International Journal of Advanced Science and Technology
Vol.71 (2014), pp.15-30
http://dx.doi.org/10.14257/ijast.2014.71.02

A Python Based Enhanced Secret Sharing Scheme to Secure Information using Cryptography Techniques

Siyaram Gupta[1] and Madhu Sharma[2]

[1]Department of Information Technology, Dehradun Institute of Technology, Dehradun, India
[2]Department of Computer Science, Dehradun Institute of Technology, Dehradun, India
[1]siyaram421@gmail.com, [2]madhuashishsharma@gmail.com

Abstract

There are lots of algorithms and concepts that are employed to secure intellectual property of the user. In this paper a scheme known as secret sharing is used to securely distribute user's work among different parties. Objective behind the paper is to provide such a protocol that enhance and achieve privacy and other expectations of the owner of the information. The scheme states that, the information is accessible only when at least k shareholders from n must get together. User's shares are constructing through quadratic polynomial. To make such a scheme more relevant to its users, the concept of cryptography is applied. Here the shares are encrypted so that dishonest dealer and malicious participants are not able to view or alter the data. The shares are enciphered by using different encryption algorithms to make a comparative study and to know most secure and powerful among them. "PyCrypto" a python based cryptography tool is used for this purpose.

Keywords: *Secret sharing, encryption, polynomial, hashing, python, share*

1. Introduction

1.1. Brief Introduction to Shamir's sharing Scheme

In this modern digital scenario, security and privacy of the information and documents is a major concern. The information stored on media or at disk are always the subject of risk. The main challenge of the user is how to transfer such confidential information to other end. As we know the attackers or intruders are equipped with lots of tools to scan and sniff network and snatch the information. To overcome such types of security issues Adi Shamir in 1979 [1] introduce a concept in which the secret is partitioned into n piece of shares and distribute those to each one. The key concept behind the scheme is that If at least 'k' shareholders with their respective unique id meet together, only then the secret is accessible. The dealer breaks or divide the secret into shares and provide the same to shareholders, so that they can recover the secret when the condition of threshold is fulfilled. Lagrange's interpolation polynomial is used to construct share for participants.

The scheme is based on polynomial' interpolation:

t points in the two-dimensional plane (X_1, Y_1) ……

(X_t, Y_t). with unique X_i's , there is one and only one polynomial P(X) of degree t -1 such that P(X) =Y_i for all i.

We can assume that the secret Sc is a number. To divide it into pieces Sc_i, we choose a random t-1 degree polynomial $P(x) = (Sc+a_1 x+a_2 x^2+\cdots+a_{t-1}x^{t-1})$ in which a_0=Sc , and evaluate:

$$Sc_1 = P(1) \,....., \; Sc_i = P(i) \,....., \; Sc_n = P(n).$$

With any subset of t of these Sc_i values together with their distinct identifying indices, we can find the coefficients of P(X) by interpolation, and then evaluate Sc=P(0).

On the other hand knowledge of just t-1 of these values, does not suffice in order to calculate Sc. Use modular arithmetic instead of real arithmetic to make this claim more precise. The integer modulo a prime number M forms a field in which interpolation is possible. Given an integer valued data Sc, we pick a prime M which is bigger than both Sc and n. The coefficients a_1 ..... a_{t-1} in P(X) are randomly chosen over the integers in [0, M), and the values Sc_1 ..... Sc_n are computed modulo M.

So it is easy to construct shares for the participants as shown in equation (1).

$$P(x) = (Sc+a_1 x+a_2 x^2+\cdots+a_{t-1}x^{t-1}) \bmod M \qquad (1)$$

1.2. Introduction to Python

In 1991 Guido van Rossum was the first who introduce the python language. One of the important features of the language is that it is open source and user friendly. Python has huge collection of tools, libraries, functions and module that helps its user to implement his/her concept. A user with basic or no knowledge of python can use the freely available documentation to learn the language. It is an object-oriented, high-level and scripting language. The user can use the tools and functions such as pycrypto, gmpy, octave, numpy etc to implement the project. It is also an easy task for its user to construct polynomials and generate random numbers. Each functions and tools has its own importance, GMPY provide multi-precision values, and Numpy is used for numerical calculation. Pycrypto is a cryptographic tool that is used to encipher the user work. The user of python can use the power of pycrypto to hide, encrypt or hash the documents, files, images *etc*. Python language can extend the features of C/C++ and Java in the form of Cython and Jython respectively. Python is an interpreter language and has efficient data structure.

PyCrypto Toolkit: Python programming language has the Python Cryptography Toolkit that describe a package containing various cryptographic modules. The toolkit intended to provide stable and reliable base for writing cryptographic function for python programs.

PyCrypto Toolkit provides following useful cryptographic functions:

1. Crypto. Hash: Hash Functions: Hash functions can be used as a checksum, used to implement digital signatures and also along with or in association with public-key algorithms. SHA12, SHA256 MD2, and MD5 etc are the most common example of cryptographic hash functions.

The below example show the use of the MD5 algorithm:

```
>>> from Crypto. Hash import MD5
>>> msg = MD5.new ()
>>> msg.update ('python')
>>> msg.digest ()
'\x90\x01P\x98<\xd2O\xb0\xd6\x96?}(\xe1\x7fr'
```

2. Crypto. Cipher: The function of the encryption algorithms is to transform the input information *i.e.*, plaintext which is dependent on the key size or key length and producing cipher-text as an output data. AES, DES, CAST , ARC4 are the currently available block-ciphers in Crypto. Cipher package.

The example shows DES module:

```
>>> from Crypto. Cipher import DES
>>> object=DES.new ('IJASTSEC', DES.MODE_CBC)
>>> p_txt="Guido Rossum introduce python language."
>>> c_txt=object.encrypt (p_txt)
>>> object.decrypt (c_txt)
```

Octave: Octave is a high-level language, used for numerical computations. It is typically used for such problems as solving linear and nonlinear equations, statistical analysis, numerical linear algebra and for performing other numerical experiments. Octave provided a command-line interface with graphical results displayed in separate windows. Octave is also useful for histogram generation.

Here is an example of how to generate histograms:

```
octave:1>   myfile=fopen('CiphAES.txt','rb');
octave:2>   mydata=fread(myfile);
octave:3>   size(mydata)
ans =

   304     1

octave:4>   hist(mydata,(0:255)');
octave:5>   title("A HISTOGRAM CipherText");
```

GMPY Function: GMPY is multiple-precision arithmetic C-coded python module that supports for the MPFR and MPC libraries. Gmpy uses mpf or mpfr type to support multiple -precision reals. The gmpy mpz type supports arbitrary precision integers. Gmpy provides a rational type call mpq.
"gmpy.mpz ( )" for multi-precision integer value.

Example:

```
 import gmpy
G = gmpy.mpz (64)
(A 64 bit gmpy integer variable)
```

Random Function: "random.randrange ( )" and "random.getrandbits ( )" are the functions used to generate random numbers.

Example:

```
R = random.randrange (10, 64)
(Generate a random numbers in between 10 to 64.)
```

Numpy Library: Numeric Python (NumPy) package provides basic routines for Polynomials, Random numbers, Array creation routines, Linear Algebra, Statistics and much more. Numpy library is useful for numeric calculations.

Example:

```
import numpy as npy
Array Creation:        A = npy.array ([[1, 2, 3], [4, 5, 6]],    float)
Polynomials:           npy.polyint ([1, 1, 1, 1])
```

There are lots of applications of python such as scripting language for web applications, Natural language processing tasks and artificial intelligence works are also done by using python. Applications such as medical image security, EPR hiding [13] implements this scheme to keep their records secret.

1.3. Introduction to Different Encryption Algorithms

1.3.1. Advance Encryption Standard(AES): AES is the symmetric-key block cipher standard published by NIST in December 2001[11-12]. AES has 128-bits block size with three different key sizes of 128, 192, 256 bits. The main criteria behind selecting AES are its security, cost and implementation. AES has 3 versions with 10, 12 and 14 rounds and each has different cipher key size but the round keys are always 128-bits.

1.3.2. Data Encryption Standard (DES): DES is a symmetric block cipher standard published by NIST [11-12]. The key size of DES is 64-bit long with 8 byte of fixed data block size. Because its key length is too short and is vulnerable to brute-force attack, DES should not be used for new designs.

1.3.3. Triple DES (3DES): TDES is also a symmetric block cipher standardized by NIST. Its data block size is 64-bits. It has two different size of keys 128 and 192 with effective key length 112 or 168 respectively. 3DES is simply the concatenation of 3 simple DES ciphers. It is important that all the sub-keys of 3DES are different, otherwise it degrade to single DES.

1.3.4. Alleged RC4 (ARC4): Alleged RC4 (ARC4) is an implementation of Rivest's Cipher version 4(RC4), a sym metric stream cipher designed in 1987 by Ron Rivest. ARC4 does not take a nonce or an IV. Keys can vary in length from 40 to 2048 bits.

1.3.5. CAST-128: CAST-128 is a symmetric block cipher specified in RFC2144. The key size of CAST-128 can vary in length from 40 to 128 bits with fixed data block size of 8 byte. It is vulnerable to brute-force attack.

2. Related Work

The concept of secret sharing is first developed by Adi Shamir [1] in 1979 later Blakley [2] try to work on the scheme. The theme of the concept is to secure the information from crackers and malicious users. The scheme is very useful in the environment where a group of conflicting nature individuals must co-operate.

The scheme has lots of drawbacks like there is a limit number of shares can be shared during in a single process, malicious users may distribute fake shares to other group or individual.

A multi-secret sharing (MSS) scheme has been proposed to overcome such issues [9-10]. In 2004 C.C. Yang, T.-Y. Chang, M.S. Hwang [4] proposed a 2-variable one-way function [3] multi –secret-sharing (MSS). To limit the scope of dishonest dealer along with malicious users a verificable –secret-sharing (VSS) scheme has been proposed. A VSS scheme written by Chor *et al.*, [5] in 1985 assures and allow shareholders to validate and verify their shares.

Later Harn [6] improve the VSS and present a new scheme verifiable-multi-secret-sharing scheme in 1995. A YCH scheme is a efficient multi-secret scheme till date, but because of its big-ticket system and it doesn't have the property of verification the scheme is impractical. Later Jianjie Zhao, Jianzhong Zhang, Rong Zhao [14] proposed a new practical verifiable multi-secret sharing scheme which is based on discrete logarithm [8] to overcome above drawback.

3. Proposed Work

The proposed scheme demonstrate the logic how to use cryptographic tools along with secret sharing concept. The work is all about share encryption and their distribution to different participants, the scheme has a rule that if at least k people from n participants must gather then only information is accessible. The user of the algorithm decides to whom and how many shares are constructed and according to scheme the user has also right to decide the threshold value, *i.e.*, the number of shareholders required to reconstruct the secret.

3.1. Secret Sharing with Python

In the proposed scheme k different shares are generated from a secret, so here the threshold value to reconstruct the desired secret is n. Both the value of k and n is user defined. For modulus calculations we chooses a 64-bit largest prime number, a quadratic polynomial with k co-efficient is taken as shares for users or participants. The k shares are generated by evaluating the coefficients (A_s) at randomly chosen B_s. Random function, user defined evaluation function and re-construct functions are used for random share generation, share evaluation and share reconstruction respectively.

Five different encryption algorithms AES, DES, 3DES, ARC4 and CAST-128 are used to encrypt the user id and their shares to make comparative study.

The proposed work mainly consists three sections:

1. Encryption of the shares
2. Secret Sharing
3. Decrypting the shares.

1. Encryption of the shares :- The dealer divides the secret into n pieces and encrypt each to enhance security level and privacy. There are different symmetric key algorithms to chose from such as AES, 3DES , DES etc. The participants are now able to encipher their shares with the help of the algorithms. The user shares are stored in text files.

The algorithm used for share encryption are symmetric key algorithm, means the keys for both encryption and decryption are same. The original polynomial is used as encryption key.

Here an example of ARC4 encryption algorithm for share encryption

```
>>> from Crypto.Cipher import ARC4
>>> Encryption _Key = ‘ Variable key size’
>>> Obj1 = ARC4.new(Encryption _Key)
>>> Ciph_Txt = Obj1.encrypt(Share)
```

2 . Secret Sharing : - The dealer is now going to distribute the encrypted shares among different shareholders The participants are now gives their encrypted shares for sharing process. Their shares are used to recover the secret.

3 . Decrypting the shares :- Once the participants gets their shares. According to proposed work at least n participants out of k must gives their shares to reconstruct the secret. All the k shareholders have unique identification number.

Decryption of the shares using ARC4 algorithm.

```
>>> Decryption _Key = ‘ Variable key size’
>>> Obj2 = ARC4.new(Decryption _Key)
>>> Plain_Txt = Obj1.encrypt(Ciph_Txt)
```

These three phases are most important one in the proposed work.

3.2. Implementation

The proposed scheme has five main Phases :-

1. Share generation stage.
2. Share Encryption stage.
3. Share Distribution and sharing stage.
4. Share Decryption stage.
5. Share Re-Construction Stage.

3.2.1. Share Generation Stage : The user defined function for share evaluation.
This evaluate quadratic modular polynomial defined by Coffi at b, mod is the modulus.

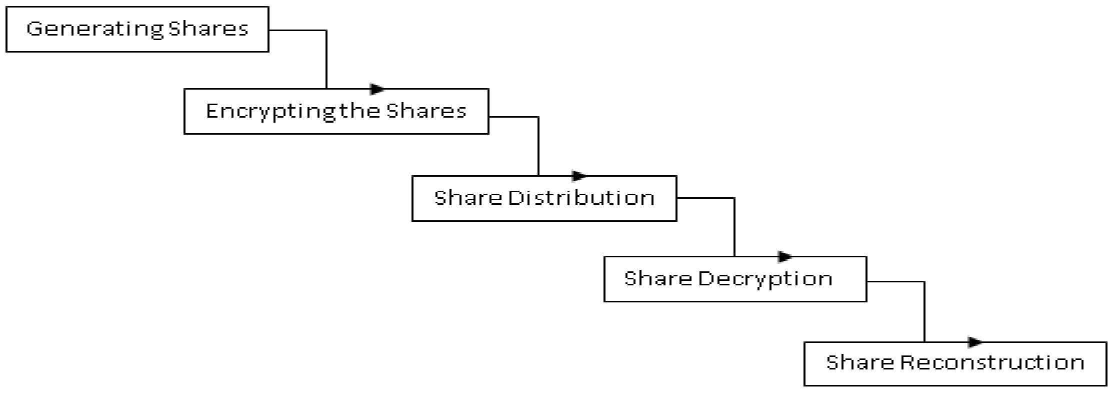

Figure 1. Stages of Secret Sharing

Python’s random function “random.randrange(valu1,value2)” is used to construct shares for participants.

```
Enter the Number of shares you want to generate from a secret (n) :  7
Enter the threshold value (k) : 4
Shares:
[(13026962538231705647L),(548623285850384019),(1096708402253082360),(15
882687035326081992L),(10674522843905371509L),   (16198329628412310189L)
, (11450824535861532127L)]

Unique Ids :
[(15963329170807749408L),(14164439201954913113L),(3852488410425486723)
,(896391978242292738),(37069081006830707926L),(972635168242542998 4),
(3960423652794329163L)]

Eval_Quad_Poly:
[(4421789032168969013 5),(95332490871238863097L),(8942310045998278693L)
, (3998126785001337930 1)]
```

Figure 2. Share Generation

```
>>> Eval_Poly(Coffi , b , mod)            # The evaluated quadratic polynomial

>>> Eval_Quad_Poly = [gmpy.mpz(random.randrange(0, MOD))]
# Where MOD is the largest 64-bit prime number defined as
>>> MOD = gmpy.mpz(2^64 -59)

>>> B_s = [gmpy.mpz (random.randrange (0, MOD))]              # Randomly chosen value
>>> A_s = [Eval_Poly (Eval_Quad_Poly, B_s [0], MOD)]          # Generated Shares A_s
```

3.2.2. Share Encryption Stage: The 2nd stag of the proposed concept is to encryption of the shares constructed in 1st stage. The share is encrypted with dealer’s key, the dealer can generate a x-byte key according to encryption algorithm. The x-byte key is then used to encrypt the share A_s and id of participants $B_{s.}$ PyCrypto a python cryptographic tool is used for the purpose. Here five different encryption algorithm is used to perform comparative study.

```
>>> from Crypto. Cipher import NA (Name of Algorithm)
>>> Object1= NA.new ("Key Value", MODE, 'Initialization Vector')
>>> C_Sahre = Object1.encrypt(A_s)                 # Share A_S is to be encrypted
```

Same for B_s but with different object.

```
C_Share : �  M  �Z�>L  �+�U���j[jv��0�G<z
C_Unique Ids:    n���3ℲTe�w*��  4G�m�'�⍰4G�m�'�
```

Figure 3. Share Encryption

Python’s cryptographic tool “Pycrypto” is being employed for share encryption. The tool consists of all symmetric and asymmetric algorithms and hashing algorithms.

3.2.3. Share Distribution and Sharing Stage: The third stage of the scheme is distribution of the encrypted shares and the ids. All k participants gets their desired shares. The dealer distribute and share the encrypted secret to its users.

3.2.4. Share Decryption Stage: Once the participants gets their shares they can decrypt it to recover the share. Because the scheme use symmetric algorithm so same key which is used for encryption is used to decrypt the share.

```
>>> from Crypto. Cipher import NA (Name of Algorithm)
>>> Object2= NA.new ("Key Value", MODE, 'Initialization Vector')
>>> P_Sahre = Object2.decrypt(C_Share)                # Share A_S is to be decrypted
```

Same for B_s but with different object.

```
P_Share : 67123980433753439170

P_Unique Ids:523981200349571289015
```

Figure 4. Share Decryption

Python’s cryptographic tool “Pycrypto” is being employed for share encryption. The tool consists of all symmetric and asymmetric algorithms and hashing algorithms.

3.2.5. Share Re-construction Stage: In these last stage of the scheme the participants are going to recover the secrets, three participants out of five are gives their share to recover the secret.

```
Eval_Quad_Poly:
[(44217890321689690135),(95332490871238863097L),(894231004599827869
30L), (39981267850013379301)]

Rec_Quad_Poly:
[(44217890321689690135),(95332490871238863097L),(894231004599827869
30L), (39981267850013379301)]
```

Figure 5. Recovered Share

```
# Decrypted Unique Ids
>>> Bs = [Plain_Unique Ids1, Plain_Unique Ids2, Plain_Unique Ids3, Plain_Unique Ids4, Plain_Unique Ids5, Plain_Unique Ids6, Plain_Unique Ids7]

# Decrypted User Shares
>>> As = [Plain_Share1, Plain_Share2, Plain_Share3, Plain_Share4, Plain_Share5, Plain_Share6 , Plain_Share7]
#Finds modular quadratic polynomial passing through points given by B_s & A_s.
>>> Regen_Poly (Bs, A_s, mod)
```

```
# Rec_Quad_Poly is the recovered quadratic polynomial.
>>> Rec_Quad_Poly = Regen_Poly ([Bs [0], Bs [1], Bs [2], Bs [3]], [As [0], As [1], As [2], As
[3]], MOD)
```

Eval_Quad_Poly is the original polynomial and Rec_Quad_Poly is the regenerated polynomial calculated using Regen_Poly function.

```
$ ipython
Python 2.7.5 (default, March 30 2014, 09:40:52)

In [1]: execfile ("SecretSharing.py")
In [2]: Eval_Quad_Poly
Out [2]:
Eval_Quad_Poly=
[(44217890321689690135),(95332490871238863097L),(89423100459982786930L),
(39981267850013379301)]

In [3]: Rec_Quad_Poly
Out [3]:
Rec_Quad_Poly=
[(44217890321689690135),(95332490871238863097L),(89423100459982786930L),
(39981267850013379301)]
```

A we can see from Out [2] and Out [3] Eval_Quad_Poly and Rec_Quad_Poly has same output. So with the help of polynomial and secret sharing concept it is easy to reconstruct the shares.

If a part, say B_s [1] and A_s [1] is repeated, an error occurred and the polynomial is not regenerated:

```
In [4]: Polynomial = Regen_Poly ([Bs [0], Bs [1], Bs [1], Bs [3]], [As [0], As [1], As [1], As [3],
MOD)
---------------------------------------------------------------------------
ZeroDivisionError             Traceback (most recent call last)
<ipython-input-5-dbafdc471c87> in <module> ()
----> 1 Polynomial = Regen_Poly ([Bs [0], Bs [1], Bs [1], Bs [3]], [As [0], As [1], As [1], As [3],
MOD)
/home/siya/SSS/SecretSharing.py in Regen_Poly (Bs, As, mod)
20                  for j in range (0, 3):
21                       if (j != i):
---> 22      coeffspart [2] *= gmpy.divm (1, Bs[i] – Bs [j], mod)
23             coeffspart [1] -= Bs [j]
24             coeffspart [0] *=   - Bs [j]
ZeroDivisionError: not invertible
```

Also, if a part, say A_s [3] in place of A_s [2], is repeated, an error occurred and the polynomial is not regenerated:

```
In [5]: Polynomial2 = Poly_Regen ([Bs [0], Bs [1], Bs [2], Bs [3] ], [As [0], As [1], As [3], As
[3]], MOD)
```

In [6]: Polynomial2
Out [6]: [26145891043882953800, 333570718118861434, 7960423652924329178, 98813670053279148236]

3.3. Demonstration of the Scheme with Example

Now we can also use the concept of secret sharing to keep the documents such as files, images safe. The file or image can be shared among the participants.

```
Enter the Number of shares you want to generate from a secret (n) :  7
Enter the threshold value (k) : 4

Shares:
[(13026962538231705647L),(548623285850384019),(1096708402253082360),(1588268
7035326081992L),(10674522843905371509L),       (16198329628412310189L)        ,
(11450824535861532127L)]

Unique Ids :
[(15963329170807749408L)   ,(14164439201954913113L),   (3852488410425486723),
(896391978242292738),(37069081006830707926L),(9726351682425429984),
(3960423652794329163L)]

Enter 1st ID File:'id1'
Enter 1st Pass File:'pass1'

Enter 2nd ID File:'id0'
Enter 2nd Pass File:'pass0'

Enter 3rd ID File:'id3'
Enter 3rd Pass File:'pass3'

Enter 4th ID File:'id2'
Enter 4th Pass File:'pass2'

Eval_Quad_Poly:
[(44217890321689690135)   ,(95332490871238863097L)   ,(89423100459982786930L),
(39981267850013379301)]

Rec_Quad_Poly:
[(44217890321689690135)   ,(95332490871238863097L)   ,(89423100459982786930L),
(39981267850013379301)]

File to Encrypt: 'file.txt'

Ciphertext AES ALGO =
� M �Z�>L �+�U���j[jv��0�G<z�%F�< $ � �̊̊ �V��4  )\�Y
 � � V�Qs!o������ �H����4f��w�����Q� j��s�E��

Plaintext AES ALGO =
Five different encryption algorithms AES ,DES, 3DES, ARC4 and CAST-128 are used to
encrypt the user id and their shares to make comparative study.
```

Figure 6. Secret Sharing with Encryption

The original or evaluated polynomial constructed in stage one can be used to generate or construct key for image or file encryption on the other hand the recovered polynomial is used as decryption key. As we know if at least k people gather then only we can regenerate the polynomial and that recovered one is used for decryption purpose.

The unique id and share of the user is stored on a 14 different files. The user of the scheme take the shares and their respective ids from k participants stored in different files to reconstruct the shares.

We have lots of symmetric key algorithm to choose for file or user data privacy. AES, DES or any algorithm is employed to maintain integrity of the information . To add more security and strengthen the scheme the keys are hashed using any of the hashing algorithm. Hashing algorithms like MD5 , SHA series create a message digest of the key and that value can be used as key for encryption algorithm.

```
# Hashed 32-byte key for Encryption (Eval_Quad_Poly) #

>>> hash=SHA256.new ()
>>> hash.update(str(Eval_Quad_Poly[0])[0:8]+str(Eval_Quad_Poly[1])[0:8]+str(Eval_Quad
_Poly [2])[0:8]+ str(Eval_Quad_Poly [3])[0:8])

# The Key_Encr is used to encrypt user information.
>>> Key_Encr = hash.digest ()

# Hashed 32-byte key for Decryption (Rec_Poly) #

>>> hash.update(str(Rec_Quad_Poly[0])[0:8]+str(Rec_Quad_Poly[1])[0:8]+str(Rec_Quad_
Poly [2])[0:8]+ str(Rec_Quad_Poly [2])[0:8])

# The Key_Dcrp is used to decrypt user information.
>>> Key_Dcrp = hash.digest()

# The Key_Encr and Key_Dcrp are 32-byte hashed key pass as key argument in AES
algorithm.

>>> Obj1/2 = AES.new(Key_Encr/Key_Dcrp, AES.MODE_CBC, 'This is an IV456')
```

Now the user has ask to choose the file or image documents to encrypt.

3.4. A comparative study among different encryption algorithms

There are lots of symmetric key algorithms to protect integrity & privacy of the user secret. The secret sharing scheme uses five different symmetric algorithm to perform a comparative study. The main motivation behind such comparison is to gather information or strength of algorithms. The study shows which algorithm perform better and strongest, securest among other. A study to know which algorithm has less time complexity. This task can be done by generating the histogram of each ciphertext file of all five algorithm. The histogram of plaintext file is same for all algorithm.

Histogram of plaintext file for AES,DES,3DES,ARC4 and CAST –

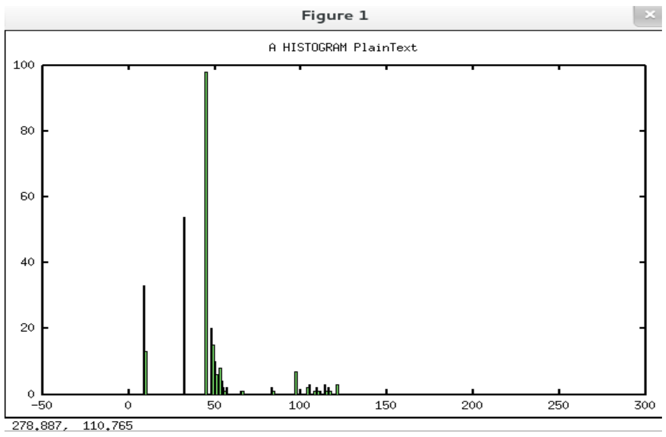

Figure 7. Histogram of Plaintext File for AES,DES,3DES,ARC4 and CAST

The histogram of ciphertext file for all 5 algorithms are different.

1. Histogram of ciphertext file for AES:

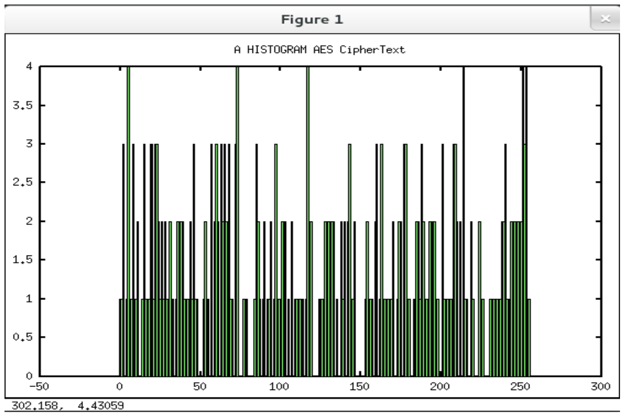

Figure 8. Histogram of Ciphertext file for AES

2. Histogram of ciphertext file for DES:

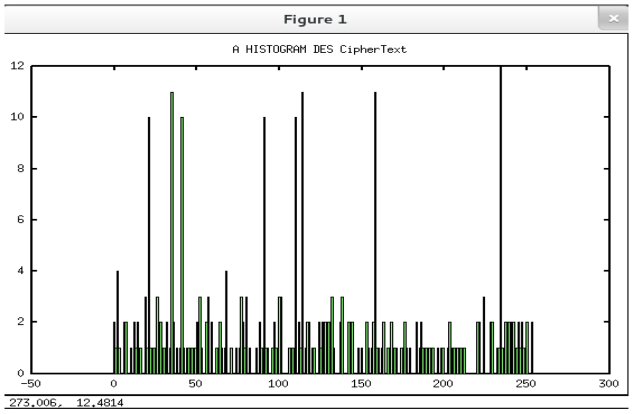

Figure 9. Histogram of Ciphertext file for DES

3. Histogram of ciphertext file for 3DES:

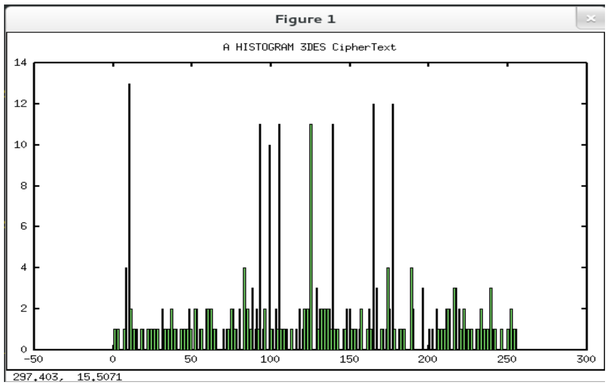

Figure 10. Histogram of Ciphertext File for 3DES

4. Histogram of ciphertext file for ARC4:

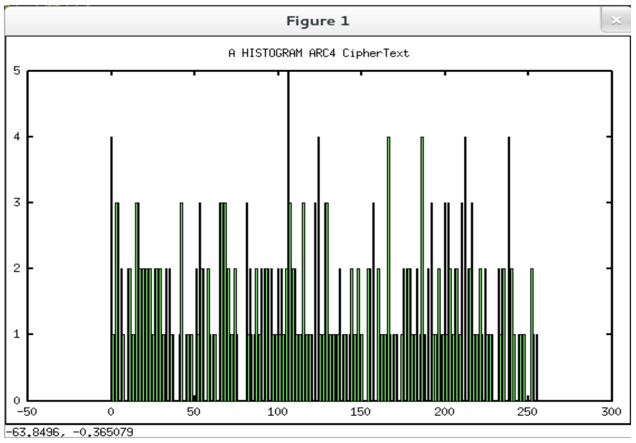

Figure 11. Histogram of Ciphertext File for ARC4

5. Histogram of ciphertext file for CAST:

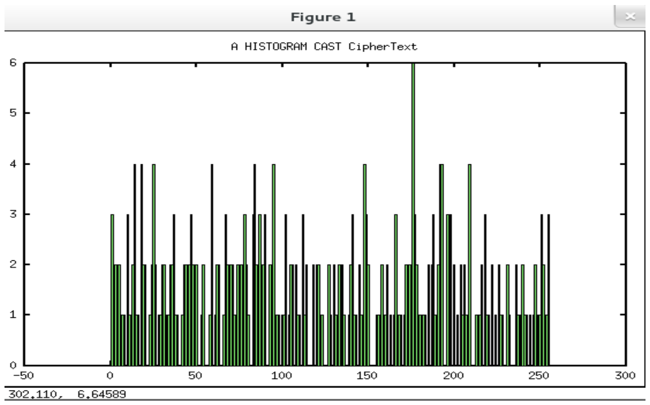

Figure 12. Histogram of Ciphertext file for CAST

4. Conclusion and Future Work

The secret sharing scheme is not a new concept in the context of security, in 1979 Adi Shamir was first who propose the concept . The main logic and aim behind such a purpose is to secure privacy and integrity of user data. The manager of the organization is always trying to make coordination among mutually suspicious individuals with conflicting interests. It is also the responsibility of the security officer of the organization to prevent employees or company's secrets from unauthorized parties these scheme is also useful for medical records and EPR hiding [13].

The proposed scheme is trying to enhance these concept with cryptographic techniques. In Shamir's scheme there some flaws such as dishonest dealer and malicious participants. Encryption of user share with encryption algorithm provides more security to the secrets. The share and also the documents such as text files, images etc are encrypted with different encryption algorithm to analyze the best one. The proposed work use five algorithms AES, DES, 3DES, ARC4 and CAST. The comparative study is done using histogram. The histogram clearly shows that advanced encryption standard (AES) is securest among other algorithms. It takes less time to encrypt and decrypt the secret as compare to other algorithms.

References

[1] A. Shamir, "How to share a secret", Communications of the ACM 22.11 **(1979)**, pp. 612-613.
[2] G. R. Blakley, "Safeguarding Cryptographic Keys", AFIPS 1979 Nat. Comput. Conf., **(1979)**, pp. 313-317.
[3] J. He and E. Dawson, "Multisecret-sharing scheme based on one-way function", Electronics Letters, vol. 31, no. 2, **(1995)**, pp. 93-95.
[4] C.-C. Yang, T.-Y. Chang and M.-S. Hwang, "A (t, n) multi-secret sharing scheme", Applied Mathematics and Computation, vol. 151, no. 2, **(2004)**, pp. 483-490.
[5] B. Chor, S. Goldwasser, S. Micali and B. Awerbuch, "Verifiable secret sharing and achieving simultaneity in the presence of faults", 2013 IEEE 54th Annual Symposium on Foundations of Computer Science. IEEE, **(1985)**, pp. 383-395.
[6] Harn, Lein. "Efficient sharing (broadcasting) of multiple secrets." IEE Proceedings-Computers and Digital Techniques 142.3 **(1995)**: 237-240.
[7] Shao, Jun, and Zhenfu Cao. "A new efficient (t, n) verifiable multi-secret sharing (VMSS) based on YCH scheme." Applied Mathematics and Computation 168.1 **(2005)**: 135-140.
[8] Hwang, Ren-Junn, and Chin-Chen Chang. "An on-line secret sharing scheme for multi-secrets." Computer Communications 21.13 **(1998)**: 1170-1176.
[9] Chien, Hung-Yu, J. A. N. Jinn-Ke, and Yuh-Min Tseng. "A Practical (t, n) Multi-Secret Sharing Scheme." IEICE transactions on fundamentals of electronics, communications and computer sciences 83.12 **(2000)**: 2762-2765.
[10] He, Jingmin, and Edward Dawson. "Multistage secret sharing based on one-way function." Electronics Letters 30.19 **(1994)**: 1591-1592.
[11] Behrouz A. Forouzan, Cryptography and Network Security, the Mc-Graw Hil Companies, **(2007)**.
[12] William Stallings, Cryptography and Network Security, 4/E. Pearson Education India, **(2006)**.
[13] Ulutas, Mustafa, Güzin Ulutas, and Vasif V. Nabiyev. "Medical image security and EPR hiding using Shamir's secret sharing scheme." Journal of Systems and Software 84.3 **(2011)**: 341-353.
[14] Zhao, Jianjie, Jianzhong Zhang, and Rong Zhao. "A practical verifiable multi-secret sharing scheme." Computer Standards & Interfaces 29.1 **(2007)**: 138-141.

Authors

Siyaram Gupta, received Graduation (B.E.) degree from Institute of Technology Guru Ghasidas VishwaVidyalay(ITGGVV), Bilaspur in 2011. He is currently pursuing M.Tech degree in the department of Information Security Management from Dehradun Institute of Technology. He has 3 publications in international journals and one in international conference. His research interests includes cryptography.

Madhu Sharma, Madhu Sharma received her B.E in Computer Science from the R.G. Technical University, Bhopal, India in 2001 and M.E. in Software Engineering from the Thapar University, Patiala, India in 2005. She is presently an Assistant professor with the Department of Computer Science in the DIT University, Dehradun, India. Her research interests include cryptography, image processing, watermarking, steganography and cryptography with cloud computing.