSNS International Journal of Computer Science and Network Security, VOL.11 No.5, May 2011

[3]. A.Menezes, P.vanOorschot and S.Vanstone, "Handbook of Applied Cryptography", CRC Press, 1996

[4]. R.Merkle and M.Hellman, "Hiding information and signatures in trapdoor knapsacks", IEEE Transactions on Information Theory – 24, 5, pp 525 – 530.

[5]. W.Diffie and M.Hellman, "New directions in cryptography", IEEE Transactions on Information Theory – 22, 6, pp 644 – 654.

[6]. http://www.mast.queensu.ca/~math418/m418oh/m418oh04.pdf

[7]. http://mathworld.wolfram.com/SubsetSumProblem.html

[8]. William Stallings," Cryptography and Network Security", fifth edition,pg-225-227.