A New Hybrid Data Encryption and Decryption Technique to Enhance Data Security in Communication Networks: Algorithm Development

Adedeji Kazeem B. and Ponnle Akinlolu A.

Abstract— Data security is an important aspect of communication system that has always been a focus for exchanging information among parties at location physically apart. In today's competitive market, different techniques have been developed to send data securely. We present a hybrid technique which combines the speed of Data Encryption Standard (DES) for encryption of data and Rivest Shamir Adleman (RSA) algorithms to encrypt DES secret key for proper key distribution and management. The proposed technique was implemented in Microsoft visual C# and the performance of the hybrid technique was assessed based on encryption and decryption time and throughput for different input text and image data sample of varying sizes. The results obtained shows that the developed hybrid technique has better performance in terms of speed of encryption (encryption time), throughput and the central processing unit (CPU) power consumption, when compared to other hybrid technique in literature. Hence, the developed hybrid technique is recommended for enhancing data security in modern applications, and systems where a high speed of encryption is required, without compromising the CPU power consumption.

Index Terms— Ciphertext, Cryptography, Encryption, Hybrid, Key Size, Security, Throughput.

1 Introduction

Over the years, the market for wireless communications has enjoyed tremendous growth. Also, with tremendous growth in technology, wireless communications is being applied to the realm of personal and business computing; therefore protection of data from misuse is essential. Security is one of the important aspects in computing and data communication. Data transfer is transferring data from a location or host to another host, or server. To have a secure data transfer, few methods of cryptography can be applied [1]. Cryptography gives the means to construct a secure logical channel over an insecure physical connection [2].

One of the methods of cryptography is encryption of data, prepared to be transferred in encrypted way and decrypted when the data is to be used. Encryption is the process of transforming plaintext data into cipher text to conceal its meaning thereby preventing any unauthorized recipient from retrieving the original data [3]. This is usually done with the use of an encryption key, which specifies how the message is to be encoded. Any adversary that can see the cipher text should not be able to determine anything about the original message. Encryption works by running the data through a special encryption formula called a key [4]. Decryption is the reverse process of converting encrypted data to its original un-encoded form. This process is shown in Fig. 1.

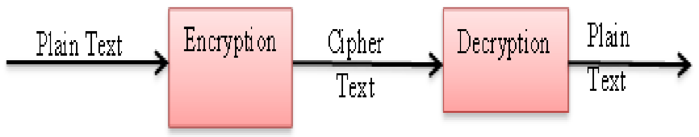

Fig. 1: Encryption and Decryption process (adopted from [5])

The aim of cryptography is not to hide the existence of a message, but rather to hide its meaning, through a method of encryption.

Cryptography is an algorithmic process of converting a data (plain text) to a cipher text or cipher message based on an algorithm that both the sender and the receiver know, so that the cipher text message can be returned to its original, plain text form. In its cipher form, a message cannot be read by anyone but the intended receiver. The act of converting a plain text message to its cipher text form is called enciphering. Reversing that act (i.e., cipher text form to plain text message) is deciphering. Enciphering and deciphering are more commonly referred to as encryption and decryption, respectively.

The encryption process can be expressed mathematically as;

$$C = E(K, P) \tag{1}$$

where *C* is the cipher text; *E*(.), is the encryption function; *K* is the key and *P*, is the Plain text. The encryption function *E*(.), operates on *P*, to produce *C*. The decryption process on the other hand is given by

$$P = D(K, C) = D(K, E(K, P)) \tag{2}$$

where *D*(.), is the decryption function. The decryption function *D*(.), operates on *C*, to produce *P*.

There are three basic types of cryptographic techniques namely; Symmetric cryptographic technique, asymmetric cryptographic technique and hash function. Symmetric technique uses a single key for both encryption and decryption of data. Data encryption standard (DES), advance encryption standard

(AES), Carlisle Adams and Stafford Tavares (CAST) Algorithm, Blowfish, Twofish, international data encryption algorithm (IDEA) and Secure and Fast Encryption Routine (SAFER) are some examples of symmetric technique [6]. In an asymmetric technique, two different keys that are mathematically related are used. It uses one key (public key) for encryption and another (private key) for decryption. The public key is widely distributed while the private key is secret, which is known to the receiver only. Rivest Shamir Adleman algorithm (RSA), Diffie-Hellman, digital signature algorithm (DSA), Elgama and elliptic curve cryptography (ECC) are some public key cryptographic systems [7]. The hash function uses a mathematical transformation to irreversibly "encrypt" information. This type of system includes message digest (MD5), SHA-1, SHA-2 [8]. The RSA algorithm at present is the most successful in use for ciphering keys and passwords or counts. The key length varies from 64 to 1024 bits [9].

Various studies and research has been conducted to analyze and study the comparative performance of algorithms. Himani and Manisha, (2012) analyzes various symmetric encryption techniques and compare them on points such as execution time and avalanche effects by varying key or plain text [10]. Jigar *et al*, (2013) worked on enhancing data security by using Hybrid Cryptography Algorithm [11]. In their work they make use of two symmetric cryptographic algorithms (AES-DES) to form a Hybrid system. Their result shows that the developed hybrid model has higher encryption time than ASE and DES counterparts; hence it will take a longer time to be broken by cryptanalyst than DES or AES alone. Their result was in conformity with the one obtained by Jignesh *et al*, (2012) who worked on Hybrid Security Algorithm for Data Transmission using AES-DES [12]. The limitation of both works is that of key management problem because two symmetric techniques were used. Based on the review of past work on various data encryption techniques; it can be seen that DES has better performance in terms of execution time, throughput and power consumption and RSA algorithm, at present is the most successful in use for ciphering keys and passwords or counts. If both algorithms are combined to form a hybrid technique, this will enhance data security in communication networks.

- *Adedeji Kazeem Bolade is currently pursuing master's degree program in Electrical and Electronics Engineering, Federal University of Technology, Akure, Nigeria. PH-+2348059187995. E-mail: kezman0474@yahoo.com*

- *Ponnle Akinlolu Adediran (PhD) is a lecturer in Electrical and Electronics Engineering Department, Federal University of Technology, Akure, Nigeria. PH-+2348103217173. E-mail: ponnleakinlolu@yahoo.co.uk*

2 DESIGN METHODOLOGY

The development of the proposed hybrid technique involves three main stages namely; the Data Encryption Standard (DES) implementation; the Rivest Shamir Adleman (RSA) implementation; and the hybrid (DES-RSA) implementation.

2.1 The DES Algorithm

The basic process in encrypting a 64 bit data with a 56 bit key using the DES algorithm as shown in Fig. 2, consist of three main stages according to Jignesh *et al*, (2012) [12];

(i) An initial Permutation;
(ii) 16 rounds of a complex key dependent computations;
(iii) A final permutation, being the inverse of the initial permutation.

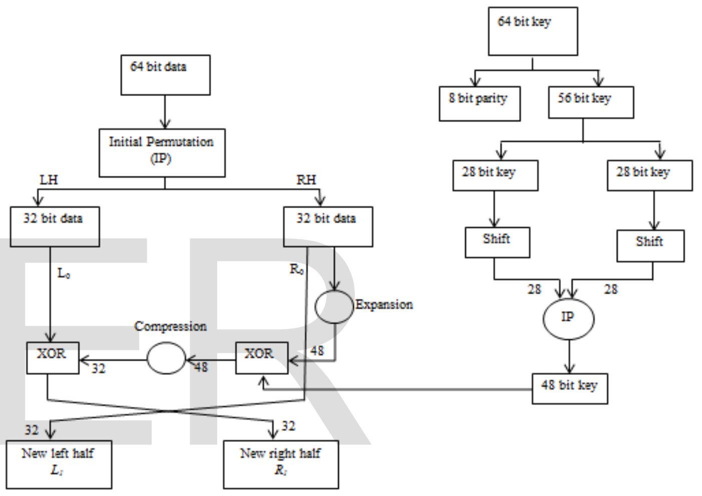

Fig. 2: Encryption process of DES (source: [11])

DES algorithm takes an input of 64-bit long plaintext (or a multiple of 64 bits) data block and 56-bit key (8 bits of parity) and generates output of 64-bit block of ciphertext. If the input data was less than or greater than 64 bits, it pad the last block of such input data with some regular pattern of zeros, ones, or alternating ones and zeros, to make it a complete block (64 bit block or a multiple of 64 bits). The plaintext block was then subjected to an Initial Permutation (IP) to shift the bits around. The 8 parity bits were removed from the key by subjecting the key to its Key Permutation thereby reducing the 64 bit key to 56 bits. After initial permutation, the plaintext and key are processed in 16 rounds of operations giving below;

(i) the key was split into two 28-bit halves.
(ii) each half of the key was shifted (rotated) by one or two bits, depending on the round.
(iii) the halves were recombined and subjected to a compression permutation to reduce the key from 56 bits to 48 bits. This compressed key was used to encrypt this round's plaintext block.

(iv) the rotated key halves from (ii) were used in next round.
(v) the data block was splitted into two halves; the left half (LH) and the right half (RH), with both halves having a 32 bit data block.
(vi) one half (RH) was subjected to an expansion permutation to increase its size to from 32 bits to 48 bits.
(vii) the output of step (vi) was XOR with the 48-bit compressed key from (iii)
(viii) the output of step (vii) was now compressed to reduce the 48-bit block down to 32-bits.
(ix) the output of step (viii) was XOR with other half (LH) of the data block.
(x) the two data halves were swapped and become the next round's input.

This process was repeated for sixteen times to get sixteen rounds of key dependent operations. The i^{th} round of the computation is describe as by (3) and (4)

$$L_i = R_{i-1} \quad \text{for } 1 \le \text{i} \le 16 \tag{3}$$

$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i) \quad \text{for } 1 \le \text{i} \le 16 \tag{4}$$

where L is the left half, R, is the right half and K, is the key. i, is an integer showing the number of rounds; whose values ranges from 1 to 16.
The result of the 16th round was reversed, obtaining the sequence $R_{16}L_{16}$. After the 16th round, The 32 bits sequence of the right half and the left half were combined together and was subjected to a final permutation (inverse of the initial permutation) to produce the cipher text.

2.2 The RSA Algorithm

The process involved in RSA algorithm is about key generation and is given by the flow chart in Figure 3. The RSA Key Generation Algorithm involves the steps listed below;

(i) random numbers were first generated by the program using Pseudo random number generator.
(ii) the algorithm selects two distinct large prime numbers p and q.
(iii) these chosen numbers were checked whether the numbers is a prime using primality test. A lot of primality testing algorithm was available in literature, but for this research the primality test command in C# was used. If p and q passes the primality test, then the algorithm proceed to stage (iii), otherwise, it returns to stage (ii) to select another p and q.

For security, p and q must be of the same length in bits, they must not be equal and they should not be too close to each other for security, that is, p-q should not be a small number [13]. After p and q have passed the primality test, the algorithm moved to the next step, that is, stage (iv)

(iv) the product of p and q were computed and attributed to n, that is, $n = p \times q$
(v) the Euler's totient function $\Phi(n) = (p-1) \times (q-1)$, was also computed
(vi) the algorithm then chooses an integer e (encryption key), such that $1 < e < \Phi(n)$, and $\gcd(e, \Phi(n)) = 1$. This means that e and $\Phi(n)$ must share no common factors other than 1, that is, e is relatively prime to $\Phi(n)$.

The choice of e are 3,5,17,257 and 65537 which were derived from the Fermat theorem given by (5),

$$F_X = 2^{2^x} + 1 \tag{5}$$

where x ranges from $(0..........N-1)$. The first five Fermat numbers F_0, F_1, F_2, F_3, F_4 are called Fermat primes and the corresponding values were given above .The number F_5 and above are not prime.
In order to increase the efficiency of encryption, one can select a small encryption exponent e; in practice $e = 3$ or $e = 23567$, is commonly used. The RSA encryption algorithms with small exponent e, are significantly faster [1].

(vii) after a value for e, have been successfully choosed, the algorithm computes the private (decryption key) d, to satisfy the Extended Euclidean Algorithm given as $d \times e = 1 \bmod(\Phi(n))$. From the expression,

$$d = e^{-1} \bmod(\Phi(n)) \tag{6}$$

(viii) The pubic key is (n, e) and the private key is (n, d). The values of d, e, p, q and $\Phi(n)$ must be kept secret.

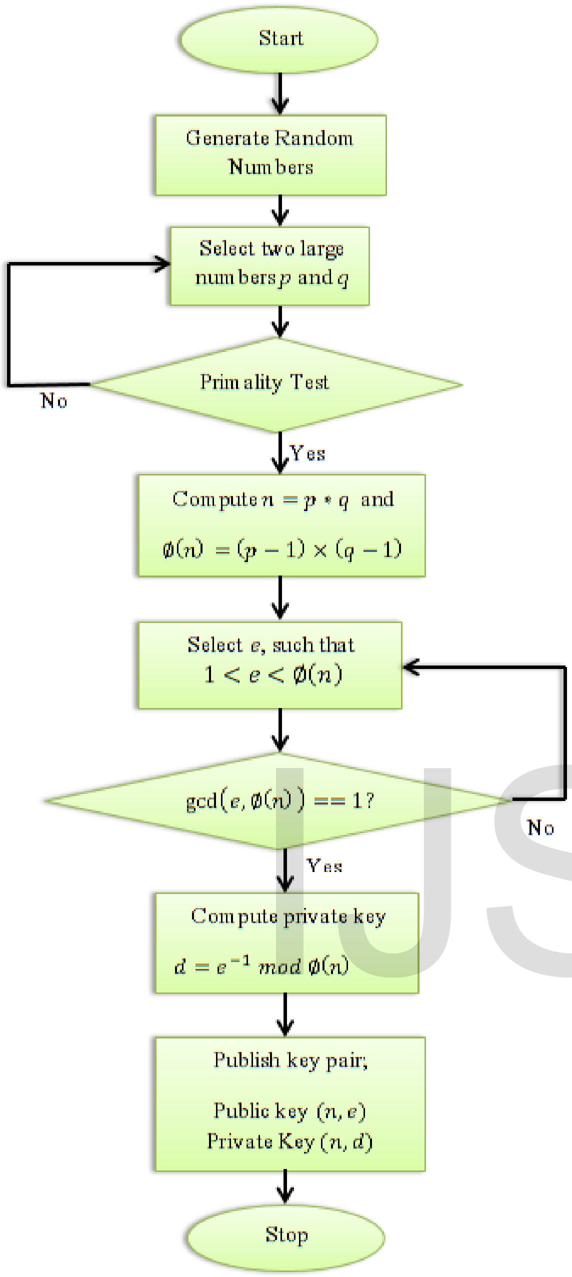
Start
Generate Random Numbers
Select two large numbers p and q
Primality Test
No
Yes
Compute $n = p * q$ and $\emptyset(n) = (p-1) \times (q-1)$
Select e, such that $1 < e < \emptyset(n)$
$\gcd(e, \emptyset(n)) == 1?$
No
Yes
Compute private key $d = e^{-1} \bmod \emptyset(n)$
Publish key pair; Public key (n, e) Private Key (n, d)
Stop

Fig. 3: RSA Key Generation

2.3 The Hybrid Technique Implementation

The block diagram of the developed hybrid technique is shown in Fig. 4. In this technique, at the transmitting side, Data Encryption Standard (DES) encryption algorithm was used to encrypt the data to be transmitted (plain text P) with the help of a randomly generated session key k, turning it into a cipher text, C. The resulting cipher text is given by

$$C = E(k, P) \tag{7}$$

The session key was generated using pseudorandom number generator. Since DES is a secret key system, this key has to be kept secret and this is achieved by encrypting the session key using Rivest Shamir Adleman (RSA) algorithm with the help of the recipient public key, e. This produces an encrypted session key u, given by (8),

$$u = k^{e}(\mathrm{mod}(n)) \tag{8}$$

where n is the product of two randomly generated large prime numbers p and q used in the RSA algorithm. The cipher text C, and the encrypted session key u, were then sent to the receiver over a communication channel.

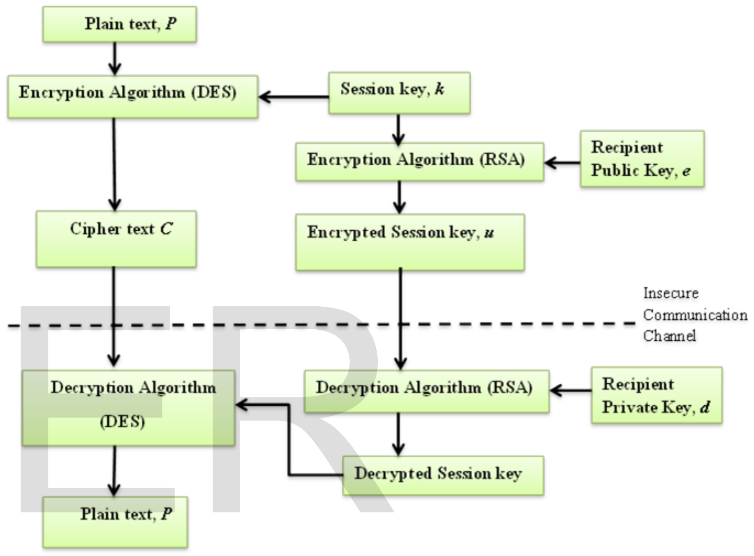
Plain text, P
Encryption Algorithm (DES)
Session key, k
Encryption Algorithm (RSA)
Recipient Public Key, e
Cipher text C
Encrypted Session key, u
Insecure Communication Channel
Decryption Algorithm (DES)
Decryption Algorithm (RSA)
Recipient Private Key, d
Decrypted Session key
Plain text, P

Fig. 4: The developed Hybrid Technique

At the receiving end, the recipient private key d, was used with the RSA decryption algorithm to decrypt u, thereby producing the session key as given by (9),

$$k = u^{d}(\mathrm{mod}(n)) \tag{9}$$

Having retrieved the session key back; this session key was used with the DES decryption algorithm to obtain the original data (plain text P) according to (10),

$$P = D(C, k) = D(E(P, k), k) \tag{10}$$

The hybrid technique was implemented using C sharp language on a computer system that has the following system parameters;

1. Processor: Intel (R) Atom CPU N270 at 1.60GHz
2. Installed Memory (RAM): 1GB
3. Operating System: Windows 7 Ultimate, 32-bit.

Ten (10) different data samples of text with (.txt) extension and images with (.jpg, jpeg and png) extension, of varying sizes were entered into the hybrid system. There are options for DES, RSA and the Hybrid technique so that algorithm type can be selected to encrypt and decrypt data. The encryption and decryption time for each data size, were recorded for each technique. The different metrics that were used to evaluate the performance of the system includes; the encryption and decryption time (in millisecond), throughput (in MB/Sec), the average data rate (in KB/Sec), and central processing unit (CPU) power consumption.

The average data rate (*ADR*) was calculated using (11), given by [6]

$$ADR = \frac{1}{N_b}\sum_{i=1}^{N_b}\frac{M_i}{t_i} \qquad (11)$$

where N_b is the number of message to be encrypted/decrypted, M_i is the message size (kb) and t_i is the time taken to encrypt/decrypt message M_i (in millisecond).
The encryption ratio (ER) was computed using [6], giving by

$$ER = \sum\frac{L_y}{Lx} \qquad (12)$$

where L_y, is the size of the encrypted data (the ciphertext) and L_x, is the size of the original plain text.
The throughput *T*, of the system was then computed using [6]

$$T = {}^{P_t}\!/\!{}_{E_t} \qquad (13)$$

Where P_t, is the size of the total plaintext to be encrypted/decrypted (in MB) and E_t, is the total encryption/decryption time (in millisecond). Throughput of an algorithm is inversely proportional to the CPU power consumption, so the throughput was used to calculate the CPU power consumption of the developed technique.

3 RESULTS AND DISCUSSIONS

The Data Encryption Standard (DES) and the proposed Hybrid technique were been developed and implemented in Microsoft Visual C# environment. Fig. 5 and Fig. 6 show the plot for the encryption and decryption time for the various sizes of the input text data samples for both DES and the developed hybrid technique. From Fig. 5, it can be observed that, the developed hybrid technique has a little bit higher encryption time than DES especially for file size above 1MB. This means that it takes a little more time to encrypt data with the developed hybrid technique than encrypting with DES only. This is as a result of computational complexity of the hybrid technique since it combines two algorithms together. The decryption times for the two techniques are almost the same.

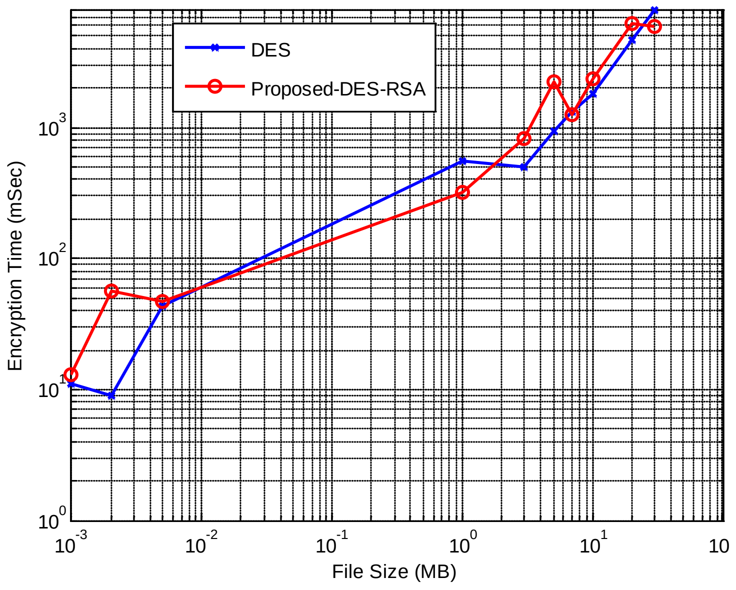

Fig. 5: Encryption time for both DES and the Hybrid technique for text data samples (Both axes are logarithmic).

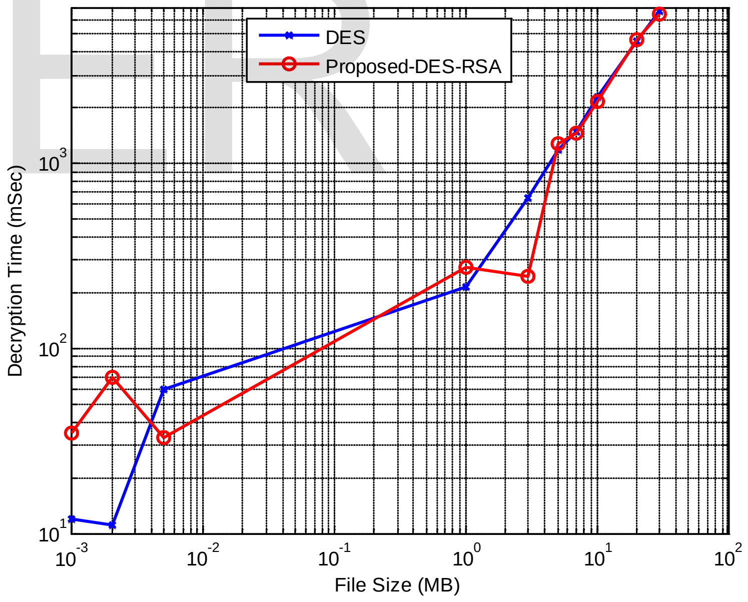

Fig. 6: Decryption time for both DES and the Hybrid technique for text data samples (Both axes are logarithmic).

Fig. 7 and Fig. 8 respectively show the encryption and decryption time for various sizes of image data samples for both DES and the developed hybrid technique. From the figures, it is observed that for image data samples, the hybrid technique has higher encryption and decryption times than the DES, which is also significant.

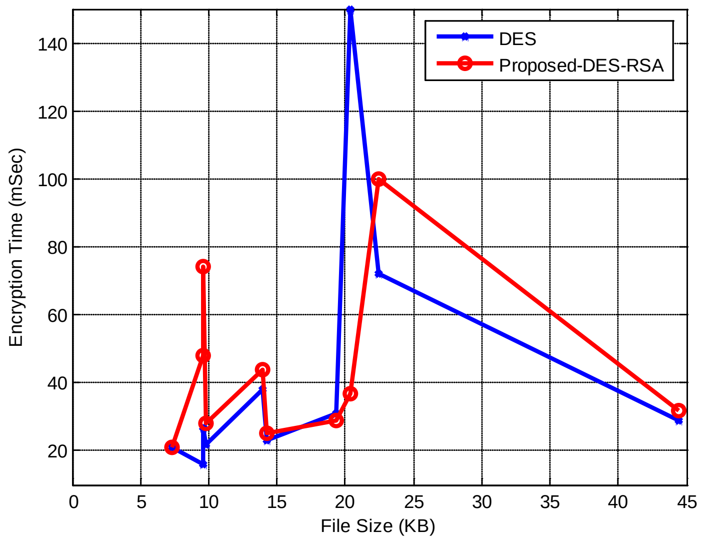

Fig. 7: Encryption time for both DES and the Hybrid technique for image data samples.

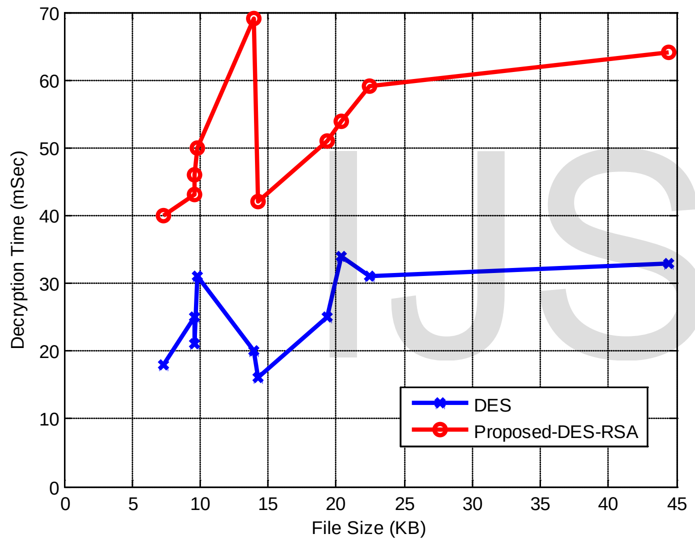

Fig. 8: Decryption time for both DES and the Hybrid technique for image data samples.

Table 1 and Table 2 show the results for the throughput of encryption and decryption, average data rate of encryption and decryption for the different data packet size of plaintext (txt) and images (jpg, jpeg, and png) respectively, for both DES and the developed hybrid technique.

From Table 1, it can be observed that the throughput of the hybrid technique for encryption of data is lower than that of DES alone, but for decryption, its throughput is higher than that of DES. The implication of this is that, the hybrid technique will consume or draw more power from the central processing unit (CPU) during encryption, but lesser power during decryption. This is because the throughput of an algorithm is inversely proportional to the CPU power consumption. It can also been seen from the table that both techniques have the same encryption ratio, which indicate a little increase in the size of the ciphertext with respect to the size of the input data. Both techniques produce the same size of ciphertext after file encryption.

Table 1: Other performance criteria used for encryption and decryption for text file input

Performance Criteria	**DES**	**Hybrid (DES-RSA)**
Encryption throughput (MB/Sec)	4.29	3.93
Decryption throughput (MB/Sec)	4.42	4.55
Average data rate of encryption (KB/Sec)	3251.59	2718.14
Average data rate of decryption (KB/Sec)	3169.14	3827.01
Encryption ratio	1.337	1.337

Table 2: Other performance criteria used for encryption and decryption for image file input

Performance Criteria	**DES**	**Hybrid (DES-RSA)**
Encryption throughput (MB/Sec)	0.399	0.391
Decryption throughput (MB/Sec)	0.674	0.330
Average data rate of encryption (KB/Sec)	533.82	474.43
Average data rate of decryption (KB/Sec)	659.20	318.27
Encryption ratio	1.918	1.918

From Table 2, it can be observed that the throughput of the hybrid technique for encryption of data is a little bit lower than that of DES alone, but for decryption, it is much higher in with a wide margin than that of DES. The value of the average data rate for the tested data was also low compared to that of DES for both encryption and decryption. This means that, for the image data samples, the developed hybrid technique will consume less memory for encryption and decryption operation. Its encryption ratio is also the same with that of DES. The average data rate indicates the amount of information or data encrypted or decrypted per second. A higher value of this consumes more CPU memory usage. According to Aman *et al*, (2012), an algorithm is considered best which use small memory and perform best task [14].

Fig. 9 shows the comparison of the encryption time for the developed technique with other hybrid techniques, implemented with the same input data type and sizes, as reported by Naser *et al.*, (2013) [3].

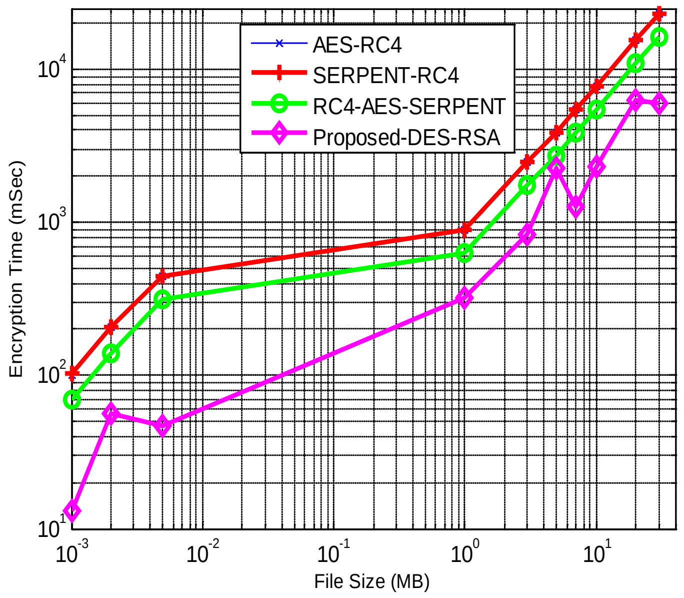

Fig. 9: Encryption time for the developed hybrid technique with other hybrid techniques for test data input (Both axes are logarithmic).

From the figure, it can be observed that, the developed hybrid technique has lower encryption time than all other hybrid techniques in Fig. 10. This means that it takes the developed hybrid (DES-RSA), a shorter time to encrypt data than other hybrid techniques indicated. From this, it can be concluded that the hybrid (DES-RSA) is faster than other hybrid techniques experimented with this work (AES-RC4, SERPENT-RC4, RC4-AES-SERPENT).

Fig. 10 shows the throughput of encryption of the developed hybrid technique in comparison with other hybrid techniques.

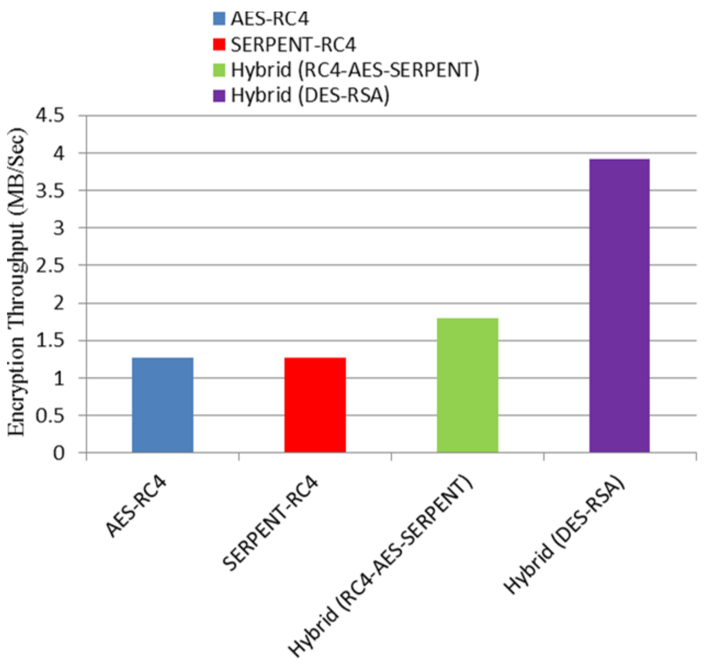

Fig. 10: Throughput of encryption for the developed hybrid with other hybrid techniques

From the figure, it can be seen that the throughput of the developed hybrid (DES-RSA) technique is better than that of other hybrid techniques examined, as it has higher value than all of them except DES.

The percentage increase in values for the throughput of the developed hybrid technique and other hybrid systems is given by

$$\%CT = \frac{T_P - T_O}{T_P} \times 100\%$$

where *%CT*, is the percent increase in throughput, T_p is the throughput for the proposed hybrid technique and T_o is the throughput for other hybrid techniques.

From Fig. 10, the percent increase in throughput of the proposed hybrid technique (from AES-RC4 and SERPENT-RC4) is given as

$$\%CT = \frac{3.926 - 1.270}{3.926} \times 100\% = 68\%$$

Similarly, the percent increase in throughput of the proposed hybrid technique (from RC4- AES-SERPENT) is given as

$$\%CT = \frac{3.926 - 1.80}{3.926} \times 100\% = 54\%$$

This gives a clear indication that the proposed hybrid (DES-RSA) consumes less power of CPU than others.

4 CONCLUSION

The combined concepts of DES and RSA to form a hybrid technique have been achieved and presented. The performance of the developed hybrid (DES-RSA) have been evaluated and compared to other hybrid systems (AES-RC4, SERPENT-RC4 and RC4-AES-SERPENT), using the same packet size of text data samples (ranging from 1KB to 30MB) on the same computer system. The developed hybrid (DES-RSA) technique encrypts data faster than hybrid AES-RC4, SERPENT-RC4 and RC4-AES-SERPENT. It was able to generate secured ciphertext in a short time compare to others.

Also, the developed hybrid (DES-RSA) technique has a better throughput than the other hybrid techniques; 54% better than RC4-AES-SERPENT and 68% better than both of AES-RC4 and SERPENT-RC4. The developed hybrid technique is better in terms of speed of encryption, throughput and CPU power consumption usage. Therefore, it can be concluded that the developed hybrid technique will be a suitable choice for encrypting data in modern applications and for systems where a high speed of encryption is required, without compromising the CPU power consumption.

REFERENCES

[1] H. Darrel, M. Alfred and V. Scott, “Guide to Elliptic Curve Cryptography”, Springer-verilag Inc., New York, pp. 2-12, 2004.

[2] T. Charomie, “Implementation of Hybrid Encryption Method using Caesar Cipher Algorithm”, Dissertation submitted to Faculty of Computer System & Software Engineering Universiti Malaysia Pahang (UMP), Malaysia, 2010.

[3] A. Naser, H. Fatemeh and K. Riza, “Developing a new hybrid cipher using AES, RC4 and SERPENT for encryption and Decryption”, *International Journal of Computer Applications,* vol. 69, no. 8, pp.53-62, 2013.

[4] S. William, “Cryptography and Network Security”, 4th Edition, Prentice-Hall Inc., pp.58-309, 2005.

[5] T. Abdel-Karim, 2012, *“Performance Analysis of Data Encryption Algorithms”*, retrieved 14th June, 2013 from http://www.cse.wustl.edu/~jain/cse56706/encryption_perf.htm

[6] K. M. Anand and S. Karthikeyan, “Investigating the Efficiency of Blowfish and Rejindael (AES) Algorithms”, *International Journal of Computer Networks and Information Security,* vol. 2, pp. 22-28, 2012.

[7] G.A. Marin, “Network Security basics”, *IEEE Security and Privacy,* vol. 1, no. 3, p 68, 2005.

[8] M. Edney, “Real 802.11 Security: Wi-Fi Protected Access and 802.11i”, Addison Wesley Inc., New York, p.7, 2003.

[9] J.C. Borie, W. Puech and M. Dumas, “Encrypted Medical Images for Secure Transfer”, *International Conference on Diagnostic Imaging and Analysis ICDIA*, Shanghai, August 2002, pp.250-255, 2002.

[10] A. Himani and S. Marisha, “Implementation and Analysis of Various Symmetric Cryptosystems”, *Indian Journal of Science and Technology,* vol. 13, pp.1173-1176, 2012. Availableble online at http://www.indjst.org

[11] C. Jigar, D. Neekhil and K. Bhagyashri, “Enhancing Data Security by Using Hybrid Cryptography Algorithm”, *International Journal of Engineering science and Innovative Technology,* vol. 2, issue 3, pp. 221-228, 2013.

[12] R.P. Jignesh, S.B. Rajesh and K. Vikas, “Hybrid Security Algorithm for Data Transmission using AES-DES”, *International Journal of Applied Information Systems,* vol. 2, pp.15-21, 2012.

[13] K. Shika and K. Ishank, “Data Security Using RSA Algorithm in Matlab”, *International Journal of Innovative Research and Development,* vol. 2, pp. 479-481, 2013.