

Enhancement of Merkle-Hellman Knapsack Cryptosystem by use of Discrete Logarithmics

Arghya Ray*, Santhoshi Bhat**

*Department of CSE, SRM University, Chennai
**Department of CSE, SRM University, Chennai

Abstract- The Merkle-Hellman invented in 1978 is based on the superincreasing subset problem. Ralph Merkle and Martin Hellman used the subset problem to create a cryptosystem to encrypt data. 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 s is used to decrypt the message.

This paper demonstrates how to strengthen the encrypted message being sent by use of discrete logarithmics so that only the intended recipient of the message is able to decipher the message.

Index Terms- Security; cryptography; cryptosystem; discrete logarithm; euler’s totient function; knapsack problem; superincreasing vector.

I. INTRODUCTION

The knapsack problem is an NP complete problem in combinatorial optimization. The knapsack problem selects the most useful items from a number of items given that the knapsack or the rucksack has a certain capacity.

Knapsack problems are widely used to model solutions industrial problems such as public-key cryptography.

The 0-1 knapsack problem states that if there is a knapsack with a given capacity and a certain number of items that need to be put in 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 is maximized and the weight does not increase the total capacity of the knapsack.

This can be denoted as –

$$\text{Maximize } \sum_{i=0}^n p_i x_i \quad (1)$$

$$\text{Subject to } \sum_{i=0}^n w_i x_i \leq W \quad (2)$$

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

where,

‘ p ’ 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 $V = (v_1, v_2, v_3, \dots, v_n)$, the subset of elements in the vector V which have a given sum S . It also determines if a vector $X = (x_1, x_2, x_3, \dots, x_n)$ exists where x_i element of $\{0,1\}$ so that $V * X = S$ [5].

Ralph Merkle and Martin Hellman used the subset problem to create a cryptosystem to encrypt data. 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 s is used to decrypt the message [2].

II. EXISTING SYSTEM

The existing system uses only Merkle-Hellman Knapsack Cryptosystem to encrypt the messages. Though the encrypted message is difficult to break, it can be found out if we get to know the pattern of the message being sent.

III. PROPOSED SYSTEM

Here we propose a cryptosystem which makes use of the Merkle-Hellman structure and also makes use of discrete logarithmics to encrypt the message. Now the hacker has to know the super-increasing sequence, the private key and public key in order to decrypt the message which strengthens the message to be sent.

IV. ENCRYPTING MESSAGES

The proposed cryptosystem performs encryption in two steps.

First the plain text is broken down into each character and the characters are converted to their binary equivalent. These characters are then 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.

Secondly, these encrypted characters are further encrypted through the use of discrete logarithmics based on RSA concepts. We choose two prime numbers and calculate ‘ n ’ as the product of these two prime numbers, euler’s totient function $\phi(n)$ as the

result of $(p-1) * (q-1)$. We choose another number 'e' which is relatively prime to the other two prime numbers which we had earlier chosen and $\text{gcd}(e, \phi(n)) = 1$.

Using these details we find out the value of 'd' such that $d \equiv e^{-1} \pmod{\phi(n)}$

This (e, n) acts as the private key and the pair (d, n) acts as the public key.

Thus the formula to encrypt the message using discrete logarithmics is $C = M^e \pmod{n}$ (4).

A. Mathematical Explanation

The first step is to choose an initial vector IV and a key of length 7 bits. 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 choose a superincreasing sequence of positive integers. A superincreasing sequence is one where every number is greater than the sum of all preceding numbers.

$$s = (s_1, s_2, s_3, \dots, s_n) \quad (4)$$

The fourth step is to choose two 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.

All the elements – $s_1, s_2, s_3, \dots, s_n$ of the sequence 's' are multiplied with the number 'r' and the modulus of the multiple is taken by dividing with the number a.

Therefore,

$$p_i = r * s_i \pmod{a}. \quad (5)$$

All elements $p_1, p_2, p_3, \dots, p_n$ of the sequence p are multiplied with the corresponding elements of the binary sequence b. The numbers are then added to create the encrypted message M_i .

The sequence $M = (M_1, M_2, M_3 \dots M_n)$ forms the cipher text of the cryptosystem.

B. Example

– Encrypting the string "get"

Step 1. Finding character equivalent

The first step is to convert all the characters in the string into their binary equivalent –

$$\begin{aligned} g &= 1100111 \\ e &= 1100101 \\ t &= 1110100 \end{aligned}$$

Step 2: For 2nd character

The next step is to choose a super-increasing sequence. In this case the sequence is –

$$s = (3, 5, 15, 25, 54, 110, 225)$$

The binary sequence is $c = (c_1, c_2, c_3 \dots c_n)$

The two numbers chosen are – 439 and 10.

$$a = 439, r = 10$$

The sequence $p = p_1, p_2, \dots, p_n$

$$\text{Where } p_i = r * s_i \pmod{a}$$

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

Therefore, the encrypted message

$$M = \sum_{i=0}^n p_i * b_i \quad (6)$$

$$p_1 = 3 * 10 \pmod{439} = 30$$

$$p_2 = 5 * 10 \pmod{439} = 50$$

$$p_3 = 15 * 10 \pmod{439} = 150$$

$$p_4 = 25 * 10 \pmod{439} = 250$$

$$p_5 = 54 * 10 \pmod{439} = 101$$

$$p_6 = 110 * 10 \pmod{439} = 222$$

$$p_7 = 225 * 10 \pmod{439} = 55$$

Encrypting the character g –

$$p = (30, 50, 150, 250, 101, 222, 55)$$

The binary equivalent of 'g' is 1 1 0 0 1 1 1

$$M_g = 30 + 50 + 101 + 222 + 55 = 458$$

This is encrypted using logarithmic functions. The logarithmic functions are based on the RSA algorithmic concepts.

Here the prime numbers chosen are 53 and 31. The value of 'e' is chosen as 7. Therefore 'd' becomes 223. The value of n is the product of the two relatively prime integers. Thus here $n = 1643$.

Thus the message is encrypted according to the formula, $C = M^e \pmod{n}$ (7)

Where, (e, n) also act as private keys

Thus the encrypted code for the first character becomes $C_g = 458^7 \pmod{1643} = 344$

Step 3: For 2nd character

The second character is 'e'.

Its binary equivalent is 1 1 0 0 1 0 1

Encrypting the character e –

$$p = (30, 50, 150, 250, 101, 222, 55)$$

The binary equivalent of 'e' is 1 1 0 0 1 0 1

$$M_e = 30 + 50 + 101 + 55 = 236$$

This is encrypted using logarithmic functions.

Thus the encrypted code for the second character becomes $C_e = 236^7 \pmod{1643} = 937$

Step 4: For 3rd character

The second character is 't'.

Its binary equivalent is 1 1 1 0 1 0 0

Encrypting the character e –
 $p = (30, 50, 150, 250, 101, 222, 55)$

The binary equivalent of 'e' is 1 1 1 0 1 0 0
 $M_t = 30 + 50 + 150 + 101 = 331$

This is encrypted using logarithmic functions.
 Thus the encrypted code for the second character becomes $C_t = 331^7 \text{ mod } 1643 = 1499$

These codes are combined together and sent to the receiver.
 Thus the message to be transmitted to the receiver is 034409371499.

V. DECRYPTING MESSAGES

During the decryption process, the blocks of encrypted code are separated. On these blocks first logarithmic decryption is performed using the RSA concepts.

The formula used is $M = C^d \text{ mod } n$ (8)

The values of the prime numbers, e, d and n are same as that used during encryption.

The output of it is decrypted using Merkle-Hellman Knapsack cryptosystem decryption process.

A. Mathematical Explanation

To decrypt the message M, the recipient of the message would have to find the bitstream which satisfies the Equation [1]–

$$M = \sum_{i=0}^n p_i * b_i \quad (9)$$

To solve the equation (8), the user would need the private key (s, a, r).

The first step is to calculate the modular multiplicative inverse of 'r' in $r \text{ mod } a$ [4].

This is calculated using the Extended Euclidean algorithm. This is denoted by r-1.

The second step is to multiply each element of the encrypted message (M) 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 is just the processes taking place in reverse order.

B. Example

Decrypting the message: C=034409371499

Step 1.

Separate the cipher text into groups of 4 digits from the starting position.

Therefore, $C1 = 0344$
 $C2 = 0937$
 $C3 = 1499$

Step 2: Decrypting 1st encrypted code.

Perform discrete logarithmic decryption schemes on $C1=0344$ using the concepts of RSA.

Thus, the prime numbers chosen were 53 and 31. The value of 'e' was chosen as 7. Therefore 'd' becomes 223. The value of n is the product of the two relatively prime integers. Thus here $n = 1643$.

Thus the message is encrypted according to the formula,
 $M = C^d \text{ mod } n$ (10)

Where, (d, n) also act as public keys

Thus, $M1 = 344^{223} \text{ mod } 1643 = 458$

The modular inverse of 10 in $10 \text{ mod } 439$ is calculated using the extended Euclidean algorithms and was found out to be 44.

The encrypted message M1 is 458 and
 $s = 3, 5, 15, 25, 54, 110, 225$.

Again, $458 * 44 \text{ mod } 439 = 397$

The largest number in the sequence s, which is smaller than 397 is 225.

$$\begin{aligned} 397 - 225 &= 172 \\ 172 - 110 &= 62 \\ 62 - 54 &= 8 \\ 8 - 5 &= 3 \\ 3 - 3 &= 0 \end{aligned}$$

Thus, the binary sequence becomes 1 1 0 0 1 1 1.
 The character equivalent to this binary sequence is 'g'.

Step 3: Decrypting 2nd encrypted code.

Perform discrete logarithmic decryption schemes on $C2=0937$ using the concepts of RSA.

Thus, $M2 = 937^{223} \text{ mod } 1643 = 236$

The encrypted message M2 is 236 and
 $s = 3, 5, 15, 25, 54, 110, 225$.

Again, $236 * 44 \text{ mod } 439 = 287$

The largest number in the sequence s, which is smaller than 287 is 225.

$$\begin{aligned} 287 - 225 &= 62 \\ 62 - 54 &= 8 \\ 8 - 5 &= 3 \\ 3 - 3 &= 0 \end{aligned}$$

Thus, the binary sequence becomes 1 1 0 0 1 0 1.
The character equivalent to this binary sequence is ‘e’.

Step 4: Decrypting 3rd encrypted code.

Perform discrete logarithmic decryption schemes on $C_3=1499$ using the concepts of RSA.

$$\text{Thus, } M_3 = 1499^{223} \text{ mod } 1643 = 331$$

The encrypted message M_3 is 331 and
 $s = 3, 5, 15, 25, 54, 110, 225$.

$$\text{Again, } 331 * 44 \text{ mod } 439 = 77$$

The largest number in the sequence s , which is smaller than 77 is 54.

$$\begin{aligned} 77 - 54 &= 23 \\ 23 - 15 &= 8 \\ 8 - 5 &= 3 \\ 3 - 3 &= 0 \end{aligned}$$

Thus, the binary sequence becomes 1 1 1 0 1 0 0.
The character equivalent to this binary sequence is ‘t’.
These decrypted characters are combined together and the resultant output is ‘get’.

Thus the original message “get” is got back.

VI. 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 354.

We take any string as inputs and we get the cipher text as output.

This encryption process is demonstrated in the figure 3 given below. Here we take the plain text “get” as input. We get the cipher text as “034409371499”. Here the first four digits represent the encrypted first character; the next four digits represent the encrypted second character and henceforth.

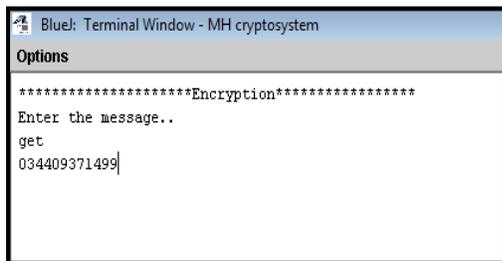


Figure 3: Output of Encryption Process

The decryption process is demonstrated in the figure 4 given below. The cipher text received is “034409371499”. The digits are extracted four at a time and the numbers are decrypted to get the original message back.

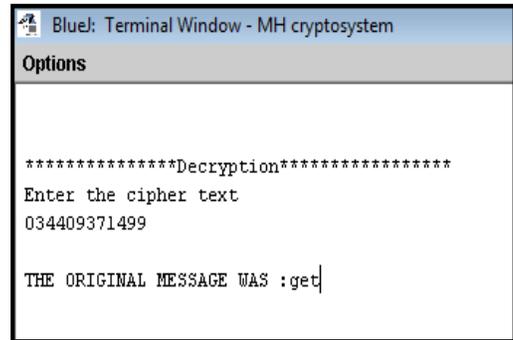


Figure 4: Output of Decryption Process

The original message and the decrypted message matched. Thus the cryptosystem works successfully.

VII. CONCLUSION

This paper explained how to encrypt and decrypt data by enhancing the working of Merkle-Hellman Knapsack cryptosystem through the use of discrete logarithmic concepts. The whole cryptosystem was demonstrated by encrypting a string “get” and then decrypting it. The decrypted string matched the original string.

VIII. FUTURE SCOPE

The future scopes include use of shifting and hashing algorithms. The encryption algorithm can be strengthened by combining it with other encryption schemes also.

REFERENCES

- [1] Ashish Agarwal, “Encrypting Messages using the Merkle Hellman Knapsack Cryptosystem”
- [2] A.Menezes, P.vanOorschot and S.Vanstone, “Handbook of Applied Cryptography”, CRC Press, 1996
- [3] R.Merkle and M.Hellman, “Hiding information and signatures in trapdoor knapsacks”, IEEE Transactions on Information Theory – 24, 5, pp 525 – 530.
- [4] W.Diffie and M.Hellman, “New directions in cryptography”, IEEE Transactions on Information Theory – 22, 6, pp 644 – 654.
- [5] <http://www.mast.queensu.ca/~math418/m418oh/m418oh04.pdf>
- [6] <http://mathworld.wolfram.com/SubsetSumProblem.html>
- [7] William Stallings, “Cryptography and Network Security”, fifth edition, pg-225-227.

AUTHORS

First Author – Arghya Ray, Department of CSE, SRM UNIVERSITY, Chennai, archiebabai@gmail.com

Second Author – Santhoshi Bhat, Department of CSE, SRM UNIVERSITY, Chennai, santhoshibhat@gmail.com

