Data Transmission using AES-RSA Based Hybrid Security Algorithms

Shashikant Kuswaha
Student, B.E IT- RGIT
Andheri, Mumbai
India

Praful B.Choudhary
Student, B.E IT-,RGIT
Andheri, Mumbai
India

Sachin Waghmare
Student, B.E IT- RGIT
Andheri, Mumbai
India

Nilesh Patil
Asst.Prof, RGIT
Andheri, Mumbai
India

Abstract--In a world that relies increasingly on electronic information, data security is more important than ever. Many of the functions of our business and personal life now rely on computers, mobile devices, and the Internet—and there's a lot of data out there to protect. . As large amount of data is transmitted over the network, it is preliminary to secure all types of data before sending them. This is achieved through security controls. Protecting the information transmitted over the network is a difficult task and the data security issues become increasingly important. At present, various types of cryptographic algorithms provide high security to information on networks, but they also have some drawbacks. To improve the strength of these algorithms, we propose a new hybrid cryptographic algorithm in this paper. The algorithm is designed using combination of two cryptographic algorithms AES and RSA. This new hybrid cryptographic technique has been designed for better security along with integrity.

Keywords-*Cryptography, Cipher; RSA; AES; Hybrid AES- RSA*

*****

1. INTRODUCTION

The RSA (Rivest, Shamir and Adleman) is a cryptographic standard. The algorithm is designed to encipher and decipher blocks of data consisting of 1028 bits under control of a 1028-bit key.

In cryptography, the Advanced Encryption Standard (AES) cipher has a 128-bit block size, with key sizes of 128, 192 and 256 bits, respectively.

The integration of AES with RSA is to enhance security for input mode as text and image. The paper outlines the possible weaknesses within the current AES encryption algorithm especially against algebraic based cryptanalysis. To understand the need for minimizing algebraic attacks on AES thereby, the idea of integrating AES with RSA is proposed. Hence the development of the hybrid AES-RSA algorithm is designed [1] [9].

1.1 CRYPTANALYSIS OF AES

AES is based on a design principle known as a Substitution permutation network. It is fast in both software and hardware. Unlike its predecessor DES, AES does not use a Feistel network [2] [5].

AES has a fixed block size of 128 bits and a key size of 128,192, or 256 bits, whereas Rijndael can be specified with block and key sizes in any multiple of 32 bits, with a minimum of 128 bits. The block size has a maximum of 256 bits [4] [7].

AES operates on a 4×4 column-major order matrix of bytes, termed the *state* (versions of Rijndael with a larger block size have additional columns in the state). Most AES calculations are done in a special finite field. The AES cipher is specified as a number of repetitions of transformation rounds that convert the input plaintext into the final output of cipher text. Each round consists of several processing steps, including one that depends on the encryption key. A set of reverse rounds are applied to transform cipher text back into the original plaintext using the same encryption key [2] [6].

The key size used for an AES cipher specifies the number of repetitions of transformation rounds that convert the input plain text into the final output, called the cipher text. The numbers of cycles of repetition are as follows:

- 10 cycles of repetition for 128-bit keys.
- 12 cycles of repetition for 192-bit keys.
- 14 cycles of repetition for 256-bit keys [6].

1.2 CRYPTANALYSIS OF RSA

The RSA scheme is a block cipher in which the plain text and cipher texts are integers between 0 and n-1 for some n. RSA implements a public-key cryptosystem, as well as digital signatures. We examine RSA in this section in some detail, beginning with an explanation of the algorithm.

1.2.1 DESCRIPTION OF THE ALGORITHM

The plain text is encrypted in blocks, with each block having a binary value less than some number n. The public key in this cryptosystem consists of the value *n*, which is called the *modulus*, and the value *e*, which is called the *public exponent*. The private key consists of the modulus *n* and the value *d*, which is called the *private exponent* [3] [10] [11] [12].

An RSA public-key / private-key pair can be generated by the following steps:

1. Generate a pair of large, random prime's *p* and *q*.
2. Compute the modulus *n* as $n = p*q$.
3. Select an odd public exponent *e* between 3 and *n*-1 that is relatively prime to *p*-1 and *q*-1.

4. Compute the private exponent *d* from *e*, *p* and *q*. (See below.)
5. Output (*n*, *e*) as the public key and (*n*, *d*) as the private key.

The encryption operation in the RSA cryptosystem is exponentiation to the *e*th power modulo *n*:

$$c = \text{ENCRYPT}\ (m) = m^e \bmod n.$$

The input *m* is the *message*; the output *c* is the resulting *cipher text*. In practice, the message *m* is typically some kind of appropriately formatted key to be shared. The actual message is encrypted with the shared key using a traditional encryption algorithm. This construction makes it possible to encrypt a message of any length with only one exponentiation.

The decryption operation is exponentiation to the d^{th} power modulo *n*:

$$m = \text{DECRYPT}\ (c) = c^d \bmod n.$$

The relationship between the exponent's *e* and *d* ensures that encryption and decryption are inverses, so that the decryption operation recovers the original message *m*. Without the private key (*n*, *d*) (or equivalently the prime factors *p* and *q*), it's difficult to recover *m* from *c*. Consequently, *n* and *e* can be made public without compromising security, which is the basic requirement for a public-key cryptosystem.
The fact that the encryption and decryption operations are inverses and operate on the same set of inputs also means that the operations can be employed in reverse order to obtain a digital signature scheme following Diffie and Hellman's model. A message can be digitally signed by applying the decryption operation to it, i.e., by exponentiation it to the d^{th} power:

$$s = \text{SIGN}\ (m) = m^d \bmod n.$$

The digital signature can then be verified by applying the encryption operation to it and comparing the result with and/or recovering the message:

$$m = \text{VERIFY}\ (s) = s^e \bmod n.$$

In practice, the plaintext *m* is generally some function of the message, for instance a formatted one-way hash of the message. This makes it possible to sign a message of any length with only one exponentiation.
Figure 1 gives a small example showing the encryption of values *m* from 0 to 9 as well as decryptions of the resulting cipher texts. The exponentiation is optimized as suggested above. To compute *m*3 mod *n*, one first computes *m*2 mod *n* with one modular squaring, then *m*3 mod *n* with a modular multiplication by *m*. The decryption is done similarly: One first computes *c*2 mod *n*, then *c*3 mod *n*, *c*6 mod *n*, and *c*7 mod *n* by alternating modular squaring and modular multiplication [12].

1.3 Advanced Encryption Standard (AES)

A byte in Rijndael is the basic data unit for all cipher operations. Such bytes are interpreted as finite field elements using polynomial representation, where a byte with bits b0, b1... b7 represents the finite field elements. Finite field operations like addition and multiplication are required for key scheduling and rounding. The addition of two finite field elements is achieved by adding the coefficients for corresponding powers in their polynomial representations, this addition being performed in GF(2), that is, modulo 2, so that 1+ 1 = 0.
Addition is nothing but performing XOR between two expressions. Finite field multiplication is more difficult than addition and is achieved by multiplying the polynomials field generator and the remainder is taken as the result.
Since there are 256 possible polynomials, a look up table can be created for a specific field generator. So the lookup table contains 256 * 256 entries [8].

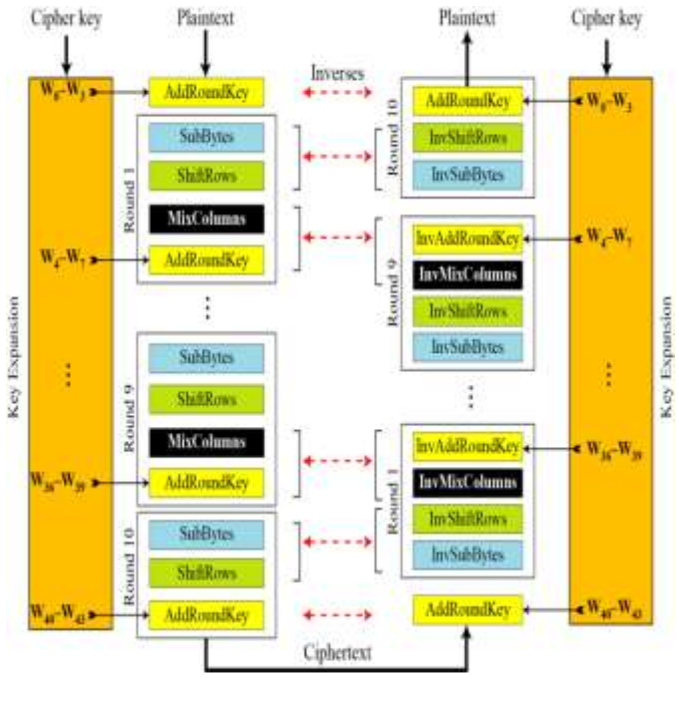


Figure 1. AES Rounds

A. Scheduling

The round keys are each round requiring NC words of key data. For 128-bits keys the key scheduling operates intrinsically on blocks of four (4) 32-bits words. calculating one new round key from the previous one, denote the ith word of the actual round key with K[i], where 0¡i¡ 4, and the ith word of the next round key with K[i]. K [0] is computed by an XOR between K[0], a constant r (field generator) and K[3], the latter being pre rotated and transformed. The other three words K[1], K[2] and K[3] are calculated as K[i] = K[i]*K[i - 1] [5].

B. Rounding

At the start of the cipher, the cipher input is copied into the internal state using the conventions described before figure 1. An initial round key is then added and the state is then transformed by iterating a round function in a number of cycles. It is the algorithm as Rijndael suggested [6].

C. Round Function

One round is termed as a cycle and each cycle has four steps. Each step of transformation is described below. Add Round Key: This is the first step of transformation. The XOR Round Key function declared in Illustration 1 has to perform a bitwise XOR of the state matrix and the round key matrix [2].

D. Sub-Bytes

Transformation: Inverse of the state matrix is found here and a fine transformation is performed [6].

E. Shift Rows

The Shift Rows transformation operates individually on each of the last three rows of the state matrix by cyclically shifting the bytes in the row. The second row is shift done time to the left, third row shifted two times and fourth rows shifted three times [2].

2. HYBRID AES-RSA

In proposed algorithm (Hybrid AES-RSA) the goal had been achieved by combining two algorithms called RSA and AES.

A. For Encryption Of Data

1. The input is consider as Text and image (.jpeg), is being converted to 128 bit plain text.
2. In this hybrid encryption approach, sender side using 128-bit session key value with AES to encrypt the message is used.
3 The hash value of message was encrypted using RSA algorithm with 2048 bit public key of the receiver.
4. Such two sets of encryption AES and RSA to texts and images are then transmitted for further decryption [3].

B. For Decryption Of Data

1. The 2048 bit encrypted data is applied to RSA algorithm, which provides decrypted set of data.
2. This one set of data is then further divided into AES data set.
3. These data sets are then further applied to AES algorithm to get the decrypted set of output.
4. In this sets of 2048 bit decrypted data it gives single original size of output data.

Figure 2 and figure 3 demonstrates the block diagram and the architecture diagram of the proposed hybrid algorithm based encryption and decryption.

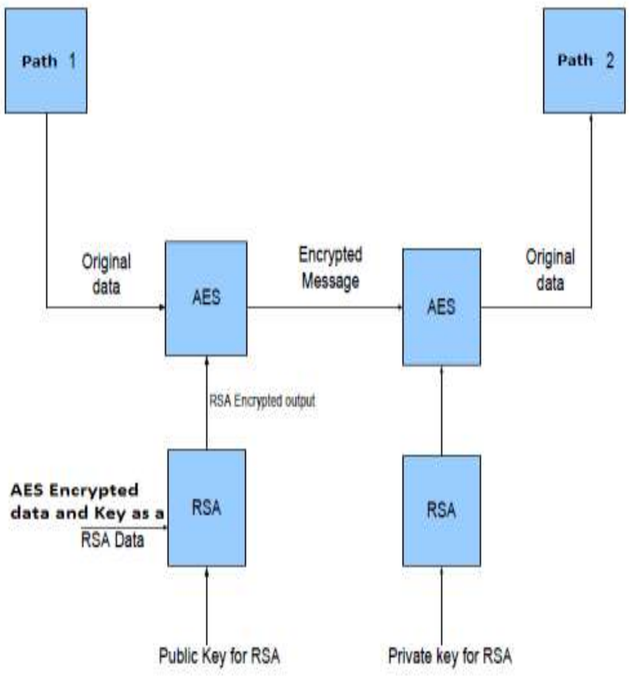


Figure 2. Proposed Hybrid Cryptosystem

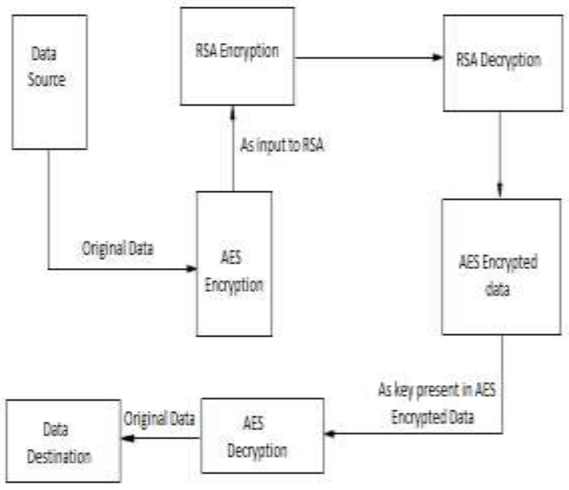


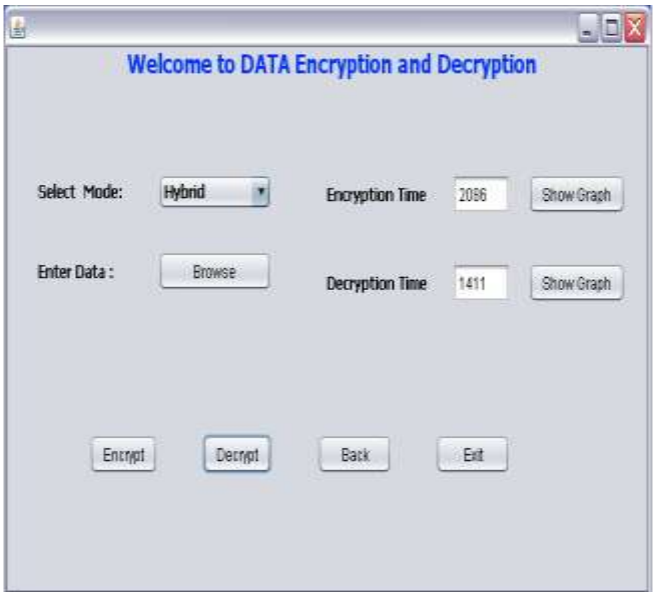
Figure 3. System Architecture

3. IMPLEMENTATION

In this system the implementation of RSA, AES, and Hybrid AES-RSA is done for comparative study; so it can be easily understood that time requirement for encryption of Hybrid

AES-RSA is greater than that of the time requirement of individual AES and RSA.
The time which we get during the experimental result may vary for same input because it depends upon the processor and memory available during the execution of the program. The decryption time is less because the decryption is done continuously with the encryption.

3.1 Hybrid AES-RSA algorithm implemented in this snapshot for input mode (Text)

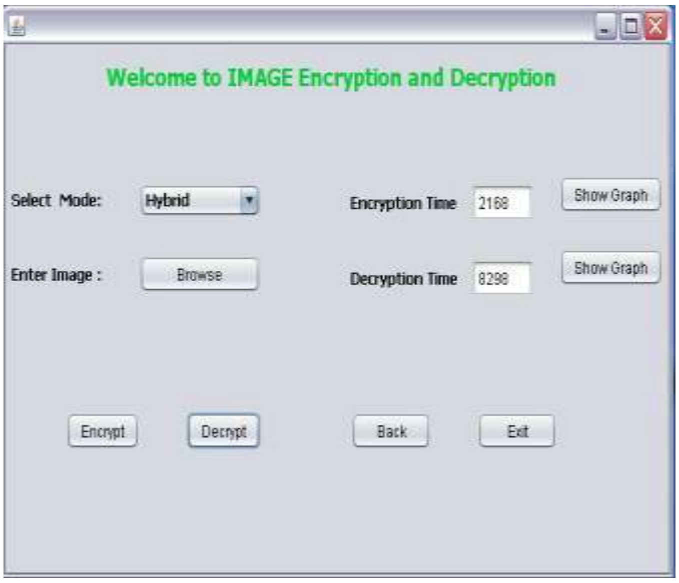

Welcome to DATA Encryption and Decryption

Select Mode: Hybrid
Enter Data :
Encryption Time: 2086
Decryption Time: 1411

Figure 4. UI for Text encryption and decryption

Figure 4 is the UI in which the text files of different sizes are taken as the input to the system for encryption and the same are decrypted.

3.2 Hybrid AES-RSA algorithm implemented in this snapshot for input mode (Image)


Welcome to IMAGE Encryption and Decryption

Select Mode: Hybrid
Enter Image :
Encryption Time: 2168
Decryption Time: 8298

Figure 5. UI for Image encryption and decryption

Figure 5 is the UI in which the jpeg images of different sizes are taken as the input to the system for encryption (such as 328 bits, 798 bits, 3600 bits, 4692 bits, 5700 bits, 2500 bits images) and the same are decrypted.

4. RESULT & PERFORMANCE ANALYSIS

Table 1. Encryption time (average) in milliseconds for Text (mode of input) for different number of bits

Size(Bytes)	145	1410	3230	6460
AES	10	11	11	12
RSA	661	1958	1663	2052
Hybrid	1640	1745	2086	2985

Table 1 and figure 6 depicts the encryption time of text files of different sizes using AES, RSA and Hybrid algorithms respectively.

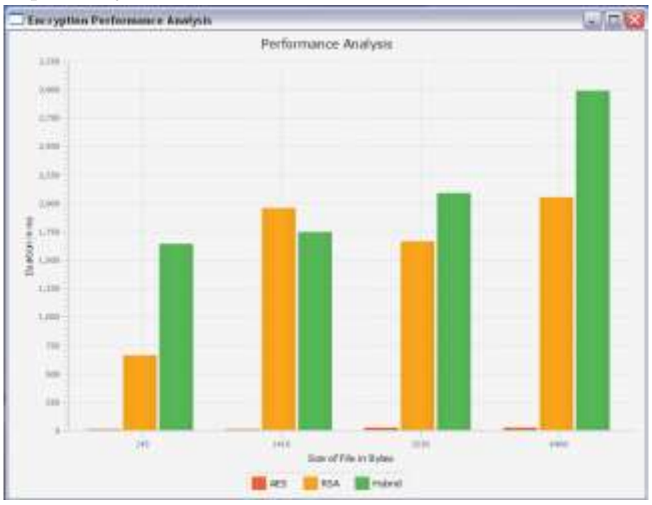


Figure 6. Encryption time for Text mode of input for different number of bits using RSA, AES and Hybrid AES-RSA algorithm.

Table 2. Decryption time (average) for Text (mode of input) for different numbers of bits

Size(Bytes)	145	1410	3230	6460
AES	09	11	11	12
RSA	180	630	1323	2465
Hybrid	283	715	1411	2558

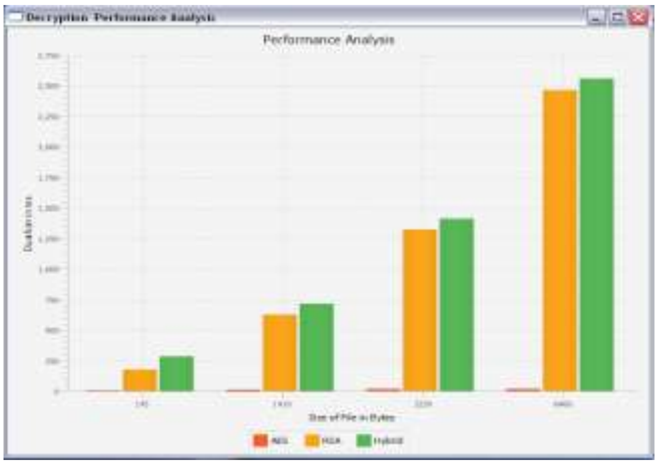


Figure 7. Decryption time for Text mode of input for different number of bits for RSA, AES, and Hybrid AES-RSA algorithm.

Table 2 and figure 7 depicts the decryption time of text files of different sizes using AES, RSA and Hybrid algorithms respectively.

Table 3. Encryption time (average) for Image (mode of input) for different sizes (in KB)

Size(Bytes)	4320	12600	20800	46500
AES	40	60	50	50
RSA	2667	2713	3078	2898
Hybrid	3227	2938	2496	4529

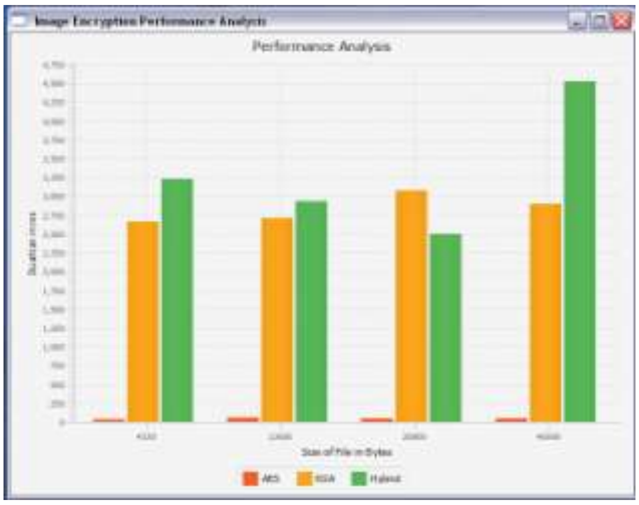


Figure 8 Encryption time for image mode of input for different size of jpeg file using RSA, AES and Hybrid AES-RSA algorithm.

Table 3 and figure 8 depicts the encryption time of text files of different sizes using AES, RSA and Hybrid algorithms respectively.

Table 4. Decryption time (average) for Image (mode of input) for different sizes (in KB)

Size(Bytes)	4320	12600	20800	46500
AES	50	30	50	40
RSA	1848	4941	8139	17957
Hybrid	1871	5053	8219	18046

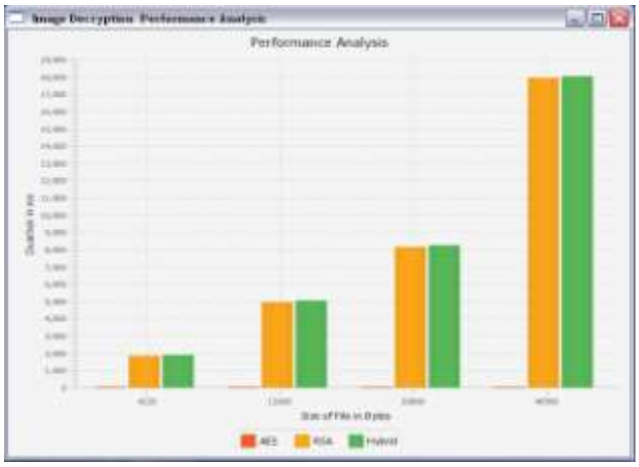


Figure 9 Decryption time for Image mode of input for different size of jpeg file using RSA, AES and Hybrid AES-RSA algorithm.

Table 4 and figure 9 depicts the decryption time of image files of different sizes using AES, RSA and Hybrid algorithms respectively.

5. CONCLUSION

Improved hybrid AES-RSA algorithms as means of strengthening the current AES architecture are studied. The hybrid model gives a better non linearity to the plain AES and as it is merged with RSA there is better diffusion; hence the possibility of an algebraic attack on the hybrid model is reduced. Also the times shown on the analysis are average time because the time may vary depending upon the processor availability and processor speed. In this algorithm the jpeg image and text file used. In this system applying different algorithm like RSA, AES & Hybrid AES-RSA for text and image, difference in time and variation in graph are seen.

In this algorithm, if key is not in valid format then error report is generated. The key format is 16 character key with 8 characters 4 numbers and 4 special characters are there for validation.

Being a hybrid of two powerful encryption standards the algorithm act as efficient and reliable encryption technique for data. The proposed algorithm can also use a double key approach which makes it resistant to linear attacks. In this case, the safety of encryption algorithm can be further

improved. Improving RSA by adding the Hashing power like SHAl and MD5 algorithm and integrating the AES algorithm into RSA structure with Hashing further improves its security.
RSA consume longest encryption time and memory usage is also very high but output byte is least in case of RSA algorithm.

6. ACKNOWLEDGEMENT

With profound sincerity and gratitude, we acknowledge our constant discussion with our professor **Mr**. **Nilesh Patil** for his valuable guidance and support extended during the work. We would also like to thank our HOD **Mr. Dilip Dalgade** for carrying out this work as a part of our B.E. project.

7. REFERENCES

[1] Behrouz A. Forouzan "*Cryptography and network security"* TATA McGraw Hill publication, 2007 edition.
[2]]Aida Janadi, 2008 "AES immunity Enhancement against algebraic attacks by using dynamic S-Boxes", Information and Communication Technologies from Theory to Applications, ICTTA2008.
[3] Avi Kak, Avinash Kak, "*Computer and Network Security on Public-Key Cryptography and RSA*" May 15, 2013 Purdue University
[4] Yuan Kun, Zhang Hanli Zhaohui, 2009 "An Improved AES algorithm based on chaos", Multimedia Information Networking and Security, INES'09, International Conference.
[5] M.Zeghid, M.Machhout, L. khriji, A.baganne and R.Tourki, 2007 "A modified AES based algorithm for image encryption", World Academy of Science, Engineering and Technology.
[6] AES mix column, http://www.angelfire.com/biz7/atleast/mix_columns.pdf
[7] AES Description, Available: http://people.eku.edu/styere/Encrypt/JS-AES.html
[8] Vikas Kaul, S K Narayankhedkar, S Achrekar, S Agrawal, P Goyal, "Security Enhancement Algorithms for Data Transmission for Next Generation Networks", International Journal of Computer Application (IJCA).
[9] William Stallings:" Cryptography and network Security: Principles and Practices".
[10] http://en.wikipedia.org/wiki/RSA.
[11] Yu; Tong Li; Na Zhao; Fei Dai "*Design and implementation of an improved RSA algorithm*", April 2010.
[12] Tanzila Afrin, K. J. Satao, "E-Voting System for on Duty Person Using RSA Algorithm with Kerberos Concept", IJARCET, Vol.2, Issue 7, July 2013.