ISSN:0975-9646

Simple and Efficient Secret Sharing Schemes for Sharing Data and Image

Binu V P[#], Sreekumar A[*]

[#]*Department of Computer Applications*
Cochin University Of Science and Technology,Kerala, India
[*] *Department of Computer Applications*
Cochin University Of Science and Technology,Kerala, India

Abstract—Secret sharing is a new alternative for outsourcing data in a secure way. It avoids the need for time consuming encryption decryption process and also the complexity involved in key management. The data must also be protected from untrusted cloud service providers. Secret sharing based solution provides secure information dispersal by making shares of the original data and distributes them among different servers. Data from the threshold number of servers can be used to reconstruct the original data. It. is often impractical to distribute data among large number of servers. We have to achieve a trade off between security and efficiency. An optimal choice is to use a (2, 3) or (2, 4) threshold secret sharing scheme, where the data are distributed as shares among three or four servers and shares from any two can be used to construct the original data. This provides security, reliability and efficiency. We propose some efficient and easy to implement secret sharing schemes in this regard based on number theory and bitwise XOR. These schemes are also suitable for secure sharing of images. Secret image sharing based on Shamir's schemes are lossy and involves complicated Lagrange interpolation. So the proposed scheme can also be effectively utilized for lossless sharing of secret images.

Keywords— **Shamir's Secret Sharing, Secure Data Storage, Secret Image Sharing Introduction**

I. INTRODUCTION

The secret sharing schemes are originally proposed by Shamir [1] and Blackley [2] in 1979. The motivation was to safeguard cryptographic keys. Their solution was to store the secret keys at several locations as shares and when authorized number of users collaborates together, they can retrieve the secret. The schemes are (t, n) threshold schemes where any t number of users can collaborate to recover the secret out of n users. This provides security, reliability and convenience. Shamir's scheme is simple and easy to implement and is based on polynomial interpolation. Blackley's scheme has a different approach and is based on hyper plane geometry. But it is difficult to implement. Secret sharing schemes have found numerous applications in designing several cryptographic protocols. Threshold cryptography [3], access control [4], secure multi-party computation [5] [6] [7], attribute based encryption [8] [9], generalized oblivious transfer [10] [11], visual cryptography[12] etc..., are some of the important areas where secret sharing schemes are used. In this paper we suggest efficient secret sharing schemes for the reliable and secure distributed storage of data on untrusted servers.

Shamir's scheme is based on polynomial interpolation over a finite field. It uses the fact that we can construct a polynomial of degree t-1 only if t data points are given. The scheme is based on polynomial interpolation. Given t points in the 2-dimensional plane (x_i, y_i)… (x_t, y_t), with distinct x_i's, there is one and only one polynomial P(x) of degree t-1 such that P(x_i) = y_i for all i. In order to share the secret S, pick a random t-1 degree polynomial $P(x) = a_0 + a_1 x + ... + a_{t-1} x_{t-1}$ with a_0 = S, and evaluate shares as S1= P(1),S2= P(2),…, Si = P(i), … ,Sn= P(n).Any subset of t of these shares Si together with their identifying indices i, we can find the coefficients of P(x) by interpolation, and then evaluate S=P(0).The knowledge of just t-1 of these values, does not suffice in order to calculate S. Efficient O(nlog^{2}n) algorithms exist for the evaluation and interpolation of polynomials.

A secret sharing scheme is called perfect if less than t shares give no information about the secret. It is known that for a perfect secret sharing scheme H(Si) >=H(S). If H(Si) =H(S) then the secret sharing scheme is called ideal. Shamir's scheme is perfect and ideal.Blackley's scheme is not perfect.

Confidentiality, reliability and efficiency are the major concerns in secure storage of data. The idea of secret sharing for the information dispersal is suggested by Krawczyk et al [13] in 1994.He proposed a computationally secure secret sharing scheme for the distributed storage using Rabin's [14]
Information Dispersal Algorithm (IDA) and Shamir's secret sharing scheme. However the data is encrypted using a symmetric key encryption and the share of the key is distributed along with the data shares. The share size is less than the secret in this case compromising the information theoretic security. Abhishek Parak et al [15] in 2010 proposed a space efficient secret sharing scheme for the implicit data security. They incorporated k-1 secrets in n shares and any k shares can be used to reconstruct the original secret. A recursive construction using Shamir's scheme is applied in which computational over head is more. Recursive methods of secret sharing is also mentioned in [16], [17].Computational secret sharing schemes are proposed for the space efficiency in [18],[19],[20].

Secret sharing based solution provides information theoretical security on confidentiality without encryption and hence avoids the complexities associated with encryption and key management. It also provides the guarantee on availability of data. Perfect secret sharing needs large amount of computational overhead. We propose specially designed secret sharing schemes using XOR and number theoretic technique to reduce the computation overhead. Unanimous consent schemes are easy to implement using XOR.But the implementation of a general (t, n) threshold scheme is difficult. Wang et al [21] proposed a scheme based on Boolean operation which is used for secret image sharing in 2007.Kurihara et al [23],[22] proposed a (3, n) and a generalized (t, n) secret sharing scheme based on simple XOR operations. Efficient and ideal threshold scheme based on XOR is proposed by Lv et al [24] in 2010. Secret sharing using number theoretic schemes are also developed based on Chinese reminder theorem [26],[25],[27].They are not widely used because of the computational complexity. The proposed scheme makes use of simple number theoretic concept and the Euclid's algorithm.

II. Proposed Secret Sharing Schemes

The proposed system suggests a method of storing and retrieving private data in a secure and effective manner. The private data include personal information, sensitive information or unique identification etc. The data storage may be a private information storage using cloud database. We propose number theoretic and XOR based scheme for efficient implementation of secret sharing scheme. It can be used for secure storage and retrieval. Since it does not involve any encryption, the PKI needed for key management can be avoided. Section 2.A contains the detailed description of the secret sharing algorithm using number theoretic concept. Section 2.B explains the XOR based schemes. The algorithms mentioned below are designed to share one byte of data at a time. The scheme can be used to share both textual data and images.must be indented. All paragraphs must be justified, i.e. both left-justified and right-justified.

A. Schemes Based on Number Theory

In this section the proposed secret sharing schemes which are based on number theoretic concepts are explained in detail. Two threshold secret sharing schemes of order (2, 3) and (2,4) are proposed. The Algorithm 1 is the (2,3) secret sharing phase and the retrieval algorithms depend on which shares are used for the reconstruction and are given in Algorithms 2,3,4.A (2,4) secret sharing scheme is mentioned in Algorithm 5.The secret revealing algorithms corresponds to different combination of shares are given in Algorithms 6,7,8,10,11.The algorithms use simple number theory concept. In order to find the inverse of a number extended Euclid's algorithm can be used. The share generation can be done with a complexity of O(n) and the secret revealing can also be done with a complexity of O(n), where n is the number of bytes to share. Table lookup can be used for faster performance.

ALGORITHM 1: (2,3) SECRET SHARING: NUMBER THEORY

```
Data: Input file S to share.
Result: Three Shares S1,S2,S3 of same size as the original file.
Choose a field Zp where p = 257.

while not at end of the input file do
s=read_byte(S) // read a byte or pixel
if s == 0 then
        s = 256
end
$a = s^{(p-1)/3}$   //find cube root of s
r=random(257)  // random number between 0-256
s1 = r * a mod p  // s1 is the share1 pixel
if s1 == 256 then
        s1 = 0
end
s2 = $r^2$ * a mod p  // s2 is the share2 pixel
if s2 == 256 then
        s2 = 0
end
s3 = $r^4$ * a mod p    // s3 is the share3 pixel
if s3 == 256 then
        s3 = 0
end
end
```

ALGORITHM 2: (2,3) SECRET REVEALING:NUMBER THEORY S1S2

```
Data: Shares S1 and S2
Result: The original secret file S which is shared
Choose a field Zp where p = 257.
while not at end of the input files do
s1=read_byte(S1) // read a byte or pixel from S1
s2=read_byte(S2) // read a byte or pixel from S2
if s1 == 0 then
        s1 = 256
end
if s2 == 0 then
        s2 = 256
end
a = $s1^2$ *$s2^{-1}$ mod p
s = $a^3$ mod p; // s is the secret data byte or pixel
if s == 256 then
        s = 0
end
end
```

ALGORITHM 3: (2,3) SECRET REVEALING:NUMBER THEORY S1S3

```
Data: Shares S1 and S3
Result: The original secret file S which is shared
Choose a field Zp where p = 257.
while not at end of the input files do
s1=read_byte(S1) // read a byte or pixel from S1
s3=read_byte(S3) // read a byte or pixel from S2
if  s1 == 0 then
        s1 = 256
end
if  s3 == 0 then
        s3 = 256
```

```
end
s = $s1^4 * s3^{-1}$ mod p // s is the secret data byte or pixel
if s == 256 then
        s = 0
end
end
```

ALGORITHM 4: (2,3) SECRET REVEALING:NUMBER THEORY S2S3

```
Data: Shares S2 and S3
Result: The original secret file S which is shared
Choose a field Zp where p = 257.
while not at end of the input files do
s2=read_byte(S2) // read a byte or pixel from S1
s3=read_byte(S3) // read a byte or pixel from S2
if s2 == 0 then
        s2 = 256
end
if s3 == 0 then
        s3 = 256
end
a = $s2^2 * s3^{-1}$ mod p
s = $a^3$ mod  p; // s is the secret data byte or pixel
if s == 256 then
        s = 0
end
end
```

ALGORITHM 5: (2,4) SECRET SHARING:NUMBER THEORY

```
Data: Input file S to share.
Result: Four Shares S1,S2,S3,S4 of same size as the original file.
Choose a field Zp where p = 257.
while not at end of the input file do
s=read_byte(S) // read a byte or pixel
if s == 0 then
        s = 256
end
r=random(257) // random number between 0-256
s1 = r     // s1 is the share1 pixel
if  s1 == 256 then
        s1 = 0
end
s2 = r * s mod p // s2 is the share2 pixel
if  s2 == 256 then
        s2 = 0
end
s3 = $r^2$ * s mod p // s3 is the share3 pixel
if  s3 == 256 then
        s3 = 0
end
s4 = $r^3$ * s mod p //s4 is the share4 pixel
if s4 == 256 then
        s4 = 0
end
end
```

ALGORITHM 6: (2,4) SECRET REVEALING:NUMBER THEORY S1S2

```
Data: Shares S1 and S2
Result: The original secret file S which is shared
Choose a field Zp where p = 257.
while not at end of the input files do
s1=read_byte(S1) // read a byte or pixel from S1
s2=read_byte(S2) // read a byte or pixel from S2
if s1 == 0 then
        s1 = 256
end
if s2 == 0 then
        s2 = 256
end
s = $s1 * s2^{-1}$ mod p
if s == 256 then
        s = 0
end
end
```

ALGORITHM 7: (2,4) SECRET REVEALING:NUMBER THEORY S1S3

```
Data: Shares S1 and S3
Result: The original secret file S which is shared
Choose a field Zp where p = 257.
while not at end of the input files do
s1=read_byte(S1) // read a byte or pixel from S1
s3=read_byte(S3) // read a byte or pixel from S3
if  s1 == 0 then
        s1 = 256
end
if s3 == 0 then
        s3 = 256
end
s = $(s1^2)^{-1} * s3$ mod p
if  s == 256 then
        s = 0
end
end
```

ALGORITHM 8: (2,4) SECRET REVEALING:NUMBER THEORY S1S4

```
Data: Shares S1 and S4
Result: The original secret file S which is shared
Choose a field Zp where p = 257.
while not at end of the input files do
s1=read_byte(S1) // read a byte or pixel from S1
s4=read_byte(S4) // read a byte or pixel from S4
if s1 == 0 then
        s1 = 256
end
if s4 == 0 then
        s4 = 256
end
s = $(s1^3)^{-1} * s4$ mod p
if  s == 256 then
        s = 0
end
end
```

ALGORITHM 9: (2,3) SECRET REVEALING:NUMBER THEORY S2S3

```
Data: Shares S2 and S3
Result: The original secret file S which is shared
Choose a field Zp where p = 257.
while not at end of the input files do
s2=read_byte(S2) // read a byte or pixel from S2
s3=read_byte(S3) // read a byte or pixel from S4
if s2 == 0 then
    s2 = 256
end
if s3 == 0 then
    s3 = 256
end
$s = s2^2 * s3^{-1} \bmod p$
if s == 256 then
        s = 0
end
end
```

ALGORITHM 10: (2,4) SECRET REVEALING:NUMBER THEORY S2S4

```
Data: Shares S2 and S4
Result: The original secret file S which is shared
Choose a field Zp where p = 257.
while not at end of the input files do
s2=read_byte(S2) // read a byte or pixel from S2
s4=read_byte(S4) // read a byte or pixel from S4
if   s2 == 0 then
   s2 = 256
end
if  s4 == 0 then
   s4 = 256
end
$s = \mathrm{sqrt}(s2^3 * s4^{-1} \bmod p)$
if  s == 256 then
  s = 0
end
end
```

ALGORITHM 11: (2,4) SECRET REVEALING:NUMBER THEORY S3S4

```
Data: Shares S3 and S4
Result: The original secret file S which is shared
Choose a field Zp where p = 257.
while not at end of the input files do
s3=read_byte(S3) // read a byte or pixel from S2
s4=read_byte(S4) // read a byte or pixel from S4
if  s3 == 0 then
     s3 = 256
end
if  s4 == 0 then
     s4 = 256
end
$s = s3^3 * (s4^2) - 1 \bmod p$
if  s == 256 then
     s = 0
end
end
```

B. Schemes based on XOR

An (n, n) scheme using XOR can easily be setup by creating n-1 random shares of same size as the secret and the nth share as the XOR of these n-1 shares and the secret k. The secret can be revealed by simply XOR ing all the shares. In this we propose two schemes. An ideal (2, 3) scheme where the size of the share is same as that of the secret is mentioned in Algorithm 16 and a non ideal scheme which is also not perfect is mentioned in Algorithm 12. In this the size of the share is reduced to half. The scheme can be used when the storage become a constraint. The secret sharing and revealing can be done in time O (n), where n is the number of bytes to share. The secret reconstruction corresponds to different combination of shares in the non ideal scheme are mentioned in Algorithms 13,14,15 and in the ideal schemes are mentioned in Algorithms 17,18,19.

ALGORITHM 12: (2,3) XOR SECRET SHARING-NON IDEAL

```
Data: Secret file S to share.
Result: Three shares S1,S2 and S3 of half the size of S.
while not at end of the input files do
s=read_byte(S) // read a byte or pixel from S
bs=binary(s)    // bs is the binary representation of s
// odd bits of bs taken as share1 data nibble s1
s1=odd bits(bs)
// even bits of bs taken as share2 data nibble s2
s2=even bits(bs)
//share3 nibble is formed by xoring s1 and s2
$s3 = s1 \oplus s2$
end
```

ALGORITHM 13: (2,3) XOR SECRET REVEALING S1S2-NON IDEAL

```
Data: Share S1 and S2
Result: The original secret file S which is shared.
while not at end of the input files do
s1=read_byte(S1)   // read a byte or pixel from S1
s2=read_byte(S2)   // read a byte or pixel from S2
s = intermix(s1,s2) // intermix the bits of s1 and s2 to
                       construct the secret byte
end
```

ALGORITHM 14: (2,3) XOR SECRET REVEALING S1S3-NON IDEAL

```
Data: Share S1 and S3
Result: The original secret file S which is shared.
while not at end of the input files do
s1=read_byte(S1) // read a byte or pixel from S1
s3=read_byte(S3) // read a byte or pixel from S3
$s2 = s1 \oplus s3$
// intermix the bits of s1 and s2 to construct the
secret byte
s = intermix(s1, s2) // intermix the bits of s1 and s2
to construct the secret byte
end
```

ALGORITHM 15: (2 ,3) XOR SECRET REVEALING S2S3-NON IDEAL

```
Data: Share S2 and S3
Result: The original secret file S which is shared.
while not at end of the input files do
s2=read_byte(S2) // read a byte or pixel from S2
s3=read_byte(S3) // read a byte or pixel from S3
$s1 = s2 \oplus s3$
s = intermix(s1,s2) // intermix the bits of s1 and s2
to construct the secret byte
end
```

ALGORITHM 16: (2,3) XOR IDEAL SECRET SHARING

```
Data: Input file S to share.
Result: Three Shares SH1,SH2,SH3 of same size as the
original file.
while not at end of the input file do
s=read_byte(S)     // read a byte or pixel
r=random(257)   // random number between 0-256
s1,s2=split_two(s)  // split s into 2 nibbles
r1, r2=split_two(r) // split r into 2 nibbles
s0 = 0000        // a dummy variable initialized to zero
$sh1 = s0 \oplus r1||s2 \oplus r2$ // sh1 is the share1 pixel and
                                ’||’ is concatenation operation
 $sh2 = s1 \oplus r1 \oplus ||s0 \oplus r2$
//sh2 is the share2 pixel
 $sh3 = s2 \oplus r1||s1 \oplus r2$ //sh3 is the share3pixel
end
```

ALGORITHM 17: (2,3)XOR IDEAL SECRET RECOVERY SH1SH2

```
Data: Shares SH1 and SH2
Result: Original secret S that is shared
while not at end of the input files do
sh1=read_byte(SH1) // read a byte or pixel
sh2=read_byte(SH2)
x1, y1=split_two(sh1)
x2, y2=split_two(sh2)
$s1 = x1 \oplus x2$
$s2 = y1 \oplus y2$
s = s1||s2
end
```

ALGORITHM 18: XOR IDEAL SECRET RECOVERY SH1SH3

```
Data: Shares SH1 and SH3
Result: Original secret S that is shared
while not at end of the input files do
sh1=read_byte(SH1) // read a byte or pixel
sh3=read_byte(SH3)
x1,y1=split_ two(sh1)
x3,y3=split_two(sh3)
$s2 = x1 \oplus x3$
$s1 = y1 \oplus y3 \oplus s2$
s = s1||s2
end
```

ALGORITHM 19: (2, 3) XOR IDEAL SECRET RECOVERY SH2SH3

```
Data: Shares SH2 and SH3
Result: Original secret S that is shared
while not at end of the input files do
sh2=read_byte(SH2) // read a byte or pixel
sh3=read_byte(SH3)
x2,y2=split_two(sh2)
x3,y3=split_two(sh3)
$s1 = y2 \oplus y3$
$s2 = x2 \oplus x3 \oplus s1$
s = s1||s2
end
```

III. CONCLUSION

The confidentiality, availability and performance requirement of storage system is addressed in this paper. Secret sharing based solutions provides information theoretic security and also provides trust and reliability. We developed simple XOR and number theory based schemes which reduce the computational complexities. The storage requirement can also be reduced if we use scheme where the share size is only half the size of the original secret. The schemes mentioned in this paper are simple and easy to implement when sharing data with third party servers. The cost factor must also be considered. A (3, 2) or a (4, 2) secret sharing scheme is the best choice. The cost factor can also be reduced by using the non ideal XOR based scheme where the share size is reduced to half but the information theoretic security is compromised. A secret vector which indicates the share number that each server stores can be kept secret. A simple substitution or transposition cipher can also be used for additional security as a pre-processing step before sharing the file. The use of these schemes can be further explored in other areas where the threshold required is as specified in the algorithm. We have used these schemes for efficient sharing of secret images also.

REFERENCES

[1] A. Shamir. ”How to share a secret”. *Communications of the ACM,* 22(11):612-613, 1979.

[2] G. R. Blakley et al.” Safeguarding cryptographic keys”. *In Proceedings of the national computer conference*, volume 48, pages 313-317, 1979..

[3] Y. Desmedt and Y. Frankel. “Shared generation of authenticators and signatures”.*In Advances in CryptologyCRYPTO91,* pages 457-469. Springer,1992

[4] M. Naor and A. Wool. “Access control and signatures via quorum secret sharing.” *Parallel and Distributed Systems, IEEE Transactions on,*9(9):909-922, 1998.

[5] M. Ben-Or, S. Goldwasser, and A. Wigderson. “Completeness theorems for non-cryptographic fault-tolerant distributed computation,” *In Proceedings of the twentieth annual ACM symposium on Theory of computing,* pages 1-10. ACM, 1988.

[6] D. Chaum, C. Crepeau, and I. Damgard. “Multiparty unconditionally secure protocols,” *In Proceedings of the twentieth annual ACM symposium on Theory of computing,* pages 11-19. ACM, 1988.

[7] R. Cramer, I. Damgard, and U. Maurer. “General secure multi-party computation from any linear secret-sharing scheme,” *In Advances in CryptologyEUROCRYPT* 2000, pages 316-334. Springer, 2000.

[8] V. Goyal, O. Pandey, A. Sahai, and B. Waters.”Attribute-based encryptionfor fine-grained access control of encrypted data,” *In Proceedings of the,13th ACM conference on Computer and communications security,* pages 89-98. ACM, 2006.

[9] J. Bethencourt, A. Sahai, and B. Waters.”Ciphertext-policy attribute based encryption,” *In Security and Privacy*, 2007. SP’07. IEEE Symposium on, pages 321-334. IEEE, 2007

[10] T. Tassa.” Generalized oblivious transfer by secret sharing”. *Designs,*
Codes and Cryptography, 58(1):11-21, 2011

[11] B. Shankar, K. Srinathan, and C. P. Rangan. “Alternative protocols for
generalized oblivious transfer,” *In Distributed Computing and Networking*, pages 304-309. Springer, 2008

[12] M. Naor and A. Shamir. “Visual cryptography,” *In Advances in Cryptology EUROCRYPT* 94,pages 1-12. Springer, 1995.

[13] Krawczyk, Hugo. ”Secret sharing made short.” *Advances in CryptologyCRYPTO*93. Springer Berlin Heidelberg, 1994.

[14] Rabin, Michael O. ”Efficient dispersal of information for security, load balancing, and fault tolerance.” *Journal of the ACM (JACM)* 36.2 (1989):335-348.

[15] Parakh, Abhishek, and Subhash Kak. ”Space efficient secret sharing for implicit data security.” *Information Sciences* 181.2 (2011): 335-341.

[16] Gnanaguruparan, Meenakshi, and Subhash Kak. ”Recursive hiding of secrets in visual cryptography.” *Cryptologia* 26.1 (2002): 68-76.

[17] Parakh, Abhishek, and Subhash Kak. ”A tree based recursive information hiding scheme.” *Communications (ICC), 2010 IEEE International Conference on*. IEEE, 2010..

[18] Bguin, Philippe, and Antonella Cresti. ”General short computational secret sharing schemes.” *Advances in CryptologyEUROCRYPT95*. Springer Berlin Heidelberg, 1995.

[19] Rogaway, Phillip, and Mihir Bellare. ”Robust computational secret sharing and a unified account of classical secret-sharing goals.”, *Proceedings of the 14th ACM conference on Computer and communications security,ACM* 2007.

[20] Vinod, V., et al. ”On the power of computational secret sharing.” *Progress in Cryptology-INDOCRYPT* 2003. Springer Berlin Heidelberg, 2003 162-176.

[21] Wang, Daoshun, et al. ”Two secret sharing schemes based on Boolean operations*.” Pattern Recognition* 40.10 (2007): 2776-2785

[22] Kurihara, Jun, et al. ”A new (k, n)-threshold secret sharing scheme and
its extension.” *Information Security. Springer Berlin Heidelberg*, 2008. 455-470.

[23] Kurihara, Jun, et al. "A Fast (3, n)-Threshold Secret Sharing Scheme Using Exclusive-OR Operations." *IEICE transactions on fundamentals of electronics, communications and computer sciences* 91.1 (2008): 127-138.

[24] Lv, Chunli, et al. "Efficient Ideal Threshold Secret Sharing Schemes Based on EXCLUSIVE-OR Operations." *Network and System Security (NSS), 2010 4th International Conference on. IEEE*, 2010.

[25] M. Mignotte. “How to share a secret”. *In Cryptography*, pages 371-375. Springer, 1983.

[26] C. Asmuth and J. Bloom. “A modular approach to key safeguarding.” *Information Theory, IEEE Transactions on*, 29(2):208-210,1983.

[27] Iftene, Sorin. "General secret sharing based on the chinese remainder theorem with applications in e-voting." *Electronic Notes in Theoretical Computer Science* 186 (2007): 67-84.