ASYMMETRIC KEY CRYPTOGRAPHY USING MERKLE-HELLMAN KNAPSACK METHOD AND GENETIC ALGORITHM

N. Vinothini
Assistant Professor, Department of Information Technology Madurai
Sivakasi Nadars Pioneer Meenakshi Womens College, Poovanthi.
senthavinocs@gmail.com

ABSTRACT

Over the twentieth century of world, security is the basic and utmost concern of the emerging technologies in all fields of science especially in information transformation. Towards them, cryptography plays a crucial role on passing the secret codes across the internet or other communication media. A newly designed encryption algorithm using traditional cryptographic concepts with augmented techniques is proposed in this paper. In the first step, a publically known random key is devised through S-DES key generation algorithm. Using the automatic key an encryption algorithm is generated with basic arithmetic and logic operators. This paper emphasizes the substitution as well as transposition technique. To improve the cryptosystem, genetic algorithm is utilized to establish an approach for transposition technique. And thus make use of all ASCII values strengthens the security issues of proposed system.

Keywords—S-DES key generation, ASCII value, genetic algorithm, encryption, decryption, knapsack problem.

I. INTRODUCTION

The field of computer science forces us to employ security measures while the computers have the capability to transfer all type of sensitive data. Nowadays their use cannot be ruled out on account of its perfection, time saving and cost cutting edge. The amount of data currently transferred over the internet is about 640 terabytes within a minute and it is expected that the number of devices forming network, which is nearly equal to global population, will double by 2015. This improvement in network traffic will lead to the requirement of the ciphers which provide quick response and have low processing delay but yet are efficient [1]. Communication is the way of transforming a message to others. In networks environment these
communication should be more careful while passing the information. This leads to the evolution of cryptography [2].

Cryptography is the art of science of keeping secrets secret. With the help of cryptography, communication is possible to securely through insecure channels. But with the development of new additional techniques in computer science, Cryptography is further expanded in its application field [2].

Traditionally, there are two ways to solve the cryptographic problems. One can disguise the very existence of the message, perhaps by writing with invisible ink or try to transmit the message via a trustworthy person [2].

II. METHODOLOGY INVOLVED

Cryptography is split into two ways of altering the message systematically to confuse anyone who intercepts it: these are codes and ciphers. A code is a way of changing the message by replacing each word with another word that has a different meaning. Ciphers, on the other hand, convert the message by a rule, known only to the sender and recipient, which change each individual letter (or sometimes groups of letters) [3].

The cryptographer’s task is creating a system which is used in easy way of both in encryption and decryption, but remains secure against attempts to break it. For this reason, many ciphers have developed over the last 4,000 years to try to stop people from discovering what it is that their secret message says [3].

Different types of ciphers are available today and are used for encryption and decryption. Some of them are block ciphers [4], stream ciphers, and hash functions [5]. Block ciphers [4] takes plain text input of fixed size and produces the same sized block of cipher text whereas stream cipher encrypts the stream of data i.e. one byte at a time. This paper mainly concentrate over the ones which use the techniques of substitution and transposition [5] using the ASCII characters [6].

Substitution method [7] involves the replacement of a character by another one whereas in transposition [5] the positions of characters are changed accordingly. And thus this paper will be a combination of these two. A number of algorithms are available today like DES [8] and AES [9], but none of them use the basic substitution and transposition schemes. Moreover, they execute in several rounds thereby contributing to a considerably large processing delay. Also certain algorithms using basic encryption techniques lack in automatic key generation thus contributing to overhead for users. Also these algorithms generally perform encryption by following a fixed order of rounds which can be randomized using a random number generator function [10].

1. GENETIC ALGORITHM

Genetic algorithms (GAs) [11] are adaptive heuristic search algorithms based on mechanics of natural selection and natural genetics. They belong to the class of Evolutionary Algorithms (EAs), which are used to find solutions to optimization problems using mechanisms based on biological evolution such as mutation, crossover, selection and inheritance.

Crossover: In general, crossover can be classified into following types:

a) Single point crossover: In this type of crossover, only one crossover point is chosen to generate new child.

Before
\[
\begin{array}{cccccc}
1 & 0 & 1 & 1 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 1 \\
0 & 1 & 1 & 0 & 0 & 0 \\
\end{array}
\]
After
\[
\begin{array}{cccccc}
1 & 0 & 1 & 1 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 1 \\
1 & 1 & 0 & 1 & 0 & 0 \\
\end{array}
\]
Fig. 1. Single Point Crossover

b) Two point crossover: This type of crossover involves selecting two crossover points to generate new child.

Before
\[
\begin{array}{cccccc}
1 & 0 & 1 & 1 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 1 \\
0 & 1 & 1 & 0 & 0 & 0 \\
\end{array}
\]
After
\[
\begin{array}{cccccc}
1 & 0 & 1 & 1 & 1 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 \\
1 & 1 & 0 & 1 & 0 & 0 \\
\end{array}
\]
Fig. 2. Two Point Crossover

c) Uniform crossover: In this type of crossover bits of child are uniformly taken from both the parents.

Before
\[
\begin{array}{cccccc}
1 & 0 & 1 & 1 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 1 \\
0 & 1 & 1 & 0 & 0 & 0 \\
\end{array}
\]
After

<table>
<thead>
<tr>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Fig. 3. Uniform Crossover

2) Mutation: Mutation is a genetic operator which changes one or more bit values in a chromosome. It is performed on a child after crossover which guarantees the entire state-space will be searched. It’s performed infrequently (depending upon probability of altering a cell in a chromosome).

Before

| 1 | 0 | 1 | 1 | 1 | 0 | 1 |

After

| 0 | 1 | 0 | 0 | 0 | 1 | 0 |

Fig. 4. Mutation

a) Flipping of Bits: It involves selecting one or more bits of chromosome and inverting it.  
b) Boundary Mutation: It involves randomly replacing chromosome with either lower or upper bound.  
c) Non-Uniform Mutation: It is used to increase the probability that amount of mutation will go to 0 with the next generation.  
d) Uniform Mutation: A chosen chromosome cell is replaced with a uniform random value whose range is selected by user.  
e) Gaussian Mutation: It involves adding a unit Gaussian random value to a chromosome cell.

3. MERKLE-HELLMAN KNAPSACK CRYPTOSYSTEM

The Merkle-Hellman knapsack cryptosystem was one of the earliest public key cryptosystems invented by Ralph Merkle and Martin Hellman in 1978. The ideas behind it are simpler than those involving RSA, and it has been broken[13].

Merkle-Hellman is an asymmetric-key cryptosystem, meaning that two keys are required for communication: a public key and a private key. Furthermore, unlike RSA, it is one-way: the public key is used only for encryption, and the private key is used only for decryption. Thus it is unusable for authentication by cryptographic signing.

The Merkle-Hellman system is based on the subset sum problem (a special case of the knapsack problem). The problem is as follows: given a set of numbers $A$ and a number $b$, find a subset of $A$ which sums to $b$. In general, this problem is known to be NP-complete. However, if the set of numbers (called the knapsack) is superincreasing, meaning that each element of the set is greater than the sum of all the numbers in the set less than it, the problem is "easy" and solvable in polynomial time with a simple greedy algorithm.

Key generation

In Merkle-Hellman, the keys are two knapsacks. The public key is a 'hard' knapsack $A$, and the private key is an 'easy', or superincreasing, knapsack $B$, combined with two additional numbers, a multiplier and a modulus. The multiplier and modulus can be used to convert the superincreasing knapsack into the hard knapsack. These same numbers are used to transform the sum of the subset of the hard knapsack into the sum of the subset of the easy knapsack, which is a problem that is solvable in polynomial time.

Encryption

To encrypt a message, a subset of the hard knapsack $A$ is chosen by comparing it with a set of bits (the plaintext) equal in length to the key. Each term in the public key that corresponds to a 1 in the plaintext is an element of the subset $A_m$, while terms that corresponding to 0 in the plaintext are ignored when constructing $A_m$ - they are not elements of the key. The elements of this subset are added together and the resulting sum is the ciphertext.
ASYMMETRIC KEY CRYPTOGRAPHY USING MERKLE-HELLMAN KNAPSACK METHOD AND GENETIC ALGORITHM

Decryption

Decryption is possible because the multiplier and modulus used to transform the easy knapsack into the public key can also be used to transform the number representing the ciphertext into the sum of the corresponding elements of the superincreasing knapsack. Then, using a simple greedy algorithm, the easy knapsack can be solved using O(n) arithmetic operations, which decrypts the message.

Mathematical method

Key generation

To encrypt an n-bit message, choose a superincreasing sequence w = (w_1, w_2, ..., w_n) of n nonzero natural numbers. Pick a random integer q, such that gcd(q, r) = 1 (i.e., r and q are coprime). q is chosen this way to ensure the uniqueness of the ciphertext. If it is any smaller, more than one plaintext may encrypt to the same ciphertext. Since q is larger than the sum of every subset of w, no sums are congruent mod q and therefore none of the private key's sums will be equal.

The public key is \( \beta \), while the private key is (w, q, r).

Encryption

To encrypt an n-bit message

\[ \alpha = (\alpha_1, \alpha_2, ..., \alpha_n), \]

where \( \alpha_i \) is the i-th bit of the message and \( \alpha_i \in \{0, 1\} \), calculate

\[ c = \sum_{i=1}^{n} \alpha_i \beta_i. \]

The ciphertext then is c.

Decryption

In order to decrypt a ciphertext c a receiver has to find the message bits \( \alpha_i \) such that they satisfy

\[ c = \sum_{i=1}^{n} \alpha_i \beta_i. \]

This would be a hard problem if the \( \beta_i \) were random values because the receiver would have to solve an instance of the subset sum problem, which is known to be NP-hard. However, the values \( \beta_i \) were chosen such that decryption is easy if the private key (w, q, r) is known.

The key to decryption is to find an integer s that is the modular inverse of r modulo q. That means s satisfies the equation \( sr \equiv 1 \mod q \). Because of rs \equiv 1 \mod q and \( \beta_i = rw_i \mod q \) follows

\[ \beta_is \equiv w_ir \equiv w_i \mod q. \]

Hence

\[ c' \equiv c s \mod q \]

Because of rs \equiv 1 \mod q and \( \beta_i = rw_i \mod q \) follows

\[ \beta_is \equiv w_ir \equiv w_i \mod q. \]

Hence

\[ c' \equiv \sum_{i=1}^{n} \alpha_i w_i \mod q. \]

The sum of all values \( w_i \) is smaller than q and

\[ \sum_{i=1}^{n} \alpha_i w_i \]

hence is also in the interval [0,q-1]. Thus the receiver has to solve the subset sum problem

\[ c' = \sum_{i=1}^{n} \alpha_i w_i. \]

This problem is easy because w is a superincreasing sequence. Take the largest element in w, say \( w_k \). If \( w_k > c' \), then \( \alpha_k = 0 \), if \( w_k \leq c' \), then \( \alpha_k = 1 \). Then, subtract \( w_k \alpha_k \) from \( c' \), and repeat these steps until you have figured out \( \alpha \).

III. SHORTCOMINGS OF EXISTING ALGORITHM

In existing algorithm, an initial key was supposed to be defined by the user and are not generated from the message itself [14]. The algorithm uses combination of substitution and transposition technique with automatic key generation lacks the multiple rounds for encryption/decryption process, it leads to the intruders job make easy [15]. Improved caesar cipher in multistage encryption process involves columnar and permutation transposition has the

N. Vinothini 86
drawback of limited characters of plaintext message, it uses only 94 ASCII value for their encryption process that ease the third party to grab the information quickly [16]. Enhanced onetime pad cipher with arithmetic and logic operations using flexible key generation algorithm has the disadvantage of memory requirement and processing speed which is applicable to high processing system as well as it requires long key as the message length [17].

The previous algorithm generates an automatic key from the original message length in a shortest manner. On the other hand the previous algorithm has the limited length of plaintext, it uses only 26 alphabetical letters for their whole encryption and decryption process. In the previous work the caesar cipher substitution is used with rail fence transposition which is the simple transposition technique in the traditional cryptographic systems. And thus on the survey of existing systems the work behind them defines the automatic key generation is performed in the simplest way or it has the drawback of size of plaintext, when the paper solves these disadvantages it lacks the property of tedious techniques applied to them [18].

4. EXISTING S-DES KEY GENERATION ALGORITHM

S-DES [19] depends on the use of a 10-bit key shared between sender and receiver. From this key, two 8-bit subkeys are produced for use in particular stages of the encryption and decryption algorithm.

Figure 5 depicts the stages followed to produce the subkeys.

![Flowchart diagram for S-DES key generation algorithm](Fig. 5)

First, permute the key in the following fashion. Let the 10-bit key be designated as (k1, k2, k3, k4, k5, k6, k7, k8, k9, k10). Then the permutation P10 is defined as:

P10 (k1, k2, k3, k4, k5, k6, k7, k8, k9, k10) = (k3, k5, k2, k7, k4, k10, k1, k9, k8, k6)

This P10 is read from left to right; each position in the P10 gives the identity of the input bit that produces the output bit in that position. So the first output bit is bit 3 of the input; the second output bit is bit 5 of the input, and so on.

For example, the key (1010000010) is permuted to (1000001100).

Next, perform a circular left shift (LS-1), or rotation, separately on the first five bits and the second five bits. In our example, the result is (00001 11000). Next we apply P8, which picks out and permutes 8 of the 10 bits according to the following rule: P8 (6, 3, 7, 4, 8, 5, 10, 9)

The result is subkey1 (K1). In our example, these yields (10100100) we then go back to the pair of 5-bit strings produced by the two LS-1 function and perform a circular left shift of 2 bit positions on each string.

In our example, the result is (00001 11000). Next we apply P8, which picks out and permutes 8 of the 10 bits according to the following rule: P8 (6, 3, 7, 4, 8, 5, 10, 9)

The result is subkey1 (K1). In our example, these yields (101000100) we then go back to the pair of 5-bit strings produced by the two LS-1 function and perform a circular left shift of 2 bit positions on each string.

In our example, the value (00001 11000) becomes (00100 00011).

Finally, P8 is applied again to produce K2.

In our example, the result is (01000011). K1=10100100; K2=01000011.

IV. PROPOSED WORK

1. ENCRYPTION ALGORITHM

Step 1: The key deduced from SDES key generation algorithm have a couple of key values i.e. k1=164 & k2=67. To find the best one, chosen is made by pick out the smallest key value among them

Step 2: The reason for taking the smallest key value is determined from the Caesar cipher method. The fact that has been defined in Caesar cipher is the key value must be small when compared to the message length.
ASYMMETRIC KEY CRYPTOGRAPHY USING MERKLE-HELLMAN KNAPSACK METHOD AND GENETIC ALGORITHM

Step 3: In Round 1, Convert the plaintext according to its ASCII value and apply Substitution using formula \( C = (P+k) \mod 256 \); where \( C \) is cipher text, \( P \) is plain text and key \( k \).

Step 4: Convert each character in the intermediate cipher 1 to its equivalent ASCII value. Intermediate cipher 1 is got from the Round 1.

Step 5: Apply two point crossover operations to the couple of character’s binary value until the last pair is reached.

Step 6: After performing crossover operation, Flipping of bits (mutation) operation is performed to each and every character separately.

Step 7: Convert the mutated value into decimal equivalent.

Step 8: Equivalent ASCII value is taken as the intermediate cipher 2.

Step 9: In Round 3, first take the intermediate cipher 2 and convert it into its corresponding ASCII value.

Step 10: Convert the ASCII value to its corresponding Binary Equivalent.

Step 11: The Round 3 starts from choosing superincreasing sequence of numbers of positive integers. A superincreasing sequence is one where every number is greater than the sum of all preceding numbers.

\[ s = (s_1, s_2, s_3, \ldots, s_n) \quad (1) \]

Step 12: Then convert all the characters of the message into binary. The binary sequence is represented by the variable \( b_t \). Step 13: 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 \( s_1, s_2, s_3, \ldots, s_n \) 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, \( p_i = r^*s_i \mod (a) \). Step 14: All elements \( p_1, p_2, p_3, \ldots, p_n \) 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 \( M_t \).

\[ M_t = \sum^n_{t=2} pt * bt \]

The sequence \( M = (M_1, M_2, M_3, \ldots, M_a) \) forms the public key of the cryptosystem.

Step 15: Final Cipher text is devised at the end of Round 3.

2. ENCRYPTION ALGORITHM

ALGORITHM

ENCRYPTION FLOWCHART

<table>
<thead>
<tr>
<th>Plain Text</th>
<th>Key (Temporary)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatic Key Generation Using S-DES Algorithm</td>
<td>Substitution Technique</td>
</tr>
<tr>
<td>Caesar Cipher</td>
<td>Intermediate Cipher1</td>
</tr>
<tr>
<td>( C = (P+k) \mod 256 )</td>
<td>Crossover and Mutation of Genetic Algorithm</td>
</tr>
<tr>
<td>Intermediate Cipher2</td>
<td>Transposition Technique</td>
</tr>
<tr>
<td>Knapsack Problem Method</td>
<td>Cipher Text</td>
</tr>
</tbody>
</table>

Fig. 6. Flowchart diagram for encryption algorithm

3. DECRYPTION ALGORITHM

Step 1: Take the Cipher text as the input.

Step 2: In Round 1, To decrypt the message \( M \), the recipient of the message would have to find the bit stream which satisfies the following equation

Maximize \( \sum^n_{t=1} pt * bt \)

\( M = \sum^n_{t=1} 1pt * bt \)
Step 3: To solve the above equation, the user would need the private key \((s, a, r)\).

Step 4: The first step is to calculate the modular multiplicative inverse of \(r\) in \(r \mod a\). This is calculated using the Extended Euclidean algorithm. This is denoted by \(r^{-1}\).

Step 5: The second step is to multiply each element of the encrypted message \((M)\) with \(r^{-1} \mod a\). The largest number in the private key which is smaller than the resulting number is subtracted from the number. This result continues until the number is reduced to zero.

Step 6: Convert the Binary equivalent to decimal equivalent.

Step 7: Convert each character into its equivalent ASCII value takes it as an Intermediate plain text 1.

Step 8: In Round 2, Take the Intermediate plain text. Apply flipping of bits (mutation) operation is performed to the binary value.

Step 11: Two point crossover is applied to the flipped bits an Intermediate plain text 2 is got from Round 2.

Step 12: In Round 3, take intermediate plain text 2 convert into its corresponding Numerical value.

Step 13: Apply Substitution formula \(P=(C-k) \mod 256\) (if the numerical value is negative apply the formula \(P=(C-k+256) \mod 256\) using the same key as chosen for encryption.

Step 14: Original text is retrieved at the end of Round 3. STOP.

4. DECRYPTION ALGORITHM FLOWCHART

V. CONCLUSION

The proposed algorithm improves the security across the communication channel and overcomes the difficulty of privacy issues through multiple stages of encryption should be taken. A symmetric (private) key is deduced from the SDES key generation algorithm and it is used for both encryption and decryption processes. It consumes the time for the entire work. In every encryption stage, binary to decimal or vice versa conversion should be made for making the cryptanalysis job tricky. A genetic programming concept modify each end every character of message according to the crossover and transposition techniques. This yields the tedious format of cipher text for the third party/hackers. In future, Automatic key generation algorithm has to be extended using hash key while enlarging their size or exploring the work in asymmetric key cryptosystem. This paper explained how to encrypt and decrypt data using the Martin-Hellman knapsack cryptosystem.

REFERENCES

ASYMMETRIC KEY CRYPTOGRAPHY USING MERKLE-HELLMAN KNAPSACK METHOD AND GENETIC ALGORITHM

80% minute-on-the-internet-640tb-data-transferred-
100k-tweets-93Hellman_knapsack_cryptosystem
204-million-e-mails-sent.html”, 2011.
ciphers.html”.
“A