Space Efficient Secret Sharing: A Recursive Approach

Abhishek Parakh and Subhash Kak

Abstract

This paper presents a k-threshold secret sharing technique that distributes a secret S into shares of size $\frac{|S|}{k-1}$, where $|S|$ denotes the secret size. This bound is close to the optimal bound of $\frac{|S|}{k}$, if the secret is to be recovered from k shares. The proposed scheme makes use of repeated polynomial interpolation. Our technique has potential applications in secure and reliable storage of information on the Web and in sensor networks.

I. Introduction

Information theoretically secure secret sharing schemes are space inefficient because a k-out-of-n secret sharing technique generates n shares each of same size as that of the secret itself, leading to a n-fold increase in required storage.

In order to improve space efficiency, computational secret sharing techniques have been developed [1], [2], [3], [4] in which a symmetric key is used to encrypt the original secret and the encrypted secret is divided into pieces to which redundancy is added by the use of block error correction techniques [5], [6], [7]. The encryption key is split into shares using information theoretically secure methods of secret sharing. This leads to an n-fold increase in key size, shares of which have to be stored with every piece of the encrypted secret, hence incurring a burdensome overhead [7]. In order to recover the secret in presence of wrong shares, robustness was added using hash-fuctions in [8] and more general implementations of computational secret sharing appeared in [9], [10], [11], [12].

An efficient method for sharing multiple secrets with security based on assumption of hardness of discrete logarithm problem is presented in [13]. Whereas [14] proposes a scheme based on systematic

A. Parakh and S. Kak are with the Computer Science Department, Oklahoma State University, Stillwater, OK, 74078 USA e-mail: (see http://cs.okstate.edu/˜parakh and http://cs.okstate.edu/˜subhashk).

block codes and [15], [16] propose schemes based on Shamir's secret sharing scheme but require a large amount of side information to be stored as public information and futher [14], [15], [16], [17] attempt to maintain ideal security. Other schemes [18], [19] focus at improving efficiency of computations involved in share creation and secret reconstruction rather than space efficiency.

Here we propose a secret sharing scheme that encodes a secret S into shares of size $\frac{|S|}{k-1}$, where $|S|$ is the size of the secret, without the use of any encryption key. This is close to the optimal bound of $\frac{|S|}{k}$, if the secret is to be recovered from k shares (please see [7]). The proposal is based on recursion, in which a secret is first divided into $k-1$ pieces and then the pieces are encoded one by one in such a manner that the shares of the already encoded pieces are reused to create new shares for the next piece.

We present an example from an earlier paper [20] that proposed a 2-out-of-2 ($k = 2$ and $n = 2$) secret sharing scheme for binary secrets, based on recursion. Although the implementation in [20] is quite different from the one proposed in this paper, it is useful in understanding the notion of recursion in secret sharing. Suppose we wish to share a secret of size 7 bits, $S = 1011011$, into 2 shares such that both of them are needed to reconstruct the secret. A simple implementation would use exclusive-OR transformation to divide the secret into shares. Hence, using conventional scheme, we would create two shares, say $D_{S1} = 1001101$ and $D_{S2} = 0010110$ of size 7 bits each. However, using recursion we can create shares smaller than 7 bits each as is shown in table 1. To execute a recursive algorithm, we first divide the 7 bit secret S into 3 pieces, s_1, s_2, and s_3 of size 1, 2 and 4 bits respectively. A concatenation of the pieces $s_1 s_2 s_3$ is equal to S. In table 1, the bits that are in bold are the bits taken from the shares created in the previous step. Only the final shares 0010 and 1001 are shared between players.

The final shares 0010 and 1001 contain the complete secret S. To recover the secret, the players reconstruct the smaller pieces by properly aligning the shares and using exclusive-OR; the concatenation of these pieces is the secret.

In the above example we have achieved a significant improvement in share sizes from 7 bits each in a conventional scheme to 4 bits each in a recursive scheme. However, the resulting shares have a lower security compared to conventional exclusive-OR based implementation because each party now only needs to determine 4 bits of the other share to recreate the complete secret on its own. Nevertheless, for a file, say of size $2^{11} - 1$ bits (about 2Kb), that must be divided into shares, each share would be of size 2^{10} (1Kb) which may provide adequate security for many applications.

Although the example is a good representative to explain recursion in secret sharing, it, nonetheless, has the following limitations. Firstly, it does not provide a threshold scheme. Secondly, secrets that are encoded using the above method may only be of size $2^t - 1$ for some positive integer t; this is due to

Secret pieces	Shares				
s_1: 1	0				$D_{s_{11}}$
	1				$D_{s_{12}}$
s_2: 01	**0**	0			$D_{s_{21}}$
	0	**1**			$D_{s_{22}}$
s_3: 1011	**0**	**0**	1	0	$D_{s_{31}}$
	1	0	**0**	**1**	$D_{s_{32}}$

TABLE I

ILLUSTRATION OF RECURSIVE SECRET SHARING. A LARGE SECRET $S = 1011011$ IS DIVIDED INTO THREE CONSECUTIVE PIECES AND THEN THE SHARES ARE RECURSIVELY ENCODED. ONLY THE FINAL SHARES 0010 AND 1001 NEED BE SHARED BETWEEN PLAYERS.

the binary tree structure of the scheme (table 1). Thirdly, it does not provide a mechanism to control the security of the resulting shares, i.e. if $2^t - 1$ bits are encoded, the resulting shares are always of size 2^{t-1}.

It is to be noted that a threshold recursive scheme for 2-out-of-n secret sharing is discussed in [21].

In this paper we present a general k-out-of-n threshold recursive secret sharing scheme and relax the size restrictions such that to share a secret S requires shares only of size $\frac{|S|}{k-1}$, which is near optimal. Further, we provide a mechanism to control the size of the resulting shares ranging from $\frac{|S|}{k-1}$ to $|S|$ thus controlling the final security level achieved by the shares (where shares of size $|S|$ provide maximum security).

The proposed recursive secret sharing scheme has applications in distributed online storage of information discussed in [5], [6]. However, Rabin [5] discarded the possibility of using secret sharing schemes because of the n-fold increase in storage space required by conventional implementations of secret sharing techniques. Since our scheme provides a near optimal way to encode data into small shares, it becomes an ideal candidate for use in secure online data storage. Such secure data storage scheme is an example of implicit data security [22], *implicit* in that there is no explicit encryption of data and no encryption keys are used. Data is so divided that each piece is implicitly secure in itself and reveals complete information only when k or more of the pieces are brought together. $k-1$ pieces or less do not reveal any information (in computational sense) [7]. Systems implementing such distributed data storage have appeared at CMU

and IBM [23], [24], [25], [26], [27], [2].

II. SPACE EFFICIENT SECRET SHARING

The proposed scheme recursively builds upon polynomial interpolation and sampling similar to [28]. However, Shamir's scheme generates shares of size $|S|$ for a secret S. By contrast, we generate shares of size $\frac{|S|}{k-1}$.

We first briefly review the scheme presented by Shamir [28] (Algorithm 1).

`Algorithm 1(Shamir's secret sharing scheme)`

1) Choose a prime p, $p > max(S, n)$, where $S \in \mathbb{Z}_p$ is the secret.
2) Choose $k-1$ random numbers a_1, a_2, ..., a_{k-1}, uniformly and independently, from the field $\mathbb{Z}_p$.
3) Using a_i, $1 \leq i \leq (k-1)$ and secret S, generate polynomial $p(x)$ of degree $k-1$, $p(x) = S + a_1 x + a_2 x^2 + ... + a_{k-1} x^{k-1}$ (mod p).
4) Sample $p(x)$ at n points $D_i = p(i)$, $1 \leq i \leq n$ such that the shares are given by (i, D_i).

Reconstruction of the secret is performed by interpolating any k points (shares) and evaluating $S = p(0)$.

In our implementation, we first divide secret S into $k-1$ pieces. The division of a secret file F is discussed by Rabin in [5] with the aim of information dispersal. For example, to share a secret of size $|S| = 30$ bits, we divide it into say 3 pieces of 10 bits each. In general, we divide the secret into pieces of size $\lceil \frac{|S|}{k-1} \rceil$. We assume the use of similar technique to create the pieces. These pieces will be denoted by s_i, $1 \leq i \leq (k-1)$, such that their concatenation $s_1 s_2 ... s_{k-1} = S$.

For the proposed scheme assume a finite field $\mathbb{Z}_p$, where p is a prime and $p > max(s_{max}, n)$, where $s_{max} = max(s_i)$, $1 \leq i \leq (k-1)$, and s_1, s_2, ..., s_{k-1} are the pieces of S. The shares will be denoted as $D_{s_i 1}$, $D_{s_i 2}$, ...,$D_{s_i m}$ at the intermediate stages, where $2 \leq m \leq k-2$ and D_1, D_2, ..., D_n at the final stage. (Note that, as in Shamir's scheme, $D_{s_i m}$ are the y-coordinates only, while the respective x-coordinates m, are tacitly known to all players.)

It is to be noted that since for all practical cases $n < \frac{|S|}{k}$, the stated share size $\frac{|S|}{k}$ is the optimal size of a share for a k-out-of-n secret sharing scheme. But strictly speaking if $n > \frac{|S|}{k}$, then the shares would be of size $|n|$. In this paper we will ignore the latter case and assume $n < \frac{|S|}{k}$.

Formally, a *space optimal* (k, n) *secret sharing scheme* is a secret sharing scheme that, for a secret S, produces shares of size $\frac{|S|}{k}$. And a *space efficient secret sharing scheme* is a secret sharing scheme that

approaches the optimal factor of $\frac{|S|}{k}$ in terms of share size.

The proposed scheme works as follows: We randomly and uniformly choose a number $a_1 \in \mathbb{Z}_p$ and generate 1^{st} degree polynomial $p_1(x) = a_1x + s_1$. Then we sample $p_1(x)$ at two points $D_{s_11} = p_1(1)$ and $D_{s_12} = p_1(2)$, to generate two shares for s_1. This first step can be viewed as a direct execution of Shamir's (2, 2) secret sharing scheme. Next we use these two shares of s_1 to generate polynomial $p_2(x) = D_{s_12}x^2 + D_{s_11}x + s_2$, where the coefficients are the previous two shares and the free term is the new piece. Sampling $p_2(x)$ at three points $D_{s_21} = p_2(1)$, $D_{s_22} = p_2(2)$, and $D_{s_23} = p_2(3)$, generates three shares of s_2. We can now delete D_{s_11} and D_{s_11} because the new shares D_{s_21}, D_{s_22}, and D_{s_23} have the shares of s_1 hidden within themselves. We then use the shares of s_2 to create a 3^{rd} degree polynomial $P_3(x)$ with s_3 as its free term, and generate shares for s_3 by sampling the newly created polynomial at 4 points. These four points, D_{s_31}, D_{s_32}, D_{s_33}, and D_{s_34} have the shares of s_1, s_2 as well as s_3 and therefore D_{s_21}, D_{s_22} and D_{s_23} can now be deleted. The process is repeated for pieces s_4, s_5, ..., s_{k-1} by creating $p_4(x)$, $p_5(x)$, ...,$p_{k-2}(x)$ and repetitive sampling and reusing of shares and deleting the older shares. At the last step, we generate a polynomial $p_{k-1}(x) = D_{s_{k-2}(k-1)}x^{k-1} + D_{s_{k-2}(k-2)}x^{k-2} + ... + D_{s_{k-2}1}x + s_{k-1}$ and sample it at n points $D_1 = p_{k-1}(1)$, $D_2 = p_{k-1}(2)$, ..., $D_n = p_{k-1}(n)$, such that the final shares are given by (i, D_i), $1 \leq i \leq n$. These final n shares have recursively hidden $k-1$ secrets within themselves. Algorithm 2 illustrates the process.

Algorithm 2 - Dealing Phase

1) Choose a prime p, $p > max(s_{max}, n)$, where where $s_{max} = max(s_i)$, $1 \leq i \leq (k-1)$, and s_1, s_2, ..., s_{k-1} are the pieces of secret S.
2) Randomly and uniformly choose a number $a_1 \in \mathbb{Z}_p$ and generate polynomial $p_1(x) = a_1x + s_1$.
3) Sample $p_1(x)$ at two points $D_{s_11} = p_1(1)$ and $D_{s_12} = p_1(2)$, which represent two shares of s_1.
4) Do for $2 \leq i \leq (k-1)$
   a) Generate polynomial,
      $p_i(x) = D_{s_{i-1}i}x^i + D_{s_{i-1}(i-1)}x^{i-1} + ... + D_{s_{i-1}1}x + s_i$.
   b) Sample $p_i(x)$ to create new shares,
      i) If $i < k-1$, sample at $i+1$ points:
         $D_{s_i1} = p_i(1)$
         $D_{s_i2} = p_i(2)$
         $\vdots$

$D_{s_i(i+1)} = p_i(i+1)$.

ii) If $i = k-1$, sample at n points:

$D_1 = p_i(1)$

$D_2 = p_i(2)$

$\vdots$

$D_n = p_i(n)$.

c) Delete old shares: $D_{s_{i-1}1}$, $D_{s_{i-1}2}$, ..., $D_{s_{i-1}i}$.

5) The final n shares are given by (i, D_i), $1 \leq i \leq n$.

`Algorithm 2 - Reconstruction Phase`

1) Interpolate any k shares (i, D_i) to generate the polynomial of degree $k-1$,
$p_{k-1}(x) = D_{s_{k-2}(k-1)}x^{k-1} + D_{s_{k-2}(k-2)}x^{k-2} + ... + D_{s_{k-2}1}x + s_{k-1}$
and evaluate $s_{k-1} = p_{k-1}(0)$.

2) Do for all $i = k-2$ down to 1

a) Interpolate $i+1$ shares given by $(m+1, D_{s_i(m+1)})$, $0 \leq m \leq i$ obtained from coefficients of $p_{i+1}(x)$ to generate polynomial of degree i,
$p_i(x) = D_{s_{i-1}i}x^i + D_{s_{i-1}(i-1)}x^{i-1} + ... + D_{s_{i-1}1}x + s_i$.

b) Evaluate $s_i = p_i(0)$.

As seen above the reconstruction of pieces is straightforward and it proceeds in a last-in, first-out manner. Any k of the players can interpolate the polynomial of degree $k-1$ such that the free term represents $s_{k-1} = p_{k-1}(0)$. Then using the $k-1$ coefficients of this polynomial as points (leaving out the free term which is s_{k-1}), interpolate the polynomial of degree $k-2$ to obtain s_{k-2}. This process is repeated until we obtain s_1.

Algorithm 2 clearly generates shares of size $|s_i|$. Since $|s_i| = \frac{|S|}{k-1}$, for all $i = 1$ to $k-1$, we have achieved shares of size $\frac{|S|}{k-1}$.

Security of the protocol: The security of the proposed protocol is predicated on the choice of the first coefficient a_1. Given that a_1 is randomly and uniformly chosen, steps 1-3 of Algorithm 2, generate two shares such that both of them are required for the reconstruction of s_1. Steps 1-3 are similar to Shamir's (2,2) secret sharing scheme. Therefore, we can assume that given any number $r \in \mathbb{Z}_p$, $Pr(r = D_{s_1 1}) = Pr(r = D_{s_1 2}) = \frac{1}{p}$. These two shares are then used as random coefficients to generate a polynomial of

third degree with the next piece s_2 as the free term and repeat the process of sampling. Thereon, we use the shares that are generated in the present step to generate new shares at the next step. Therefore, the final resulting pieces are random if a_1 is chosen randomly with a uniform probability distribution from the field.

For a secret message or file of size 3KB, with $k = 4$ and $n = 7$, we would create $k - 1 = 3$ pieces of size 1KB ($= 8 \cdot 10^3 bits$) each and choose a prime $p > 8 \cdot 10^3 bits$. Therefore, when $k - 1$ players collude, the resulting security is on the order of $\frac{1}{p}$ or $\frac{1}{2^{10^3}}$. This implies that without the knowledge of the k^{th} share, $k - 1$ players can guess the k^{th} share correctly only with a probability of $\frac{1}{2^{10^3}}$.

Example. Assume that we have a secret $S = 17280512$ that is broken into 4 pieces $s_1 = 17$, $s_2 = 28$, $s_3 = 05$, and $s_4 = 12$. Here the pieces are created with a decimal base, i.e. number of decimal digits in the pieces (2 digits each). S is to be shared between 7 players such that any 5 of them can reconstruct all the 4 secrets. We can now use a prime $p = 31$. (If one were to use a conventional method for secret sharing, he would have to choose a large prime $p > 17280512$, which clearly will yield pieces of that order; however here $p = 31$ suffices, yielding smaller pieces.)

Dealing phase.

1) Randomly and uniformly choose a number $a_1 \in \mathbb{Z}_p$. Let $a_1 = 22$. Generate polynomial, $p_1(x) = a_1 x + s_1 = 22x + 17$ (mod 31).
2) Sample $p_1(x)$ at two points to generate two shares of piece s_1, i.e. $D_{s_1 1} = p_1(1) = 8$ and $D_{s_1 2} = p_1(2) = 30$.
3) Generate polynomial $p_2(x) = D_{s_1 2} x^2 + D_{s_1 1} x + s_2 = 30x^2 + 8x + 28$.
4) Sample $p_2(x)$ at 3 points to generate three shares of s_2, i.e. $D_{s_2 1} = p_2(1) = 4$, $D_{s_2 2} = p_2(2) = 9$, and $D_{s_2 3} = p_2(3) = 12$.
5) Delete $D_{s_1 1}$ and $D_{s_1 2}$.
6) Generate polynomial $p_3(x) = D_{s_2 3} x^3 + D_{s_2 2} x^2 + D_{s_2 1} x + s_3 = 12x^3 + 9x^2 + 4x + 5$.
7) Sample $p_3(x)$ at 4 points to generate four shares of s_3, i.e. $D_{s_3 1} = p_3(1) = 30$, $D_{s_3 2} = p_3(2) = 21$, $D_{s_3 3} = p_3(3) = 19$, and $D_{s_3 4} = p_3(4) = 3$.
8) Delete $D_{s_2 1}$, $D_{s_2 2}$, and $D_{s_2 3}$.
9) Generate polynomial
   $p_4(x) = D_{s_3 4} x^4 + D_{s_3 3} x^3 + D_{s_3 2} x^2 + D_{s_3 1} x + s_4 = 3x^4 + 19x^3 + 21x^2 + 30x + 12$.
10) Sample $p_4(x)$ at 7 points, which represents the final 7 shares. Hence, $D_1 = p_4(1) = 23$, $D_2 =$

$p_4(2) = 15$, $D_3 = p_4(3) = 24$, $D_4 = p_4(4) = 3$, $D_5 = p_4(5) = 8$, $D_6 = p_4(6) = 12$, and $D_7 = p_4(7) = 29$.

11) Delete $D_{s_3 1}$, $D_{s_3 2}$, $D_{s_3 3}$, and $D_{s_3 4}$.

The final seven shares are given by $(1, D_1) = (1, 23)$; $(2, D_2) = (2, 15)$; $(3, D_3) = (3, 24)$; $(4, D_4) = (4, 3)$; $(5, D_5) = (5, 8)$; $(6, D_6) = (6, 12)$; and $(7, D_7) = (7, 29)$.

Reconstruction phase.

All the four pieces can be reconstructed using any 5 out of 7 final shares.

Using 5 shares, say $(1, 23)$, $(3, 24)$, $(4, 3)$, $(5, 8)$, and $(7, 29)$, we can interpolate the 4^{th} degree polynomial $p_4(x) = 3x^4 + 19x^3 + 21x^2 + 30x + 12$ (mod 31), thus retrieving piece s_4 (the free term of the polynomial) by evaluating $s_4 = p_4(0)$.

Then extracting the coefficients of $p_4(x)$ and using them as y-coordinates of points x=1, 2, 3, and 4, i.e. $(1, 30)$, $(2, 21)$, $(3, 19)$, and $(4, 3)$ we can regenerate the 3^{rd} degree polynomial $p_3(x) = 12x^3 + 9x^2 + 4x + 5$ by interpolation and retrieve the s_3 as the free term, $s_3 = p_3(0)$.

The coefficients of $p_3(x)$ are then used as points $(1, 4)$, $(2, 9)$, $(3, 13)$ to interpolate $p_2(x) = 30x^2 + 8x + 28$ and reconstruct $s_2 = p_2(0)$.

The coefficients of $p_2(x)$ are used as $(1, 8)$ and $(2, 30)$ to interpolate $p_1(x) = 22x + 17$ and reconstruct $s_1 = p_1(0)$.

The algorithm simulates a Last In First Out (LIFO) data structure.

III. Controlling the security

It is clear from the construction provided in the previous section that we are trading security for share size. Information theoretic security is achieved when the secret is not divided into pieces and Shamir's scheme is directly applied, creating shares of size as large as the complete secret itself. Consequently, least security is achieved when the shares are $\frac{1}{k}^{th}$ the size the secret, which however may be an adequate level of security for large secrets and certain applications. The proposed algorithm achieves a factor of $\frac{1}{k-1}$ for the size of the share.

Instead of dividing the secret into $k - 1$ pieces we may divide the secret into $k - 2$ pieces, or $k - 3$ pieces, ..., or no pieces at all depending on desired level of security. Hence, if in general, we denote by m the number of pieces into which the secret is divided then replacing all occurrences of $k - 1$ with m, the algorithm proceed as follows:

1) Choose a prime p, $p > max(s_{max}, n)$, where $s_{max} = max(s_i)$, $1 \leq i \leq m$, and s_1, s_2, ..., s_m are the pieces of secret S.
2) Randomly and uniformly choose $k - m$ numbers $a_i \in \mathbb{Z}_p$ and generate polynomial $p_1(x) = a_{k-m}x^{k-m} + a_{k-m-1}x^{k-m-1} + \ldots + a_1x + s_1$.
3) Sample $p_1(x)$ at $k-m+1$ points $D_{s_1 1} = p_1(1)$, ..., $D_{s_1(k-m+1)} = p_1(k-m+1)$, which represent $k - m + 1$ shares of s_1.
4) If $m \geq (k - m + 1)$
   a) $j = k - m + 1$
   b) Do for $2 \leq i \leq m$
      i) Generate polynomial,
      $p_i(x) = D_{s_{i-1}j}x^j + D_{s_{i-1}(j-1)}x^{j-1} + ... + D_{s_{i-1}1}x + s_i$.
      ii) Sample $p_i(x)$ to create new shares,
         A) If $i < m$, sample at $j + 1$ points:
         $D_{s_i 1} = p_i(1)$
         $D_{s_i 2} = p_i(2)$
         $\vdots$
         $D_{s_i(j+1)} = p_i(j + 1)$
         B) If $i = m$, sample at n points:
         $D_1 = p_i(1)$
         $D_2 = p_i(2)$
         $\vdots$
         $D_n = p_i(n)$.
      iii) Delete old shares: $D_{s_{i-1}1}$, $D_{s_{i-1}2}$, ..., $D_{s_{i-1}j}$.
      iv) $j = j + 1$
5) The final n shares are explicitly given by (i, D_i), $1 \leq i \leq n$.

The reconstruction phase works in a manner similar to that presented in Algorithm 2 and the details are omitted here.

The resulting security, when $k - 1$ players collude, in the general case, can be written as $\frac{1}{p}$ (which is the probability of correctly guessing the k^{th} share with only the knowledge of $k - 1$ shares) where the size of p is on the order of $|\frac{S}{m}|$. Here, m may therefore be called a *security factor*.

IV. CONCLUSIONS

We have presented a recursive scheme that generates shares of size $\frac{|S|}{k-1}$ for any secret S. The scheme is general and it places no restriction on the secret size. The results are close to the optimal factor of $\frac{|S|}{k}$ and represent significant improvement over conventional secret sharing schemes that generate shares of size $|S|$. Further, we have not used any key based encryption to achieve the reduction in share sizes.

A general case, when the shares are of size $\frac{|S|}{m}$, m being the security factor varying from 1 to $k-1$, is presented. The security upon collusion of $k-1$ players is $\frac{1}{p}$ where size of p depends on m.

The proposed scheme will have applications in secure distributed storage of information on the Web and in sensor networks and in secure parallel transmission of data.

REFERENCES

[1] B. Schneier, *Schneier's Cryptography Classics Library: Applied Cryptography, Secrets and Lies, and Practical Cryptography*. Wiley, 2007.

[2] P. Rogaway and M. Bellare, "Robust computational secret sharing and a unified account of classical secret-sharing goals," in *CCS '07: Proceedings of the 14th ACM Conference on Computer and Communications Security*. New York, NY, USA: ACM, 2007, pp. 172–184.

[3] V. Vinod, A. Narayanan, K. Srinathan, C. P. Rangan, and K. Kim, "On the power of computational secret sharing," *Indocrypt 2003*, vol. 2904, pp. 265–293, 2003.

[4] A. Cresti, "General short computational secret sharing schemes," in *Advances in Cryptology EUROCRYPT 95, volume 921 of Lecture Notes in Computer Science*. Springer, 1995, pp. 194–208.

[5] M. O. Rabin, "Efficient dispersal of information for security, load balancing and fault tolerance," *Journal of the ACM*, vol. 36, no. 2, pp. 335–348, 1989.

[6] J. Garay, R. Gennaro, C. Jutla, and T. Rabin, "Secure distributed storage and retrieval," *Theoretical Computer Science*, pp. 275–289, 1997.

[7] H. Krawczyk, "Secret sharing made short," *Proceedings of the 13th Annual International Cryptology Conference on Advances in Cryptology*, pp. 136–146, 1994.

[8] ——, "Distributed fingerprints and secure information dispersal," in *Twelfth Annual ACM Symposium on Principles of Distributed Computing (PODC 1993)*. ACM Press, 1993, pp. 207–218.

[9] P. Beguin and A. Cresti, "General short computational secret sharing schemes," in *Advances in Cryptology EUROCRYPT 95, LNCS vol. 921*. Springer, 1995, pp. 194–208.

[10] C. Cachin, "On-line secret sharing," in *IMA Conference on Cryptography and Coding, LNCS vol. 1025*. Springer, 1995, pp. 190–198.

[11] A. Mayer and M. Yung, "Generalized secret sharing and group-key distribution using short keys," in *Compression and Complexity of Sequences 1997*. IEEE Press, 1997, pp. 30–44.

[12] V. Vinod, A. Narayanan, K. Srinathan, C. Rangan, and K. Kim, "On the power of computational secret sharing," in *Progress in Cryptology INDOCRYPT 2003, LNCS vol. 2904*. Springer,, 2003, pp. 162–176.

[13] L. Harn, "Efficient sharing (broadcasting) of multiple secrets," *IEE Proceedings - Computers and Digital Techniques*, vol. 142, no. 3, pp. 237–240, May 1995.

[14] H.-Y. Chien, J.-K. Jan, and Y.-M. Tseng, "A practical (t,n) multi-secret sharing scheme," *IEICE transactions on fundamentals of electronics, communications and computer sciences*, vol. 83, no. 12, pp. 2762–2765, 2000.

[15] L.-J. Pang and Y.-M. Wang, "A new (t,n) multi-secret sharing scheme based on shamir's secret sharing," *Applied Mathematics and Computation*, vol. 167, no. 2, pp. 840 – 848, 2005.

[16] 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, pp. 483 – 490, 2004.

[17] C.-W. Chan and C.-C. Chang, "A scheme for threshold multi-secret sharing," *Applied Mathematics and Computation*, vol. 166, no. 1, pp. 1 – 14, 2005.

[18] M. Liu, L. Xiao, and Z. Zhang, "Linear multi-secret sharing schemes based on multi-party computation," *Finite Fields and Their Applications*, vol. 12, no. 4, pp. 704 – 713, 2006, special Issue Celebrating Prof. Zhe-Xian Wan's 80th Birthday.

[19] M. H. Dehkordi and S. Mashhadi, "New efficient and practical verifiable multi-secret sharing schemes," *Information Sciences*, vol. 178, no. 9, pp. 2262 – 2274, 2008.

[20] M. Gnanaguruparan and S. Kak, "Recursive hiding of secrets in visual cryptography," *Cryptologia*, vol. 26, pp. 68–76, 2002.

[21] A. Parakh and S. Kak, "A recursive threshold visual cryptography scheme," *Cryptology ePrint Archive, Report 535*, 2008.

[22] ——, "Online data storage using implicit security," *Information Sciences*, vol. In Press, Corrected Proof, pp. –, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/B6V0C-4WF4J67-1/2/432f2904ee2b98d825db0f5b83761021

[23] G. R. Ganger, P. K. Khosla, M. Bakkaloglu, M. W. Bigrigg, G. R. Goodson, G. R, V. Pandurangan, S. Oguz, V. P, C. A. N. Soules, J. D. Strunk, and J. J. Wylie, "Survivable storage systems," in *In DARPA Information Survivability Conference and Exposition, IEEE*. IEEE Computer Society, 2001, pp. 184–195.

[24] A. Iyengar, R. Cahn, J. A. Garay, and C. Jutla, "Design and implementation of a secure distributed data repository," in *In Proc. of the 14th IFIP Internat. Information Security Conf*, 1998, pp. 123–135.

[25] S. Lakshmanan, M. Ahamad, and H. Venkateswaran, "Responsive security for stored data," *International Conference on Distributed Computing Systems*, vol. 0, p. 146, 2003.

[26] A. P. Sameer, A. Paul, S. Adhikari, and U. Ramach, "Design of a secure and fault tolerant environment for distributed storage," 2004.

[27] M. Waldman, A. D. Rubin, and L. F. Cranor, "The architecture of robust publishing systems," *ACM Trans. Internet Technol.*, vol. 1, no. 2, pp. 199–230, 2001.

[28] A. Shamir, "How to share a secret," *Communications of ACM*, vol. 22, no. 11, pp. 612–613, 1979.