An enhanced python based approach of secret sharing scheme with encryption

Siyaram Gupta
Department of Information Technology
Dehradun Institute of Technology
Dehradun, India
siyaram421@gmail.com

Sumita Lamba
Department of Computer Science
Dehradun Institute of Technology
Dehradun, India
sumitadatsme@gmail.com

Abstract— There are lots of protocols and algorithms which define the security of users' assets. Secret sharing scheme is one among them. To achieve the motivation of individual privacy along with integrity of the information a sharing scheme with the power of cryptographic techniques is proposed. According to the well-known (k, n) threshold scheme k shareholders from pool of n must gather to access or use the secret [2]. The Quadratic polynomial constructs the shares for participants [3] and the shares are encrypted using some sort of encryption algorithms to strengthen the proposed concept. Secret sharing is not a new concept but this paper implement the scheme in python language and encryption of the shares with hashed key that is used to encrypt the share makes the scheme more relevant to its users.Python cryptography tool, "pycrypto" is used to implement the concept of secret sharing.

Keywords- Encryption; Secret Sharing; Polynomials; Python; Hashing.

I. INTRODUCTION

Secrecy and privacy of the information is one of the major issues in digital world. However it is responsibility of the owner of the information to secure it from disclosure and unauthorized access. Using insecure channel for transmission of information may harm the integrity of the data. So the main concern is to secure such type of information which are stored on disk or transmit via transmission channel. There are some cryptographic techniques used at the time of data transmission through which user can assure that user's information should not be tempered in any aspect. Adi Shamir in 1979 proposed a scheme through which a secret can be distributed among a number of parties. According to his scheme, the secret is divided into n pieces and distributed among n-participants so that each participant has their own shares. They also have their unique identification number for validation purpose. If at least 'k' shareholders with their respective unique id meet together, only then the secret is revealed. The dealer or the owner distributes the secret in such a way that each person gets his part. It is not possible for (k-1) number of participants to recover the secret.

The dealer can enhance the concept by using some cryptographic techniques such as encryption, hashing etc. In the proposed scheme, the owner of secret can use encryption technique. The dealer encrypts the share with his own key to enhance the performance of secret sharing scheme along with privacy of user data. Advanced Encryption Standard (AES) [13-14] algorithm is used by dealer to encipher the participant's share. AES is the symmetric-key block cipher standard published which has 128-bits block size with three different key sizes of 128, 192, 256 bits. The reason for selecting AES is 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.

So this paper demonstrates how to secure critical user information by using AES encryption algorithm to enhance the power of secret sharing scheme.

II. RELATED WORKS

Shamir (1979) [2] and Blakley [4] introduced the concept of secret sharing scheme to keep the secret confidential and privacy of the information. The scheme has lots of applications such as hiding medical images and patient's records, missile codes, bank account, encryption keys etc.

However, as this scheme is discussed, there are several drawbacks such as dishonest dealer, malicious shareholders may distribute fake share to the other participants, limited number of shares can be shared during the process. The issues are further improved and fixed by other scholars.

To overcome the drawback of single share a multi-secret sharing (MSS) scheme has been proposed [11-12]. In 2004 two-variable one-way function [5] multi-secret sharing (MSS) was proposed by C.C. Yang, T.-Y. Chang, M.S. Hwang [6]. A verifiable secret sharing (VSS) scheme has been proposed to overcome the problem of dishonest dealers and malicious participant. The scheme allows participants to verify the validity of the shares of other participants and his own. A VSS was written by Chor et al. [7] in 1985.

Harn [8] presented the verifiable multi-secret sharing (VMSS) scheme in 1995. The YCH scheme is relatively efficient multi-secret scheme at the present time. But the issue is that the scheme doesn't have the property of verification. Because of its big-ticket system, a new literature [9] that has a property of verification based on YCH scheme is impractical. later a new practical verifiable multi-secret sharing scheme proposed by Jianjie Zhao ,Jianzhong Zhang ,Rong Zhao [3] to overcome the drawback of the scheme which is based on discrete logarithm [10].

III. BRIEF INTRODUCTION TO SHAMIR'S SHARING SCHEME AND PYTHON

A. Shamir's Sharing Scheme

The secret sharing scheme proposed by Shamir divides the secret into n pieces so that each participant gets his share. The participants also have a unique identity. Participants are able to reconstruct the share only when k of them out of n gathers. Any number of participants less than k is not permitted to recover the secret. Shamir uses Lagrange's interpolation polynomial to construct shares for the participants.

A secret say S is divided into n pieces or shares. A (k-1) degree polynomial with a large prime number M is employed for share construction as shown in equation (1).

$$M(x) = (b_1 x + b_2 x^2 + \cdots + b_{k-1} x^{k-1}) \bmod P \qquad (1)$$

$(b_1, b_2, \cdots, b_{k-1})$ are the coefficient of the polynomial selected randomly from within range (0, P]. The computed shares are shown in equation (2).

$$Sh_1 = (1, M(1)), Sh_2 = (2, M(2)), Sh_3 = (3, M(3)). \ldots\ldots\ldots Sh_n = (n, M(n)). \qquad (2)$$

So each of the participants has two integer values one a unique identification and other is the share. The term S of the polynomial is the secret, which is a constant.

B. Introduction to Python

In 1991, Guido van Rossum was the first to introduce python. It is an open source and user friendly language. Python has huge collection of tools, libraries, functions and module that helps 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 describes 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, to implement digital signatures, or, in association with a public-key algorithm. Examples of cryptographically secure hash functions include SHA12, SHA256, MD2 and MD5.

Here's an example, using 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: Encryption Algorithms: Encryption algorithms transform their input data, or plaintext, in some way that is dependent on a variable key, producing cipher text. The currently available block ciphers in the Crypto. Cipher package are AES, DES, CAST, ARC4 etc.

An example usage of the DES module:

```
>>> from Crypto. Cipher import DES
>>> obj=DES.new ('abcdefgh', DES.MODE_ECB)
>>> plain="Guido Rossum introduce python language."
>>> ciph=obj.encrypt (plain)
>>> obj.decrypt (ciph)
```

GMPY Function: GMPY is multiple-precision arithmetic C-coded python module that supports for the MPFR and MPC libraries. It uses mpf or mpfr type to support multiple-precision real. 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 provide 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])
```

Applications such as medical image security, EPR hiding [1] implements this scheme to keep their records secret.

IV. PROPOSED WORK

The scheme demonstrates the logic how a secret can be shared so that it is recovered only when sufficient number of shares is available. Logic is that the encrypted shares of a secret being shared among five people. Out of those five, if at least three of them give their shares along with unique id, the secret can be regenerated. The participants decipher the share to use the desired secret.

A. Secret sharing with python

In the Shamir's (k, n) we take (3, 5)**:**

The proposed scheme chooses a largest 64-bit prime number for modulus calculations. The modular arithmetic will become more impractical and tedious if the mod is not a prime number. Quadratic polynomial with 3 coefficients is used as the secret to be shared. By evaluating the coefficients (Ys) at randomly chosen Xs the five shares are generated. Three shares out of five are must to recover the secret. Two different functions are used to evaluate and re-generate the secrets. The shares are randomly generated using random functions.

Advance Encryption Standard (AES) algorithm is used to encipher the generated shares. AES with 256-bit key is used to encrypt the share for each participant.

B. Implementation of proposed scheme

The proposed scheme has five main Phases :-

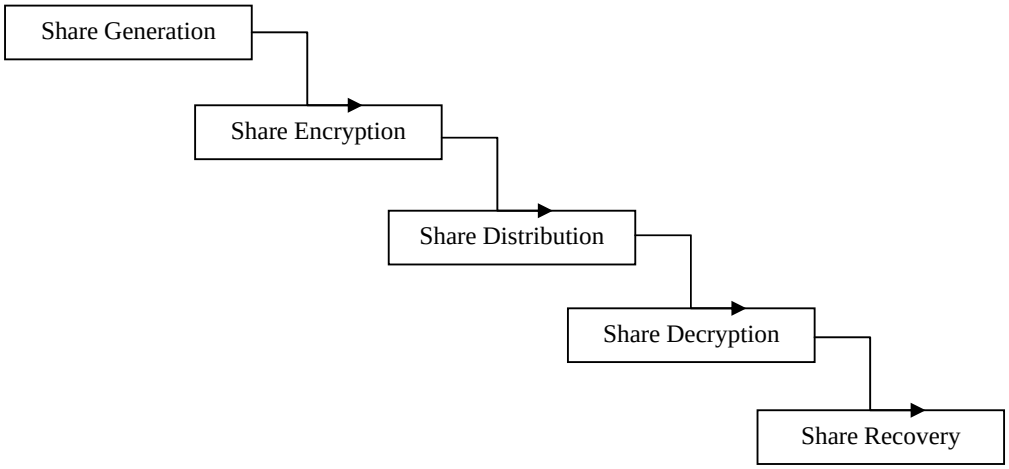

Figure 1. Stages of Secret Sharing

1) Share Construction Phase

P_Mod = gmpy.mpz (2^{64} - 59)

(A Largest 64-bit prime number)

Poly_Eval (coeffs, x, mod)

(This evaluate quadratic modular polynomial defined by coeffs at x, mod is the modulus)

UIDs:
[(11450824535861532127L),(8769961006588243981),(15963329170807749408L),(14164439201954913113L) ,(3852488410425486723)]

Shares:
[(8245331363280405549),(12711414002300077493L),(15882687035326081992L),(10674522843905371509L), (16198329628412310189L)]

Poly_Org:
[mpz(6930013510669805067),mpz(2050487392748600585),mpz(14831304354619697816L)]

Figure 2. Share Construction

Poly_Org = [gmpy.mpz (random.randrange (0, My_mod))]

(Poly_Org is the evaluated or original quadratic polynomial)

X_s = [gmpy.mpz (random.randrange (0, P_Mod))

(Randomly chosen value)

Y_s = [Poly_Eval (Poly_Org, X_s [0], P_Mod)

(Generate share for each participants)

2) Encryption of the shares

The second phase of the scheme is to encipher the shares generated in share construction phase. The shares are encrypted with the key generated by the dealer. AES with 32-byte key is used to encrypt the share Ys along with unique id of participants Xs. Python's cryptographic tool Pycrypto is employed for these purpose.

From Crypto, Cipher import AES

Obj1 = AES.new ("32-byte key", AES.MODE_CBC, 'This is an IV456')

Obj2 = AES.new ("32-byte key", AES.MODE_CBC, 'This is an IV456')

(IV = Initialization Vector)

Cipher_Share = Obj1.encrypt (Ys)

(Share Ys is to be encrypted)

Cipher_UID = Obj2.encrypt (Xs)

(Unique ID Xs is to be encrypted)

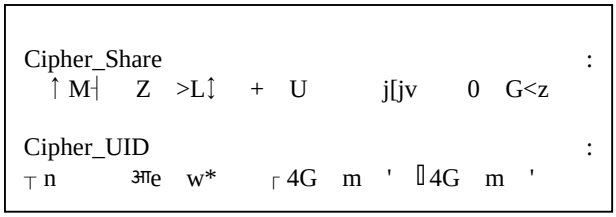

Figure 3. Share Encryption

3) Distribution and sharing process

Now the encrypted shares are to be distributed among five participants. Each of the five shareholders gets their piece of secret. To each of them, a unique id is allotted.

4) Decryption of the Shares

The next phase of the scheme is to decrypt the recovered share. Same algorithm with same key set is used to decipher the shares.

From Crypto, Cipher import AES

Obj3 = AES.new ("32-byte key", AES.MODE_CBC, 'This is an IV456')

Obj4 = AES.new ("32-byte key", AES.MODE_CBC, 'This is an IV456')

(IV = Initialization Vector)

Plain_Share = Obj3.encrypt (Cipher_Share)

(Share Ys is to be decrypted)

Plain_UID = obj4.encrypt (Cipher_UID)

(Unique ID Xs is to be decrypted)

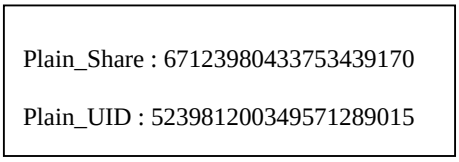

Figure 4. Share Decryption

5) Share Re-construction stage

Now the participants will recover the encrypted shares.

Xs = [Plain_UID1, Plain_UID2, Plain_UID3, Plain_UID4, Plain_UID5] (Decrypted Unique Ids)

Ys = [Plain_Share1, Plain_Share2, Plain_Share3, Plain_Share4, Plain_Share5] (Decrypted Shares)

Poly_Regen (Xs, Y_s, mod)

(Finds modular quadratic polynomial passing through points given by X_s & Y_s)

Poly_Rec = Poly_Regen ([X_s [0], X_s [1], X_s [2]], [Y_s [0], Y_s [1], Y_s [2]], P_mod)

(Rec_Poly is the recovered quadratic polynomial)

Poly_Org is the original polynomial and Poly_Rec is the regenerated polynomial calculated using Poly_Regen function.

$ ipython

Python 2.7.5 (default, Jan 10 2014, 09:40:52)

In [1]: execfile ("mypolym.py")

In [2]: Poly_Org

Out [2]:

Poly_Org:[mpz(6930013510669805067),mpz(2050487392748600585), mpz(14831304354619697816L)]

In [3]: Poly_Rec

Out [3]:

Poly_Rec:[mpz(6930013510669805067),mpz(2050487392748600585), mpz(14831304354619697816L)]

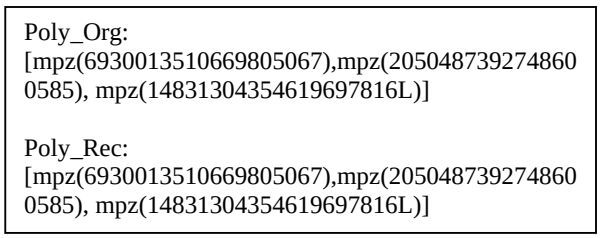

Figure 5. Share Recovery

Out [2] and Out [3] are the outputs of Org_Poly and Rec_Poly respectively. Both outputs show same results, so by using the concept of polynomial and sharing scheme it is easy to evaluate and regenerate shares.

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

In [4]: poly1 = Poly_Regen ([X_s [0], X_s [1], X_s [1]], [Y_s [0], Y_s [1], Y_s [1]], P_mod)

ZeroDivisionError Traceback (most recent call last)

<ipython-input-5-dbafdc471c87> in <module> ()

----> 1 poly1 = Poly_Regen ([X_s [0], X_s [1], X_s [1]], [Y_s [0], Y_s [1], Y_s [1]], P_mod)

/home/ash/myresearch/Cryptography/mypolym.py in Poly_Regen (X_s, Y_s, mod)

28 for j in range (0, 3):

29 if (j != i):

---> 30 coeffspart [2] *= gmpy.divm (1, X_s[i] – Xs[j], mod)

31 coeffspart [1] -= X_s[j]

32 coeffspart [0] *= - X_s[j]

ZeroDivisionError: not invertible

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

In [5]: poly2 = Poly_Regen ([X_s [0], X_s [1], X_s [2]], [Y_s [0], Y_s [1], Y_s [3]], P_mod)

In [6]: poly2

Out [6]: [(13145891074882953838L),

(543580718118861428), (3960423652794329163)]

C. Information sharing using the concept of sss

The user or participant now is able to share important and confidential information using secret sharing scheme. The original evaluated polynomial is used to construct 32-byte strong key for encryption purpose. As shown and implemented if three users among five are gather then we re-generate the secret or say polynomial. The recovered polynomial is used as decryption key.

As we know that AES is a symmetric key algorithm, so it is necessary that both keys are same. Because evaluated polynomial and recovered polynomial is used as encryption and decryption keys respectively, so it is necessary that both are identical. If and only if three shareholders amongst five with their respective ids are gathered and recover the polynomial then only the information is decrypted and meaningful for its user.

The 32-byte keys used for such purpose are hashed by using SHA-256 hashing algorithm to make the key more strong and intangible for the users.

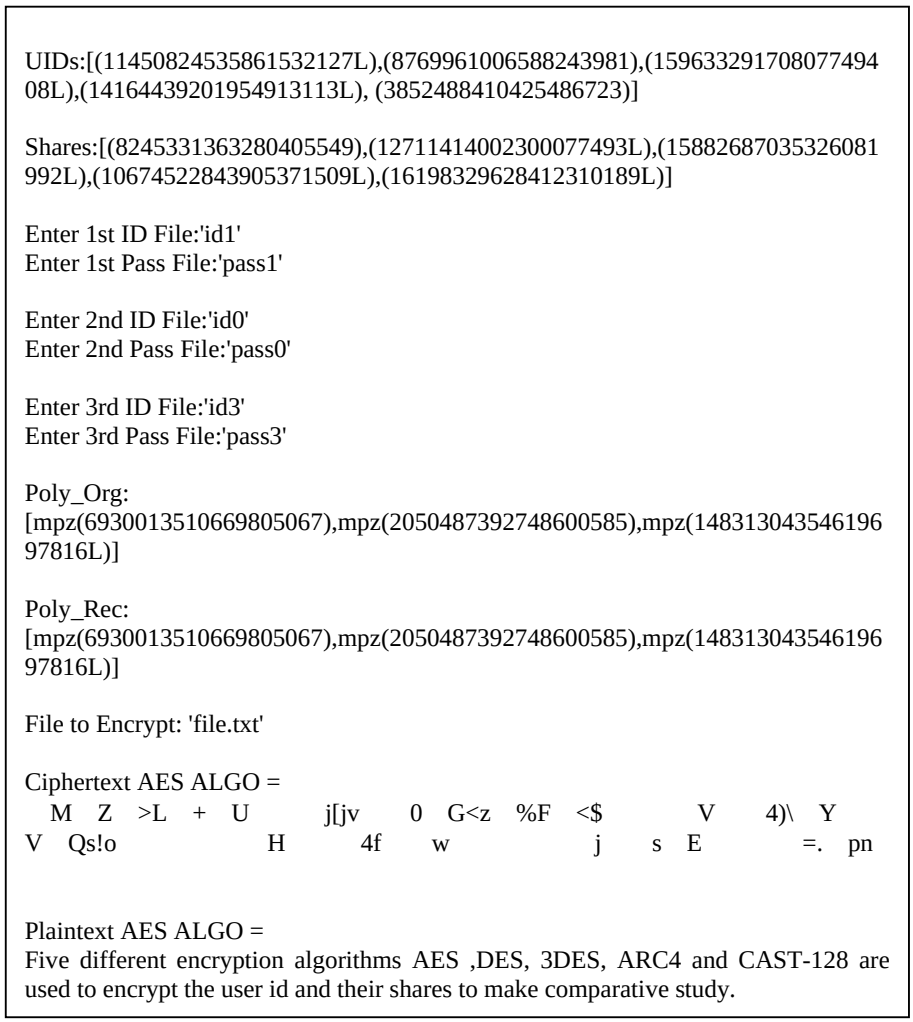

Figure 6. Secret Sharing with Encryption

From Crypto. Hash import SHA256

Hashed 32-byte key for Encryption (Org_Poly)

hash= SHA256.new ()

hash.update (str(Org_Poly[0])[0:11]+str(Org_Poly[1])[0:11]+ str(Org_Poly[2])[0:10])

Enc_Key = hash.digest ()

(The Enc_Key is used to encrypt user information.)

Hashed 32-byte key for Decryption (Rec_Poly)

hash.update (str(Rec_Poly[0])[0:11]+str(Rec_Poly[1])[0:11]+ str(Rec_Poly [2])[0:10])

Dcr_Key = hash.digest()

(The Drc_Key is used to decrypt user information.)

The Enc_Key and Drc_Key is 32-byte hashed key pass as key argument in AES algorithm.

Obj5/6 = AES.new(Enc_Key/Drc_Key, AES.MODE_CBC, 'This is an IV456')

Now the user is asked to choose the file or image documents to encrypt.

V. CONCLUSION AND FUTURE WORK

The proposed concept uses secret sharing scheme first developed by Adi Shamir in 1979. The main property of the methodology is to secure the intellectual work of the people. A group of mutually suspicious individuals with conflicting interests must co-operate so it is necessary for the manager of the organization or network security officer to prevent the disclosure of the secrets for those persons who are not supposed to authorize. Unauthorized participants with a forged key are not able to recover the data because the veto power is not owned by any single participant. The proposed work assures the user that his information is kept secret and is not altered or tempered by malicious user. The scheme fixes the privacy and security issues arise at the time of information transmission. The concept with encryption technique prevent the drawback of dishonest dealer and malicious participant .Use of AES cryptographic tool with python implementation makes the scheme relevant for large applications such as medical image and EPR(Electronic Patient record) hiding etc. The scheme can also be implemented with other algorithms for comparative study.

ACKNOWLEDGMENT

First of all I would like to thank my mentor/guide **Mrs. Madhu Sharma** for all her help, guidance, continuous support throughout my studies and encouragement. Thanks for always taking the time to answer my questions and share her knowledge.

I would like to thank all my friends and especially my classmates **Abdul Mannan, Yogesh Patel, Amritansh Gupta, Vishnu Kamat, Sumita Lamba and Neha Upadhyay.** I shared so many things with them. They also taught me so many virtue. Sincere friendship to start with unselfishness, tolerance and helpfulness. I have enjoyed their companionship so much during my stay at DIT, Dehradun. I am grateful to them for making me a better person.

My sincere appreciation goes to my family especially my father **Shri Santosh Kumar Gupta** my brother **Hemsagar and** my sister **Shweta Gupta** for always encouraging me for pursuing higher degrees. It is not possible to thank them enough but I want them to know that I will be grateful to them throughout my life.

REFERENCES

[1] Medical image security and EPR hiding using Shamir's secret sharing scheme, Mustafa Ulutas, Güzin Ulutas∗, Vasif V. Dept. Of Computer Engineering, Karadeniz Technical University, 61080 Trabzon, Turkey.

[2] A. Shamir, How to share a secret, Communications of the ACM 22 (11) (1979) 612–613.

[3] A practical verifiable multi-secret sharing scheme. Jianjie Zhao, Jianzhong Zhang, Rong Zhao. College of Mathematics and Information Science, Shaanxi Normal University, Xi'an 710062, People's Republic of Chin b Natural Science Institute, Xi'an University of Technology, Xi'an 710048, People's Republic of China.

[4] G. Blakley, Safeguarding cryptographic keys, Proc AFIPS 1979 National Computer Conference, AFIPS Press, New York, 1979, pp. 313–317.

[5] J. He, E. Dawson, Multisecret-sharing scheme based on one-way function, Electronics Letters 31 (2) (1995) 93–95.

[6] C.-C. Yang, T.-Y. Chang, M.-S. Hwang, A (t, n) multi-secret sharing scheme, Applied Mathematics and Computation 151 (2004) 483–490.

[7] B. Chor, S. Goldwasser, S. Micali, B. Awerbuch, Verifiable secret sharing and achieving simultaneity in the presence of faults, Proc. 26th IEEE Symp. FOCS, 1985, pp. 251–260.

[8] L. Harn, Efficient sharing (broadcasting) of multiple secret, Computers and Digital Techniques 142 (3) (1995) 237–240.

[9] J. Shao, Z.-F. Cao, A new efficient (t, n) verifiable multi-secret sharing (VMSS) based on YCH scheme, Applied Mathematics and Computation 168 (2005) 135–140.

[10] R.-J. Hwang, C.-C. Chang, An on-line secret sharing scheme for multisecrets, Computer Communications 21 (13) (1998) 1170–1176.

[11] H.-Y. Chien, J.-K. Tseng, A practical (t,n) multi-secret sharing scheme, IEICE Transactions on Fundamentals of Electronics, Communications and Computer 83-A (12) (2000) 2762–2765.

[12] J. He, E. Dawson, Multistage secret sharing based on one-way function, Electronics Letters 30 (19) (1994) 1591–1592.

[13] Cryptography and Network Security- the Mc-Graw Hil Companies, Behrouz A. Forouzan.

[14] Cryptography and Network Security Principles and Practices, William Stallings.