Evaluation of the RC4 Algorithm for Data Encryption

Allam Mousa (1) and Ahmad Hamad (2)

(1) Electrical Engineering Department
An-Najah University, Nablus, Palestine
(2) Systems Engineer
PalTel Company, Nablus, Palestine
e-mail allam@najah.edu, ahmad.yasin@paltel.net

Abstract

Analysis of the effect of different parameters of the RC4 encryption algorithm where examined. Some experimental work was performed to illustrate the performance of this algorithm based on changing some of these parameters. The execution time as a function of the encryption key length and the file size was examined; this has been stated as complexity and security. Various data types were analyzed and the role of the data type was also emphasized. The results have been analyzed and interpreted as mathematical equations showing the relationship between the examined data and hence can be used to predict any future performance of the algorithm under different conditions. The order of the polynomial to approximate the execution time was justified.

Key words: RC4 cryptography, Stream Cipher, encryption key, file size, data type.

1. INTRODUCTION

Encryption is the process of transforming plaintext data into ciphertext in order to conceal its meaning and so preventing any unauthorized recipient from retrieving the original data. Hence, encryption is mainly used to ensure secrecy. Companies usually encrypt their data before transmission to ensure that the data is secure during transit. The encrypted data is sent over the public network and is decrypted by the intended recipient. Encryption works by running the data (represented as numbers) through a special encryption formula (called a key). Both the sender and the receiver know this key which may be used to encrypt and decrypt the data as shown in Fig.1.

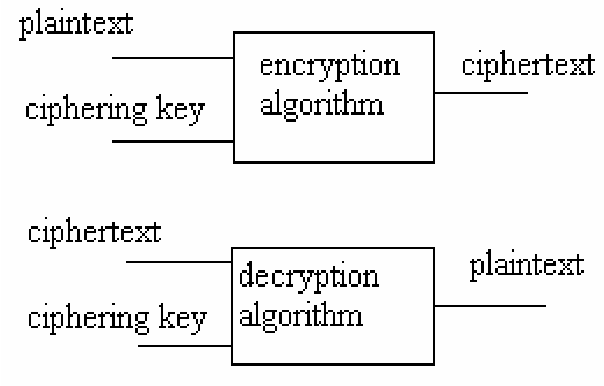

Fig.1 Encryption/Decryption Block Diagram

Cryptography is a tool that can be used to keep information confidential and to ensure its integrity and authenticity [2]. All modern cryptographic systems are based on Kerckhoff's principle of having a publicly-known algorithm and a secret key. Many cryptographic algorithms use complex transformations involving substitutions and permutations to transform the plaintext into the ciphertext. However, if quantum cryptography can be made practical, the use of one-time pads may provide truly unbreakable cryptosystems [3].

Cryptographic algorithms can be divided into symmetric-key algorithms and public-key algorithms. Symmetric-key algorithms mangle the bits in a series of rounds parameterized by the key to turn the plaintext into the ciphertext. Triple DES and Rijndael (AES) are the most popular symmetric-key algorithms at present. These algorithms can be used in electronic code book mode, cipher block chaining mode, stream cipher mode, counter mode, and others [3].

Public-key algorithms have the property that different keys are used for encryption and decryption and that the decryption key cannot be derived from the encryption key. These properties make it possible to publish the public key. The main public-key algorithm is RSA, which derives its strength from the fact that it is very difficult to factor large numbers [4].

Legal, commercial, and other documents need to be signed. Accordingly, various schemes have been devised for digital signatures, using both symmetric-key and public-key algorithms. Commonly, messages to be signed are hashed using algorithms such as MD5 or SHA-1, and then the hashes are signed rather than the original messages [3].

Public-key management can be done using certificates, which are documents that bind a principal to a public key. Certificates are signed by a trusted authority or by someone (recursively) approved by a trusted authority. The root of the chain has to be obtained in advance, but browsers generally have many root certificates built into them.

2. HOW ENCRYPTION WORKS

The encryption process involves taking each character of data and comparing it against a key. For example, one could encrypt the string “THE SKY IS HIGH” of data in any number of ways, for example, one may use a simple letter-number method. In this method, each letter in the alphabet corresponds to a particular number. If one uses a straight alphabetic to number encryption (i.e., A=1, B=2, C=3, and so on), this data is translated into the following numbers: 20 8 5 19 11 25 9 19 8 9 7 8. This series of numbers is then transmitted over a network, and the receiver can decrypt the string using the same key in reverse. From left to right, the number 20 translates to the letter T, 8 to H, 5 to E, and so on. Eventually, the receiver gets the entire message: “THE SKY IS HIGH”.

Most encryption methods use much more complex formulas and methods. The sample key was about 8 bits long; some keys are extremely complex and can be as large as 128 bits. The larger the key (in bits), the more complex the encryption and the more difficult it is to be cracked [4].

2.1. Encryption Keys

To encode a message and decode an encrypted message, one needs the proper encryption key or keys. The *encryption key* is the table or formula that defines which character in the data translates to which encoded character. Here, encryption keys fall into two categories: public and private key encryption [5].

2.2. Private Key Encryption

Private keys are also known as *symmetrical keys.* In private key encryption technology, both the sender and receiver have the same key and use it to encrypt and decrypt all messages. This makes it difficult to initiate communication for the first time. How does one securely transmit the single key to each user? However, public keys encryption is used [5].

2.3. Public Key Encryption

Public key encryption, or a Diffie-Hellman algorithm, uses two keys to encrypt and decrypt data: a public key and a private key. Public keys are also known as *asymmetrical keys*.

The receiver’s public key is used to encrypt a message then this message is sent to the receiver who can decrypt it using its own private key. This is a one-way communication. If the receiver wants to send a return message, the same principle is used. The message is encrypted with the original sender’s public key (the original sender is now going to be the receiver of this new message) and can only be decrypted with his or her private key. If the original sender does not have a public key, a message can still be sent with a digital certificate (also sometimes referred to as a digital ID). The digital ID verifies the sender of the message. Fig.2 shows public key– encrypted communication between two units, User X and User Y [5].

3. METHODS OF ENCRYPTION

There are a variety of different types of encryption methods, they can be classified according to the way in which the plaintext is processed (which can be either stream cipher or block cipher), or according to the type of operations used for transforming plaintext to ciphertext. The second class can be one of two styles, substitution (which maps each element in the plaintext into another element) and transposition (which rearranges elements in the plaintext).

Basically the two methods of producing ciphertext are stream cipher and block cipher. The two methods are similar except for the amount of data each encrypts on each pass. Most modern encryption schemes use some form of a block cipher [7].

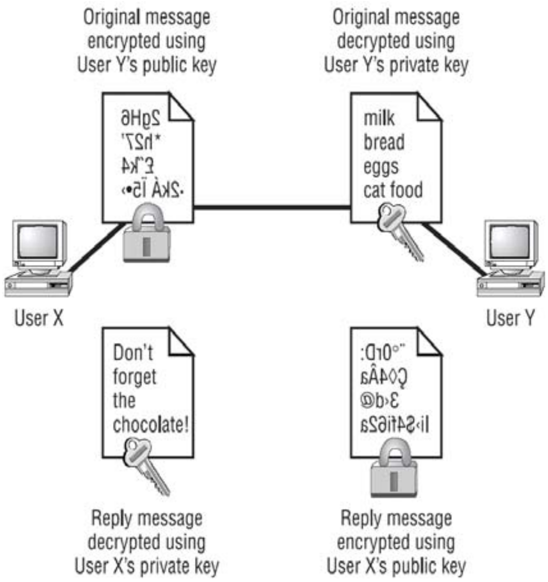

Fig.2 Public key encryption

3.1 Stream Cipher

Stream cipher is one of the simplest methods of encrypting data where each bit of the data is sequentially encrypted using one bit of the key as shown in Fig.3.

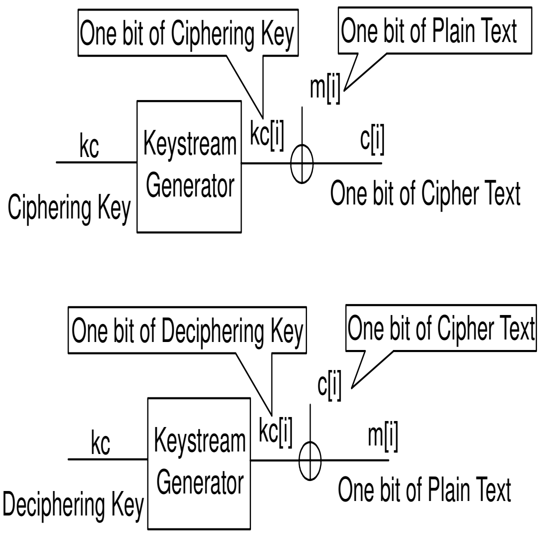

Fig.3 Stream ciphering and deciphering

In order to make a stream cipher more difficult to crack, one could use a crypto key which varies in length. This would help to mask any discernible patterns in the resulting ciphertext. In fact, by randomly changing the crypto key used on each bit of data, one can produce ciphertext that is mathematically impossible to crack. This is because using different random keys would not generate any repeating patterns which can give a cracker the clues required to break the crypto key. The main advantage of the stream cipher is that it is faster and more suitable for streaming application but its main disadvantage is that it is not suitable in some architecture. One example of the stream cipher method is the RC4 technique.

3.2 Block Cipher

Unlike stream ciphers, which encrypt every single bit, block ciphers are designed to encrypt data in chunks of a specific size as shown in Fig.4. A block cipher specification will identify how much data should be encrypted on each pass (called a block) as well as what size key should be applied to each block. For example, the Data Encryption Standard (DES) specifies that DES encrypted data should be processed in 64-bit blocks using a 56-bit key .

There exist a number of different algorithms that can be used when processing block cipher encryption. The most basic is to simply take the data and break it up into blocks while applying the key to each block. While this method is efficient, it can produce repetitive ciphertext. If two blocks of data contain exactly the same information, the two resulting blocks of ciphertext will be identical, as well; a cracker can use ciphertext which repeats in a nonrandom fashion to break the crypto key. Blowfish encryption technique is an example of the block ciphering.

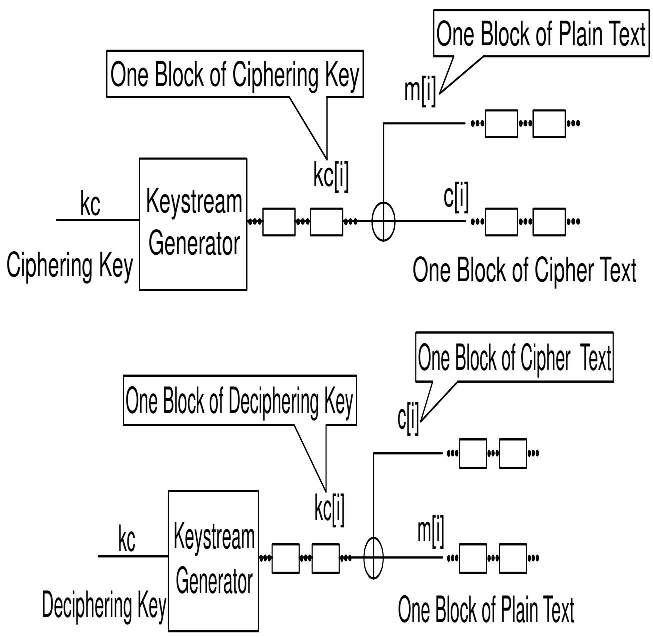

Fig.4 Block ciphering and deciphering

3.3 One Way Encryption

Another special type of encryption is the one way encryption, which is a method where the enciphering process is irreversible [3]. The plaintext can never be recovered from the cipher text. This may seem pointless but it is probably the form of encryption that is the most familiar to computer users. Passwords on UNIX systems are encrypted by a one way algorithm. When a password is first chosen it is enciphered and placed into permanent storage. When the user logs on, the password entered at the login prompt is encrypted by the same algorithm and it is the resultant cipher text that is compared with the ciphertext held on disk. An encrypted password can only be broken by somebody correctly guessing the password, an important reason why passwords should be carefully chosen.

3.4 Hybrid Systems

By combining public and private key cryptosystems, it is possible to overcome some of the disadvantages of each. Public key pairs are used to set up a secure session, and then data is exchanged using a secret key system. This provides both the security and authentication processes associated with public key systems and the bulk data encryption capabilities of secret key systems.

Pretty Good Privecy (PGP) is a well known security system used by computer enthusiasts to encrypt their email; it is an example of a practical hybrid encryption system which uses both secret key and public key [4].

4. RC4 ALGORITHM

RC4 is a stream cipher, symmetric key algorithm. The same algorithm is used for both encryption and decryption as the data stream is simply XORed with the generated key sequence. The key stream is completely independent of the plaintext used. It uses a variable length key from 1 to 256 bit to initialize a 256-bit state table. The state table is used for subsequent generation of pseudo-random bits and then to generate a pseudo-random stream which is XORed with the plaintext to give the ciphertext.

The algorithm can be broken into two stages: initialization, and operation. In the initialization stage the 256-bit state table, **S** is populated, using the key, **K** as a seed. Once the state table is setup, it continues to be modified in a regular pattern as data is encrypted. The initialization process can be summarized by the pseudo-code [6];

```
j = 0;
for i = 0 to 255:
S[i] = i;
for i = 0 to 255:
j = (j + S[i] + K[i]) mod 256;
swap S[i] and S[j];
```

It is important to notice here the swapping of the locations of the numbers 0 to 255 (each of which occurs only once) in the state table. The values of the state table are provided. Once the initialization process is completed, the operation process may be summarized as shown by the pseudo code below;

```
i = j = 0;
```

```
for (k = 0 to N-1) {
i = (i + 1) mod 256;
j = (j + S[i]) mod 256;
swap S[i] and S[j];
pr = S[ (S[i] + S[j]) mod 256]
output M[k] XOR pr
}
```

Where M[0..N-1] is the input message consisting of N bits.

This algorithm produces a stream of pseudo-random values. The input stream is XORed with these values, bit by bit. The encryption and decryption process is the same as the data stream is simply XORed with the generated key sequence. If it is fed in an encrypted message, it will produce the decrypted message output, and if it is fed in plaintext message, it will produce the encrypted version [6]. The RC4 encryption algorithm is shown in Fig.5.

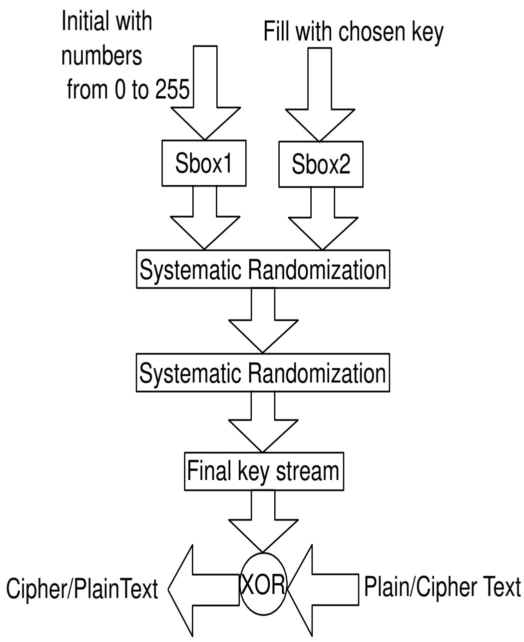

Fig.5 RC4 Encryption Algorithm

4.1. RC4 Steps

The steps for RC4 encryption algorithm is as follows:

1- Get the data to be encrypted and the selected key.
2- Create two string arrays.
3- Initiate one array with numbers from 0 to 255.
4- Fill the other array with the selected key.
5- Randomize the first array depending on the array of the key.
6- Randomize the first array within itself to generate the final key stream.
7- XOR the final key stream with the data to be encrypted to give cipher text.

Some of the RC4 algorithm features can be summarized as:
1- Symmetric stream cipher
2- Variable key length.
3- Very quick in software
4- Used for secured communications as in the encryption of traffic to and from secure web sites using the SSL protocol.

The RC4 algorithm was also tested when sound data was encrypted and decrypted. A typical wave file of size 136 KB is shown in Fig. 7. This file reads the words “one two three four five six” recorded by a male speaker, it was processed twelve times and the encrypted/decrypted time was recorded as shown in Table 3 for each execution time. This wave file was applied to the system and the encryption key length was allowed to vary. It has been shown that the encryption time is a function of the key length as shown in Fig. 8. The best fitting curve which represents this behavior was obtained as shown by Eq. 1.

<table>
<tr><th>The key(# of Characters)</th><th>The Text</th><th>Final Key Stream</th></tr>
<tr><td>10</td><td>Halloo</td><td>§_Ù_uËÍjdá-_sxŸ±
èÒä_{„QMe_˜.Ïqp34___ÈR÷
·_"«Ö /]¶º0O;k_Z#_N_wžÛ Šo°
òâ Å‡ß çV¾_K_ íY¡ª_,X¦v®´
³y úÔnæŒfÿÌSlÀ²|ôƒ…Â×
þ(Ýim_”µE_Þé@Ž=~I[9!ó™-
†_ØH}å½'õ¨
Bø)É‹ã€+¬__¸7<JFt Ð81Ã©Aöš_
Ú*¹^ÁÄ$?cDïa6_¢-&¤z’
¼ÊÆr_îðC-¯
LÜ5g»ê,Õìü£P2\ý¯TbÑ“</td></tr>
</table>

Table 1 example of a final key stream for RC4

$$y = -7E\text{-}06x^6 + 0.0002x^5 - 0.0027x^4 + 0.0164x^3 - 0.0506x^2 + 0.077x + 0.6707 \quad (1)$$

Where (x) is the length of the encryption key in (characters) and (y) is the time for encryption in seconds.

The order of the polynomial could be chosen differently, but it was selected in a way such that the curve passes by the maximum number of points with the minimum possible order.

The performance of the RC4 is tested here based on the processing time under certain conditions. However, this may vary according to the processor and the software used to implement the system. Hence, a comparison with another encryption algorithm may be used. The Blowfish encryption algorithm is referred to for this case [1]. The two algorithms were executed under identical conditions. Accordingly, the encryption time for the varying size wave file is shown in Fig. 9 for both RC4 and Blowfish encryption algorithms where the key length is kept fixed. It is clear that the two algorithms have almost the same performance in terms of execution time.

Key Length	Cipher Text
1	-yå-ÌX
2	)}1ø¥¬
3	ø¶Ú_ßÌ
4	•_Æ’w
5	iLa\ r
6	1_ç#_S
7	,b°˜=§
8	’>Í•=
9	9]-Yi•
10	´9_×O

Table 2 Typical key length effect on encrypting a short data

Order	Encryption time	Decryption time
1	5202935	5009464
2	5017437	5039983
3	5009278	5034061
4	5358964	5029369
5	5074751	5149366
6	5151504	5064671
7	5266730	5008304
8	5015440	5062030
9	5217584	5041085
10	5198026	5102031
11	5028613	5028713
12	5140114	5051734

Table 3 The processing time of a typical wave file in microseconds

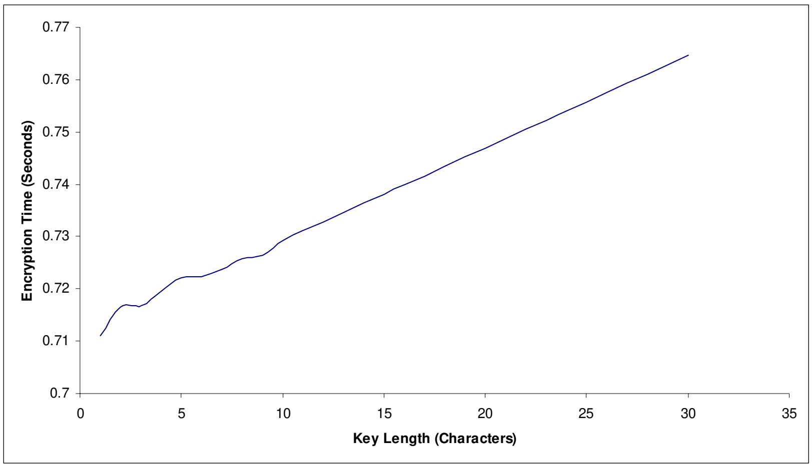

Fig.6 Encryption time versus variable encryption key length for a small text data.

Utilizing these results, a module of the relationship between the file size and the processing time can be illustrated as given by Eq.2 and Eq.3 for the RC4 and Blowfish algorithms respectively.

$$y = 0.0003x^2 - 0.0058x + 0.2621 \qquad (2)$$

$$y = 3E\text{-}06x^3 - 0.0005x^2 + 0.0594x - 1.4331 \qquad (3)$$

Where (x) is the size of the wave file to be encrypted in kilo byte (KB) and (y) is the time for encryption in seconds.

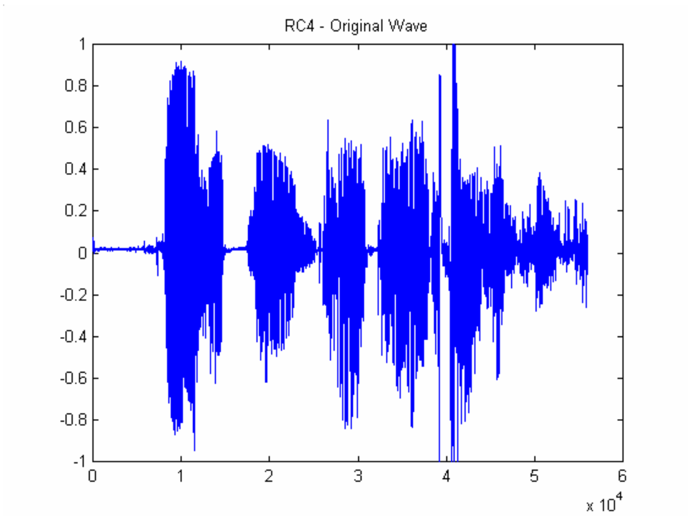

Fig. 7 A typical wave file that is used.

The order of the polynomial here is different from that in Eq.1 since a second or third order polynomial for this case can achieve the best fitting curve as desired. Noting that the third order in Eq.3 has a very small coefficient (approaching zero) making Eq.2 and Eq.3 so comparative.

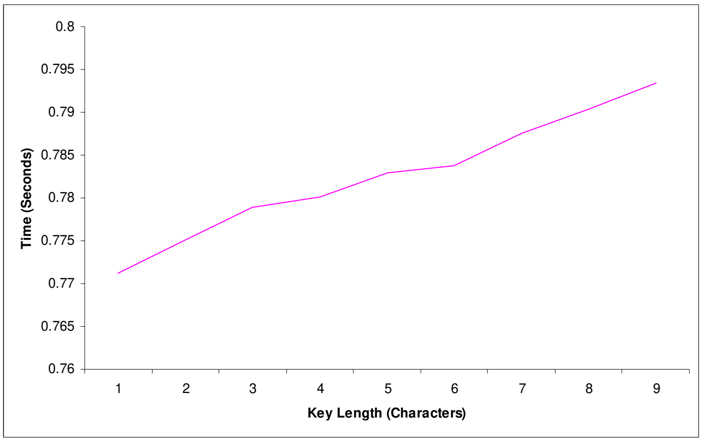

Fig. 8 Key length effect on the encryption time for a wave file.

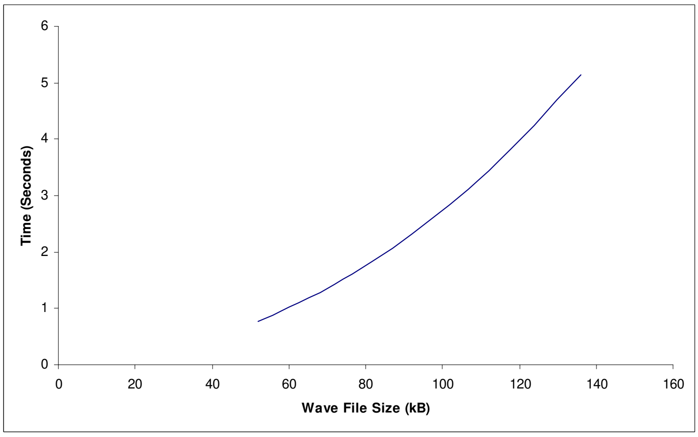

Fig.9 Wave files size effect on the processing time for both RC4 and Blowfish.

Similarly, the image data type was also examined, the file size was allowed to change and the curves which are best fitting to module the encryption time for RC4 and Blowfish are given by Eq.4 and Eq.5 respectively when the same data was processed by the two algorithms. The two equations were set to an order of 3 where this order was good enough to produce a best fitting curve while simplifying the comparison between the two algorithms.

$$y = -9E\text{-}08x^3 +0.0018x^2 -0.0645x +0.3796 \qquad (4)$$

$$y = -3E\text{-}07x^3 +0.0019x^2 -0.064x +0.3722 \qquad (5)$$

Where (x) is the size of the image file in Kilo bytes (KB) to be processed and (y) is the execution time in seconds.

Although the algorithm is not a data dependent, but the same exact equations could not be obtained for the wave and image data type, this is because the file size were not identical in both cases and so the executions time were not the same for the files which leads to different points to consider in best fitting curves.

The variable key length and variable file size are also important in the decryption process. Simulation results obtained here for the relation between the variable key length and the decryption time for a sound data executed by both RC4 and Blowfish algorithms were obtained and modeled as given by Eq. 6 and Eq. 7 and that for the relation between the ciphertext file size and the decryption time is illustrated by Eq. 8 and Eq. 9 respectively.

$$y = 1E\text{-}06x^6 -4E\text{-}05x^5 +0.0005x^4 -0.0031x^3 +0.0088x^2 -0.007x +0.7721 \qquad (6)$$

$$y = -8E\text{-}07x^6 + 2E\text{-}05x^5 - 0.0003x^4 + 0.0013x^3 - 0.0015x^2 + 0.0009x + 0.7763 \qquad (7)$$

Where (x) is the length of the decryption key in characters and (y) is the time for decryption in seconds.

$$y = 0.0003x^2 - 0.0053x + 0.2669 \qquad (8)$$

$$y = 3E\text{-}06x^3 - 0.0005x^2 + 0.0603x - 1.4542 \qquad (9)$$

Where (x) is the size of the wave file to be decrypted in kilo byte (KB) and (y) is the time for decryption in seconds.

The RC4 decryption time for an image file as a function of the key length is moduled as given by Eq. 10 and that versus the file size is illustrated by Eq. 11.

$$y = -4E\text{-}07x^3 + 0.0018x^2 -0.0675x +0.4091 \qquad (10)$$

$$y = -8E\text{-}09x^3 -0.0018x^2 -0.0609x -0.3625 \qquad (11)$$

6. CONCLUSIONS

Analyzing the RC4 parameters have shown that the speed of encryption or decryption time is directly related to the encryption key length and to the data file size if the data is large enough. Data type is also important since image data reqires larger time to be processed than text or sound data mainly due to the larger file size. This relationship had been converted into equations to model these relationships and so may be used to predict the performance of the RC4 under different conditions.

7. AKNOWLEDGEMENT

The authors would like to thank the anonymous reviewers for their helpful suggestions that considerably improved the quality of this paper.

8. REFERENCES

[1] Allam Mousa, Data Encryption Performance Based on Blowfish, 47th International Symposium ELMAR-2005 focused on Multimedia Systems and Applications, pp. 131-134, Zadar, Croatia, 08-10 June 2005.
[2] Andrew S. Tanenbaum, Computer Networks, Fourth Edition, Prentice Hall, 2003
[3] David Groth, Network+ ™, Study Guide, Third Edition, SYBEX, Inc., Alameda, CA, 2002
[4] Glover, P. and M. Grant, Digital Communications, 2nd edition, Person Education, 2004
[5] Joset Pieprzyk, et. al., Fundamentals of Computer Security, Springer, 2003
[6] William Stallings, Cryptography and network security: Principles and practice, Prentice Hall, Upper Saddle River, New Jersey, 2003
[7] Wenbo Mao, Modern Cryptography Theory and Practice, Prentice Hall, New Jersey, 2004