International Journal of Computer Applications (0975 – 8887)
Volume 26– No.8, July 2011

Shared Cryptography with Embedded Session Key for Secret Audio

Prabir Kr. Naskar, Hari Narayan Khan, Ujjal Roy, Ayan Chaudhuri , Atal Chaudhuri

Department of Computer Science & Engineering
Jadavpur University
Kolkata 700032, West Bengal, India

ABSTRACT

In today's scenario many audio/voice files are needed to be transmitted over internet for various important purposes. So protection of these files is an important issue. Efficient cryptographic methods are there to protect data but every thing depends on the protection of encryption key. That leads to single point failure. To overcome this drawback shared cryptography becomes more popular. Here we are suggesting a novel secret sharing scheme which employs simple graphical masking method, performed by simple ANDing for share generation and reconstruction can be done by performing simple ORing the qualified set of legitimate shares. Not only that, the generated shares are compressed and each share contains partial secret information, that leads to added protection to the secret and reduced bandwidth requirement for transmission.

General Terms

Shared cryptography, security.

Keywords

Threshold Cryptography, Audio sharing, Compression, Perfect Secret Sharing (PSS).

1. INTRODUCTION

The effective and secure protections of sensitive information are primary concern in commercial, research and military systems. Today secret audio is transferred over internet for various commercial purposes. So it is also important to ensure data is not being tampered.

Imagine that a group of scientists came up with important information which is recorded in an audio file, which must be kept secret. Therefore this file is encrypted and the key is stored in some place which is hopefully safe. The file is so important for the scientists that they can not stand loosing it. For this reason they make several copies of the encryption key and store them in different locations. But this would increase the risk of the key being stolen or compromised. The better solution would be to split the key into multiple pieces and store them in different locations and be able to reconstruct the key from association of all or part of those pieces i.e. if some of the pieces even be lost or stolen, then also the key can be recovered but individual piece cannot do any help to get the idea of the key.

Again there lies the possibility of the original encrypted secret audio be corrupted. So we propose the ciphered audio be also split into pieces. Thus we propose a scheme where both the key and the ciphered audio together are split into pieces and the original is recovered from a subset of legitimate pieces. Obviously this allows additional security for both the secret as well as the session key.

This is basically a variant of threshold cryptography, which deals with sharing sensitive secret among a group of n users so that only when a sufficient number k ($k \leq n$) of them come together, the secret can be reconstructed. Well known secret sharing schemes (SSS) in the literature include Shamir[1] based on polynomial interpolation, Blakley[2] based on hyper plane geometry and Asmuth-Bloom[3] based on Chinese Remainder theorem.

Shamir's[1] scheme based on a polynomial of degree (k-1) to any set of k points that lie on the polynomial. The method is to create a polynomial of degree (k-1) as follows-

$$f(x) = d_0 + d_1x^1 + d_2x^2 + ... + d_{k-1}x^{k-1} (\text{mod}\, p)$$

Where d0 is the secret and p is a prime number and the remaining coefficients picked at random. Next find n points on the curve and give one to each of the participants. When at least k out of the n participants revel their points, there is sufficient information to fit an (k-1)th degree polynomial and then the secret value d0 can be easily obtained by using Lagrange Interpolation.

Blakley[2] used geometry to solve the secret sharing problem. The secret message is a point in a k-dimensional space and n shares are affine hyperplanes that intersect in this point. The set solution $x = (x_1, x_2, x_3, ..., x_k)$ to an equation $a_1x_1 + a_2x_2 + ... + a_kx_k = b$ forms an affine hyperplane. The secret, the intersection point, is obtained by finding the intersection of any k of these planes.

Asmuth-Bloom's[3] scheme in which reduction modulo operation is used for share generation and the secret is recovered by essentially solving the system of congruence using Chinese Remainder Theorem (CRT).

Above all secret sharing schemes are regarded as a Perfect Secret Sharing (PSS) scheme because coalition of (k-1) shares doesn't expose any information about the secret. A shortcoming of above secret sharing schemes is the need to reveal the secret shares during the reconstruction phase. The system would be more secure if the subject function can be computed without revealing the secret shares or reconstructing the secret back.

This is known as function sharing problem where the function's computation is distributed according to underlying SSS such that distributed parts of computation are carried out by individual user and then the partial results can be combined to yield the final result without disclosing the individual secrets. Various function sharing protocols are been proposed [4], [5], [6], [7], [8], [9], [10] mostly based on Shamir's secret sharing as the underlying scheme. Some work [11] is also available on Blakley's secret sharing scheme, and Asmuth-Bloom scheme[12] as well.

In all of above secret sharing schemes, each share hold the complete secret information in encrypted or ciphered form. We have suggested a different concept, where simple graphical masking (ANDing) technique is used for shared generation and all the shares contain partial secret information and reconstruction is done by simply ORing the predefined minimal set of shares.

The success of the scheme depends upon the mask generation, a step wise algorithm is suggested for such mask design for any (k, n) scheme where n numbers of masks are designed to generate n different shares and any k shares on ORing reconstruct the original secret. Here we have further proposed an unique compression technique on the shares as a solution towards bandwidth requirement.

2. SECRET SHARING ALGORITHM

The proposed work is based upon a novel secret sharing scheme which employs simple graphical masking method using simple ANDing for share generation and reconstruction can be done by simple ORing the predefined minimal number of shares.

2.1 Concept

For better understanding let us consider any secret as a binary bit file (i.e. bit is the smallest unit to work upon, in actual implementation one can consider a byte or group of bytes or group of pixels as the working unit). The secret could be an image, an audio or text etc. We shall decompose the bit file of any size onto n shares in such a way that the original bit file can be reconstructed only ORing any k number of shares where $k \leq n \geq 2$ but in practice we should consider $2 \leq k < n \geq 3$.
Our basic idea is based on the fact that every share should have some bits missing and those missing bits will be replenished by exactly (k-1) other shares but not less than that. So every individual bit will be missed from exactly (k-1) shares and must be present in all remaining (n-k+1) shares, thus the bit under consideration is available in any set of k shares but not guaranteed in less than k shares. Now for a group of bits, for a particular bit position, (k-1) number of shares should have the bit missed and (n-k+1) number of shares should have the bit present and similarly for different positions there should be different combinations of (k-1) shares having the bits missed and (n-k+1) number of shares having the bits present. Clearly for every bit position there should be ${}^{n}C_{k-1}$ such combinations and in our scheme thus forms the mask of size ${}^{n}C_{k-1,}$ which will be repeatedly ANDed over the secret in any regular order. Different masks will produce different shares from the secret. Thus 0 on the mask will eliminate the bit from the secret and 1 in the mask will retain the bit forming one share. Different masks having different 1 and 0 distributions will thus generate different shares.

Next just ORing any k number of shares we get the secret back but individual share having random numbers of 1's & 0's reflect no idea about the secret. As an example a possible set of masks for 5 shares with threshold of 3 shares is shown below:

Share 1:	1 1 1 1 1 1 0 0 0 0
Share 2:	1 1 1 0 0 0 1 1 1 0
Share 3:	1 0 0 1 1 0 1 1 0 1
Share 4:	0 1 0 1 0 1 1 0 1 1
Share 5:	0 0 1 0 1 1 0 1 1 1

One can easily check that ORing any three or more shares we get all 1's but with less than three shares some positions still have 0's i.e. remain missing.

2.2 Algorithm

Here we are presenting the algorithm for designing the masks for n shares with threshold k.

Step-1: List all row vectors of size n having the combination of (k-1) numbers of 0's and (n-k+1) numbers of 1's and arrange them in the form of a matrix. Obvious dimension of the matrix will be ${}^{n}C_{k-1} \times n$.

Step-2: Transpose the matrix generated in Step-1. Obvious dimension of the transposed matrix will be $n \times {}^{n}C_{k-1.}$ Each row of this matrix will be the individual mask for n different shares. The size of each mask is ${}^{n}C_{k-1}$ bits, i.e. the size of the mask varies with the value of n and k.

Pseudo Code for mask generation:

Input: n, k

Output: mask[n][ ] and length of mask pattern (say len)

```
int mask_generator(n, k, mask[n][])
{
    bin[][n] : integer array
    len = 0; //initialization
    max_val = 2^n – 1; //calculate decimal value of
    //n numbers of 1's
    for i=max_val-1  to 0
            decimal_to_binary(i, bin[len][n]);
            //calculate binary equivalent of decimal i and
             //store in bin[][] array
             if(zero_check(bin[len][n], k))
                len++;
            //check whether (k-1) nos. of zero exist or
            //not, if true then increment len by 1
     end for
     transpose(mask, bin);
     //take transpose matrix of bin[][n] and store in
     //mask[n][]
    return len;
}
```

Let us consider the previous example where n=5 and k=3.

Step-1: List of row vectors of size 5 bits with 2 numbers of 0's and 3 numbers of 1's.

$$\begin{matrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{matrix}$$

Dimension of the matrix is ${}^5C_2 \times 5$ i.e. 10×5

Step-2: Take the transpose of the above matrix and we get the desired masks for five shares as listed above in the form of matrix of dimension $5 \times {}^5C_2$ i.e. 5×10. There are five masks each of size 10 bits.

3. AUDIO SHARING PROTOCOL

Here we are presenting stepwise protocol for our audio secret sharing scheme. In our scheme we share both secret data and key. Therefore every share has two parts, secret share and header share.

3.1 Sharing Phase:

Step-1: First construct Header Structure of five fields and put share number (S) in 1st field, total number (n) of shares in 2nd filed, threshold number (k) in 3rd filed, key ($\mathcal{K}$) in 4th filed, and the size of secret audio in bytes (B) in 5th filed.

1-byte	1-byte	1-byte	16-bytes	4-bytes
Share number [S]	Total number of Shares [n]	Threshold [k]	Encryption Key [$\mathcal{K}$]	Size of Secret File in Bytes [B]

Figure-1. Header Structure

Step-2: Generates n masks for n individual shares using the proposed mask generation algorithm for n and k.

Step-3: Generate 16-byte digest from the session key ($\mathcal{K}$) defined for encrypting the secret audio.

[Share Generation]

Step-4: Now select a mask and apply logical AND (byte in the secret audio corresponding to bit one of the mask will be retained and that corresponding to bit zero of the mask will be set to zero) repeatedly with the secret audio and the zero byte in the generated share corresponding to zero bit of the mask be discarded, this generates one compressed secret share.

Step-5: Then the 1st retained byte (P_1) will be ciphered by the 1st digest byte (Q_1) by the following operation:

$$R_1 = (P_1 \times Q_1) \bmod 251 \ldots\ldots\ldots\ldots\ldots\ldots (i)$$

And 2nd retained byte will be ciphered by the 2nd digest byte Q_2.

[Header Share]

Step-6: Now the header [Figure-1] excluding the leftmost field is also shared by applying logical bit wise ANDing with individual mask.

Step-7: Next each header share is appended with the share number (S) in the first field and concatenated with the corresponding secret share, which forms one complete share for transmission.

3.2.Reconstruction Phase:

[Header Reconstruction]

Step-1: First collect k or more number of shares and for each share separate the header share and secret share parts.

Step-2: Next from k or more header shares separate out the share number (S) and OR the remaining portions of all headers which reconstructs total number of shares n, threshold k, encryption key $\mathcal{K}$ and original size of secret file B.

[Secret Reconstruction]

Step-3: Once the original Header is reconstructed, we extract the Key ($\mathcal{K}$), and using $\mathcal{K}$ we generate same 16-byte digest string.

Step-4: Now using n and k, extracted from reconstructed Header structure, generate n masks (the masks used in sharing phase) using our mask generation algorithm.

Step-5: According to the share number of the share holder appropriate mask is used to expand the secret share part by inserting zero bytes corresponding to zero bit in the corresponding mask.

Step-6: Ciphered bytes (R_1) corresponding to 1-bit position in the mask, have generated by the equation-(i). So here we apply following operation to get original byte (P_1).

$$P_1 = (R_1 \times M_1^{-1}) \bmod 251 \ldots\ldots\ldots\ldots\ldots\ldots (ii)$$

Where M_1^{-1} is the multiplicative inverse of Q_1.

Step-7: Next k numbers of expanded secret shares are ORed to reconstruct the original secret. Here we extract the original size of the secret file (B) from Header structure, by which we can easily reconstruct the lossless original secret.

4. ANALYSIS OF COMPRESSION

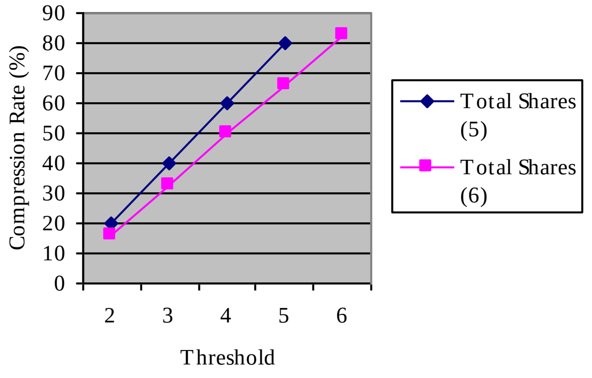
All masking pattern has equal number of zeros with different distribution only. In every share we collapse all zero bytes corresponding to zero bit in the corresponding mask. It may be noted that as k is closer to n, more is compression i.e. maximum for k = n.

Next for lossless expanding, knowing n and k we can redesign all n masks using our original mask generation algorithm. According to the share number of the share holder appropriate mask is used to expand the secret share by inserting zero bytes corresponding to zero bit in the corresponding mask.

In our example of (3, 5) the mask size is of 10 bits and every mask has 4 zeroes, thus every secret can be compressed by approximately 40%, obviously the compression varies with (k, n). (In case of an example of (5, 6) the mask size is 15 bits and every mask has 10 zeroes, thus compression will be 66.6%).

Threshold (k)	Length of Masking Pattern	Number of zero in masking pattern	Approximate Compression Rate (%)
Total number of Shares (n) = 5			
2	5	1	20
3	10	4	40
4	10	6	60
5	5	4	80
Total number of Shares (n) = 6			
2	6	1	16
3	15	5	33
4	20	10	50
5	15	10	66
6	6	5	83

Table-1. Shows compression rate for different (k, n)



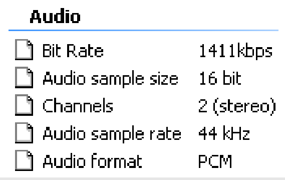
Graph-1. Threshold vs. compression

5. STRENGTH OF THE PROTOCOL

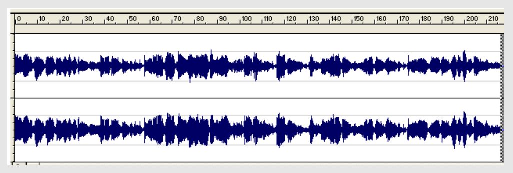
Here we use an audio file as a secret. But our proposed scheme is equally applicable for any binary file such as Image (.bmp), Text etc.

In our scheme if and only if numbers of collating shares are equal to k or more, then only the original secret audio is reconstructed; otherwise reconstructed audio will be completely noisy. Because fewer shares can not reconstruct the original header, thus we can not have either right key ($\mathcal{K}$) or the information to construct the correct masking pattern. So our proposed scheme can claim to be a Perfect Secret Sharing (PSS) Scheme.

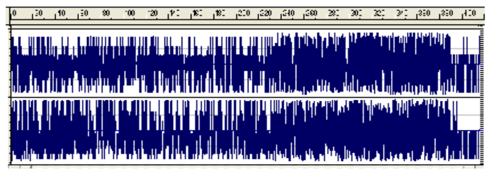
Here all generated shares are compressed and contain partial secret information in encrypted form. That not only provides additional protection to the secret file but also reduces the bandwidth requirement for transmission. Only when legitimate group of shares come together, then only the original secret is reconstructed.


Audio

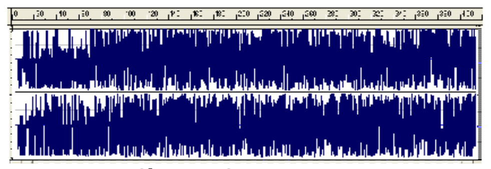
Bit Rate	1411kbps
Audio sample size	16 bit
Channels	2 (stereo)
Audio sample rate	44 kHz
Audio format	PCM



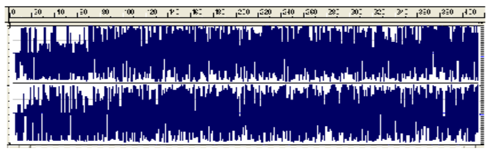
Original Secret Audio (.wav), size= 2.42 MB



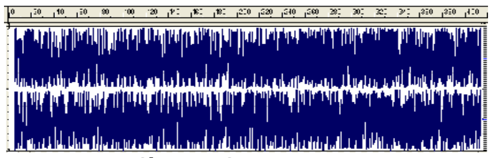
Share-1, Size=1.45MB



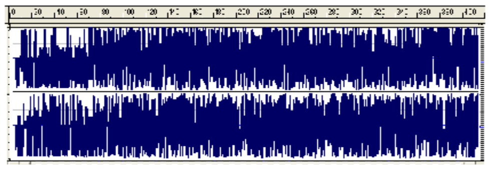
Share-2, Size=1.45MB



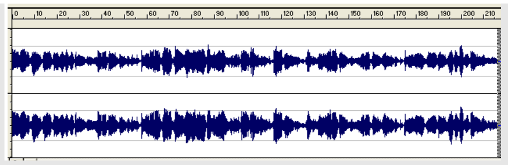
Share-3, Size=1.45MB



Share-4, Size=1.45MB



Share-5, Size=1.45MB


Reconstructed Secret Audio using any 3 or more shares, size= 2.42 MB

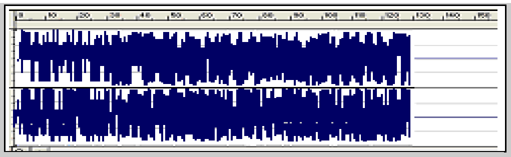

Reconstructed completely noisy Audio using share-1 and share-4. size= 2.04 MB

Figure-2. Secret audio, five shared audios, reconstructed secret audio and reconstructed noisy audio (courtesy Sound Forge).

6. CONCLUSION

We present a novel secret sharing approach which is PSS and generates compressed noisy shares. Like other group of researchers the shares can be sent in some cover medium. The cover medium may be Image, Audio, and Video. Therefore noisy shares become meaningful shares that protect from attackers eyes. However, all of these methods need cover file (i.e. the meaningful shares) size bigger than the secret but in our case cover image of same size as the secret is good enough. Our future effort will try to reduce the size of the cover image further i.e. cover image size may be lesser than the secret.

7. ACKNOWLEDGMENTS

We are thankful to the department of Computer Science & Engineering of Jadavpur University, Kolkata, for giving us the platform for planning and developing this work in departmental laboratories.

8. REFERENCES

[1] A. Shamir: “How to share a secret ?” Comm ACM, 22(11):612-613, 1979.

[2] G. Blakley : “Safeguarding cryptographic keys “ Proc. of AFIPS National Computer Conference, 1979.

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

[4] Y. Desmedt “Some recent research aspects of threshold cryptography” Proc of ISW’97 1st International Information Security Workshop vol.1196 of LNCS paper 158-173 Springer-Verlag 1997.

[5] Y. Desmedt and Y. Frankel “Threshold cryptosystems” Proc of CRYPTO’89 volume 435 of LNCS, paper 307-315 Springer Verlag 1990.

[6] Y. Desmedt and Y. Frankel “Shared generation of authenticators and signatures” Proc. of CRYPTO’91 volume 576 of LNCS pages 457-469 Springer Verlag 1992.

[7] Y. Desmedt and Y. Frankel “Homomorphic zero knowledge threshold schemes over any finite abelian group” SIAM journal on Discrete Mathematics 7(4): 667-675, 1994.

[8] H. F. Hua ng and C.C. Chang “A novel efficient (t, n) threshold proxy signature scheme” Information Sciences 176(10): 1338-1349, 2006.

[9] A. De Santis, Y. Desmedt,Y. Frankel and M. Yung “How to share a function securely ?” In proc of STOC 94, paper 522-533, 1994.

[10] V. Shoup “Practical threshold signatures” In Proc of Eurocrypt 2000. volume 1807 of LNCS paper 207-220, Springer-Verlag 2000.

[11] Bozkurt, Kaya, Selcuk, Guloglu “Threshold Cryptography Based on Blakely Secret Sharing” Information Sciences.

[12] K. Kaya and A. A. Selcuk “Threshold Cryptograhy based on Asmuth-Bloom Secret Sharing” Information Sciences 177(19) 4148-4160, 2007