Volume 5, Issue 4, 2015 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Advance Hybrid Cryptography Algorithm for Improving Document Security and Its Specific Implementation

Sankalp Agarwal*
M.Tech. C.S., Jamia Hamdard,
New Delhi, India

Jawed Ahmed
Asst. Professor, Jamia Hamdard,
New Delhi, India

Abstract: The purpose of document security is to protect the important documents and information of an organization. With the selection and application of security organization can achieve its business objectives by protecting the physical and electronic documents and other assets. Document security deals with the security of document for people within and outside the organization. Unauthorized persons try to gain access of the document while the person inside the organization tries to protect the document. They try to protect the weakest link of the security. The measure of security along with the polices for protection can improve the security for end users and results in a more secured document management systems.
My area of working at my organization also helped me to gain knowledge and experience in this related field. In this proposed algorithm I have used logical operation like XOR and shifting operations. Experimental results show that the algorithm is secure and efficient. I have divided this paper in various parts. Section I deals with the introduction to the hybrid cryptosystems. Section II describes the proposed algorithm. Section III deals with the proposed work done for this algorithm, Section IV deals with the implementation in Python while section V deals with the result comparisons.

Keywords: Document Security, Encryption, Decryption, Cryptography.

I. INTRODUCTION

In Cryptography, Asymmetric techniques are very convenient to use as they do not require the sender and receiver to share the encryption key used. However they are inefficient as compared to symmetric methods in terms of execution time and cost, as complex mathematical computations are involved. It may be very costly in terms of encrypting and decrypting a large document. A hybrid cryptosystem is developed for gaining the combined convenience of asymmetric and symmetric key algorithms. As a lot of research work has been continuously going on and new algorithms are proposed as the extension of some of the existing algorithms. Normally a hybrid algorithm is used as

- a key encapsulation scheme which is asymmetric and
- a data encapsulation scheme which is symmetric.

II. PROPOSED ALGORITHM

Here is the new proposed algorithm named “Advance Hybrid Cryptography Algorithm for improving document security” is discussed. This algorithm uses both symmetric and asymmetric methods. A key generator method is also developed for generating a random number. In this method a substitution method is used in which 16 characters of input is taken from a document and been encrypted from a substituted random number of same number of characters. The result is stored sequence wise in a new encrypted output document. The key is now encrypted with RSA algorithm using the public key of the recipient. The key contains all possible words comprising of characters each generated from all characters whose ASCII code is from 0 to 255 in a random order.
To decrypt the document one needs to exactly know the private key of the recipient which is not easy to access. Only after that the key is generated and can be used to decrypt the text only if the algorithm is also known. I have applied this algorithm on various type of files like word, PDF, text file etc. and it is encrypting and decrypting the document accurately, efficiently and have found in all cases it giving 100% correct solution.

III. PROPOSED WORK

In this section I am presenting the new block based symmetric cryptography and way to protect the key used to encrypt and decrypt the block. In this technique I am using a random number of 64 characters to be used for key. The text block is encrypted multiple times. The Pattern of the key blocks will depend on the text input from the document. To Decrypt the document one needs exactly to know the private key of the recipients. Theoretical it will require to have atleast 2^{128} trail run which is not easily possible.
Method of encryption:
Here I am using symmetric encryption approach. Here I am choosing the block cipher type because its efficiency and security. In this proposed technique a common random key is used between the sender and receiver, which is a private key. This private key is then further encrypted using the asymmetric encryption approach and hence making the full method as a hybrid approach.

Reason for choosing the hybrid approach

1. Encryption technique for plain text is easy and simple
2. Each encrypting partner is using the same algorithm and no need to exchange algorithms
3. Security depends on the security of the private key
4. Keys are relatively small as compared to asymmetric keys for data encryption
5. Encryption and decryptions takes short execution time as compared to only using asymmetric algorithms

A- Proposed Method for generating key

1. Select or create a random number containing ASCII characters from 0 to 255 of 64 characters or 512 bytes and store the result in KeyBlock
2. Divide the block of 64 bytes into 4 blocks of 16 bytes each and call them KeyBlockA, KeyBlockB, KeyBlockC, and KeyBlockD.
3. Apply XOR operation between KeyBlockA and KeyBlockC and store the result in new KeyBlockAC.
4. Apply XOR operation between KeyBlockB and KeyBlockAC and store the result in new KeyBlockBAC
5. Apply XOR operation between KeyBlockD and KeyBlockBAC and store the result in new KeyBlockDBAC

B- Proposed method of encrypting text from a document.

1. Initially select 16 bytes from a document(pdf or text file) and store it in TEXTBLOCK
2. Apply XOR operation between KeyBlockDBAC and TEXTBLOCK (plain text block) and store the result in new CipherBlockA
3. Apply right circular shift with value 3 and store the result in new CipherBlockB
4. Apply XOR operation between CipherBlockB and KeyBlockB and store the result in CipherBlockC
5. Apply XOR operation between CipherBlockC and KeyBlockD and store the result in CipherBlockD
6. Store CipherBlockD sequentially in new document to be transmitted.

Proposed Method of transmitting the encryption key

1. Capture the public key of the receiver of the file either from the certificate available or from the public key certificate repository.
2. Encrypt the key KeyBlock the using this public key using the RSA algorithm and store the result in KeyBlockENC.
3. Send the key to the recipient along with the encrypted document

C- Proposed Method for decrypting the key

1. Authorized Recipient will use his private key and apply RSA algorithm on KeyBlockENC to generate the plain text key that will be equivalent to KeyBlock
2. Apply the same algorithm to generate key as mentioned in section III- A above.

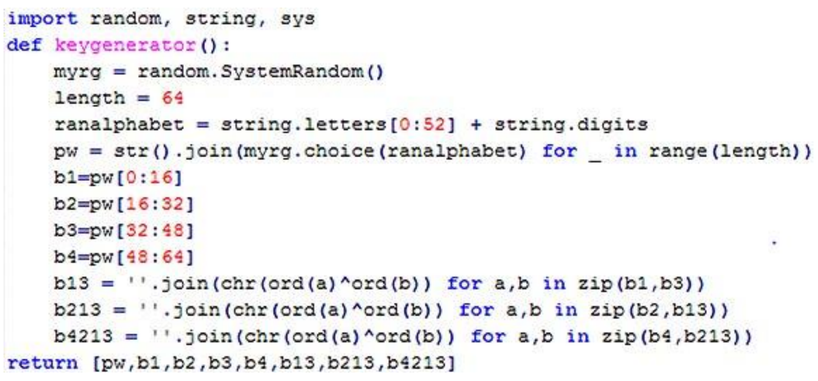
D- Proposed Method for decrypting the Cipher text.

1. Initially select 16 bytes from the encrypted document(pdf or text file) and store it in CIPHERBLOCK
2. Apply XOR operation between CIPHERBLOCK and KeyBlockD and store the result in CipherBlockA
3. Apply XOR operation between CipherBlockA and KeyBlockB and store the result in CipherBlockB
4. Apply left circular shift with value 3 and store the result in new CipherBlockC
5. Apply XOR operation between KeyBlockDBAC and CipherBlockC and store the result in new TEXTBLOCK(plain text block)
6. Store TEXTBLOCK sequentially in new document to get the original document.

IV. SPECIFIC IMPLEMENTATION

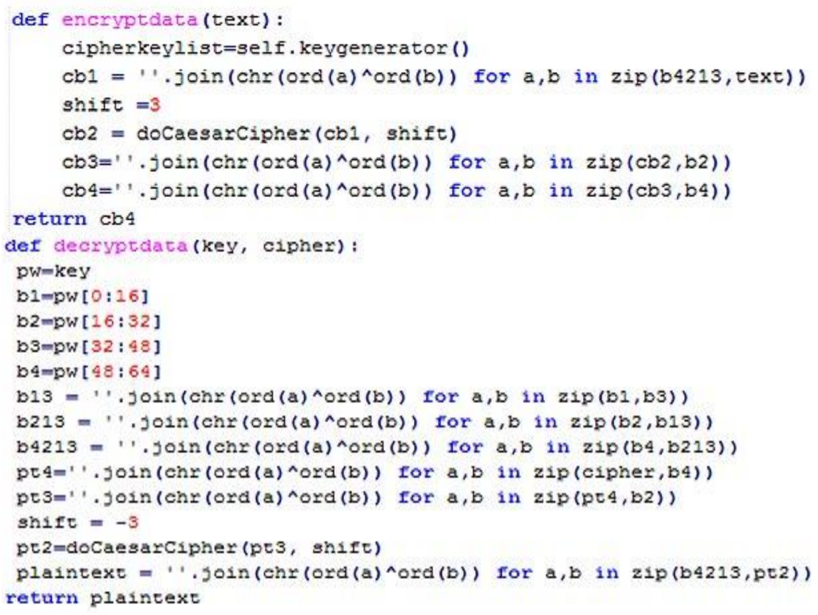
In this section we will show a specific implementation of the proposed algorithm in python.

Key generation



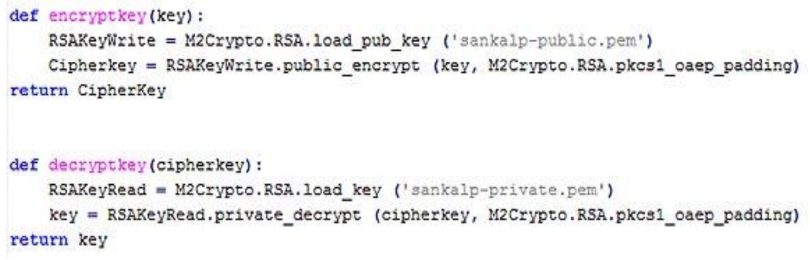
```
import random, string, sys
def keygenerator():
    myrg = random.SystemRandom()
    length = 64
    ranalphabet = string.letters[0:52] + string.digits
    pw = str().join(myrg.choice(ranalphabet) for _ in range(length))
    b1=pw[0:16]
    b2=pw[16:32]
    b3=pw[32:48]
    b4=pw[48:64]
    b13 = ''.join(chr(ord(a)^ord(b)) for a,b in zip(b1,b3))
    b213 = ''.join(chr(ord(a)^ord(b)) for a,b in zip(b2,b13))
    b4213 = ''.join(chr(ord(a)^ord(b)) for a,b in zip(b4,b213))
return [pw,b1,b2,b3,b4,b13,b213,b4213]
```

Encryption and Decryption



```
def encryptdata(text):
    cipherkeylist=self.keygenerator()
    cb1 = ''.join(chr(ord(a)^ord(b)) for a,b in zip(b4213,text))
    shift =3
    cb2 = doCaesarCipher(cb1, shift)
    cb3=''.join(chr(ord(a)^ord(b)) for a,b in zip(cb2,b2))
    cb4=''.join(chr(ord(a)^ord(b)) for a,b in zip(cb3,b4))
return cb4
def decryptdata(key, cipher):
 pw=key
 b1=pw[0:16]
 b2=pw[16:32]
 b3=pw[32:48]
 b4=pw[48:64]
 b13 = ''.join(chr(ord(a)^ord(b)) for a,b in zip(b1,b3))
 b213 = ''.join(chr(ord(a)^ord(b)) for a,b in zip(b2,b13))
 b4213 = ''.join(chr(ord(a)^ord(b)) for a,b in zip(b4,b213))
 pt4=''.join(chr(ord(a)^ord(b)) for a,b in zip(cipher,b4))
 pt3=''.join(chr(ord(a)^ord(b)) for a,b in zip(pt4,b2))
 shift = -3
 pt2=doCaesarCipher(pt3, shift)
 plaintext = ''.join(chr(ord(a)^ord(b)) for a,b in zip(b4213,pt2))
return plaintext
```

Encryption and Decryption of Key



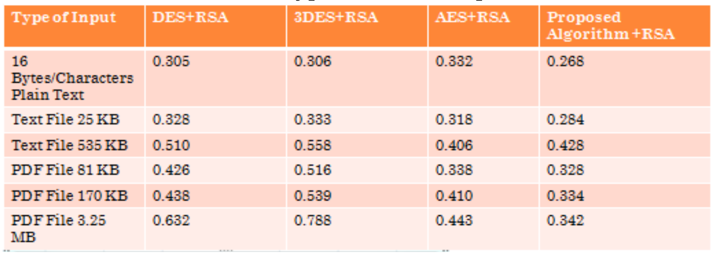
```
def encryptkey(key):
    RSAKeyWrite = M2Crypto.RSA.load_pub_key ('sankalp-public.pem')
    Cipherkey = RSAKeyWrite.public_encrypt (key, M2Crypto.RSA.pkcs1_oaep_padding)
return CipherKey


def decryptkey(cipherkey):
    RSAKeyRead = M2Crypto.RSA.load_key ('sankalp-private.pem')
    key = RSAKeyRead.private_decrypt (cipherkey, M2Crypto.RSA.pkcs1_oaep_padding)
return key
```

V. RESULT COMPARISONS

For result comparision I am using two parameters, execution time and the document size. The encryption execution time is shown in table 1 and decryption execution time is shown in table 2. Here I am doing comparision with various other hybrid methods including DES and AES along with RSA. Finally, the outputs execution time is measured in numeric form. An encryption algorithm execution time not only depends on the algorithm's complexity, but also the key and the plaintext have certain impact.

Table 1 Encryption time Comparison



Type of Input	DES+RSA	3DES+RSA	AES+RSA	Proposed Algorithm +RSA
16 Bytes/Characters Plain Text	0.305	0.306	0.332	0.268
Text File 25 KB	0.328	0.333	0.318	0.284
Text File 535 KB	0.510	0.558	0.406	0.428
PDF File 81 KB	0.426	0.516	0.338	0.328
PDF File 170 KB	0.438	0.539	0.410	0.334
PDF File 3.25 MB	0.632	0.788	0.443	0.342

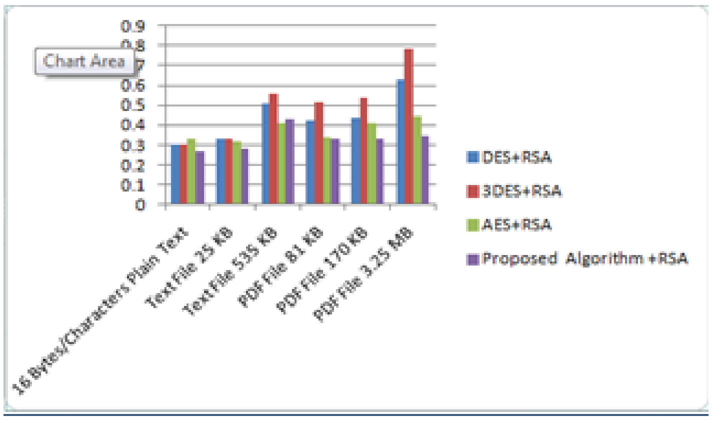
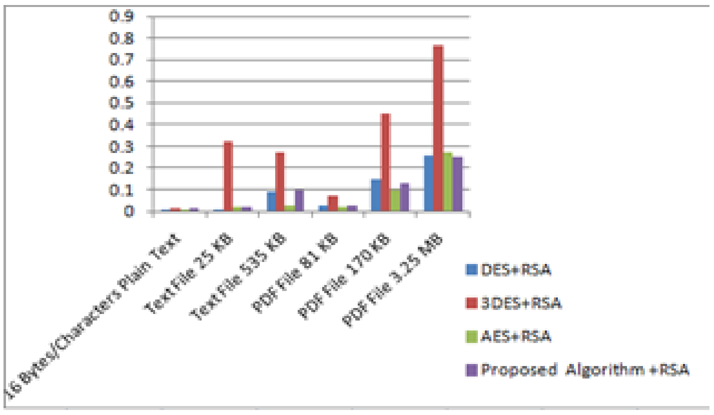


Table 2 Decryption Time Comparison

Type of Input	DES+RSA	3DES+RSA	AES+RSA	Proposed Algorithm +RSA
16 Bytes/Characters Plain Text	0.007	0.014	0.010	0.012
Text File 25 KB	0.010	0.321	0.020	0.018
Text File 535 KB	0.091	0.268	0.026	0.099
PDF File 81 KB	0.028	0.071	0.018	0.024
PDF File 170 KB	0.150	0.452	0.099	0.128
PDF File 3.25 MB	0.260	0.768	0.268	0.252



VI. CONCLUSION AND FUTURE ENHANCEMENT

From the results it is shown that my proposed technique is having better execution time as compared to Hybrid techniques with Data encryption Standard (DES) and Advance encryption standards(AES). If security is concerned then proposed algorithm can be used for transmitting the encrypted documents. This method is essentially block cipher method and it will take less time for a large file. The most important aspect is that it is nearly impossible to break the encryption without knowing the private key. Further the identity of the sender can also be identified using the RSA algorithm which may be a future enhancement.

REFERENCES

[1] Aamer Nadeem, Dr M. Younus Javed, A Performance Comparison of Data Encryption Algorithms", Information and Communication Technologies, 2005..

[2] Yan Wang, Ming Hu , "Timing evaluation of the known cryptographic algorithms", International Conference on Computational Intelligence and Security, 2009.

[3] Akash kumar Mandal, Chandra Prakash , "Performance Evaluation of Cryptographic Algorithms: DES and AES", IEEE Students' Conference on Electrical, Electronics and Computer Science , 2012

[4] Akash Kumar Mandal, Mrs. Archana Tiwari , "Comparative Study of Avalanche Effect in DES Using Binary Code" , National Conference on Computing and Communication Systems (NCCCS), 2012

[5] Vishwa Gupta, Gajendra Singh, Ravindra Gupta, "Advance cryptography algorithm for improving data security", International Journal of Advanced Research in Computer Science and Software Engineering ,2012.