A Construction of Practical Secret Sharing Schemes using Linear Block Codes

Michael Bertilsson and Ingemar Ingemarsson

Department of Electrical Engineering

Linköping University, 581 83 Linköping, Sweden

Abstract. In this paper we address the problem of constructing secret sharing schemes for general access structures. The construction is inspired by linear block codes. Already in the beginning of the eighties constructions of threshold schemes using linear block codes were presented in [6] and [7]. In this paper we generalize those results to construct secret sharing schemes for arbitrary access structure. We also present a solution to the problem of retrieving the secret.

1 Introduction

Secret sharing is a method of dividing a secret into shares that can be distributed among a set of participants. The shares are chosen in such a way that only some predefined subsets of participants can retrieve the secret. The idea of secret sharing was first presented independently in [2][3]. Since then several papers has been written on the subject, see [4] for a list of references.

We will use linear codes to do the construction of our secret sharing schemes. Linear error correcting block codes of length n, dimension k and alphabet q is a k-dimensional sub space of the n-dimensional space. For a thorough discussion of linear codes see [1]. Linear codes has been used earlier in some constructions of threshold schemes [6][7].

In [8] a construction using matroids is presented. Our construction can be seen as special case of that one, actually they show that most of the other known constructions can be seen as special cases of their construction. However that construction has drawbacks from a practical point of view. Assume that we have N participants and a secret of size q. They need to store a matrix of size $\sim q^N \times N$ with entries of size q, even if the matrix is public and only have to be stored in one place it requires far to much storage. Especially since q has to be large to give a secure system.

In our construction we only need a generator matrix of size $\sim N \times N$ with entries of size q. Note that the size of the matrix is independent of q. This matrix can be made public. We also give an algorithm that constructs our generator matrix for a given access structure. Finally, given the shares from a set of participants who should be able to retrieve the secret we show how to retrieve the secret. Furthermore, our construction will always give perfect schemes and in many cases the schemes will be ideal.

The basic properties of secret sharing schemes and the notation used are introduced in section 2. Our construction and the algorithm constructing the generator matrix is presented in section 3. Finally in section 4 we show how to retrieve the secret.

2 Notation and Basic results

2.1 Secret Sharing

Let $P = \{P_1, P_2, \ldots, P_N\}$ be the set of all participants. Then 2^P is the set of all subsets. The size of 2^P is 2^N.

The set of all subsets of users that can recover the secret will be called the *access structure* and will be denoted F, $F \subseteq 2^P$. The set of all subsets that can not recover the secret will be denoted $\bar{F}$, $\bar{F} \subseteq 2^P$.

Let us look at the following situation. We have a group of users that can recover the secret. What happens if we add more users to this group? It seams reasonable that this new group should still be able to recover the secret. This property means that F is a monotone set which can be written formally as follows

$$A, B \subseteq 2^P, A \in F, A \subseteq B \Rightarrow B \in F \quad (2.1)$$

For small sets P we can use F or $\bar{F}$ to describe the access structure. But the size of 2^P grows exponentially with the size of P. This means that in most cases both F and $\bar{F}$ will be large and it will be difficult to describe the access structure using neither F nor $\bar{F}$. Fortunately we can use the fact that F is monotone to give an easier description of the access structure. Since F is monotone it is described by a unique minimal subset in the following sense. A set A is *minimal* if the following condition is satisfied.

$$A, B \subseteq 2^P, A \in F, B \subseteq A \Rightarrow B \notin F \quad (2.2)$$

We denote the set of these sets Γ and use it to give a short description of F. Instead of writing Γ as a set, we will write it as a sum of products.

$$\Gamma = \sum_i \gamma_i \quad (2.3)$$

Each term γ_i is a product of P_i:s representing a minimal subset in F. In a similar way $\bar{F}$ can be represented by its maximal subsets denoted $\bar{\Gamma}$.

From now on we will refer to the access structure using either F, $\bar{F}$, Γ, $\bar{\Gamma}$. When we want to make it explicit that we refer to $\bar{F}$ or $\bar{\Gamma}$ we will call them the negative access structure.

We can regard (2.3) as a logical expression. By this we mean that + represents "or" and juxtaposition represents "and". Then we can read $\Gamma = AB + AC + AD$ as, (user A and B) or (user A and C) or (user A and D). This means that we can use relations and concepts from logic. Let us now introduce the dual of Γ.

Definition 1

The dual of an access structure Γ is constructed by replacing each sum in Γ by a product and each product by a sum. The dual is denoted by F^* or Γ^*.

As in the case of Γ we write the dual access structure as a sum of products

$$\Gamma^* = \sum_i \gamma_i^* . \tag{2.4}$$

A dual has the property that the dual of a dual is the primal, i.e. $\Gamma^{**} = \Gamma$. It should also be pointed out that an access structure can be the dual of itself, i.e. $\Gamma = \Gamma^*$.

Usually we describe an access structure using Γ and we would like an easy way of getting $\bar{\Gamma}$ from Γ. The dual can help us with this. The following theorem is a minor rewriting of some results from [5] and we state it without proof.

Theorem 1

Given an access structure Γ we can get $\bar{\Gamma}$ using the dual in the following way. For each term γ_i^* in Γ^* we get a term $\bar{\gamma}_i$ in $\bar{\Gamma}$ as

$$\bar{\gamma}_i = P \backslash \gamma_i^* \tag{2.5}$$

Finally we remind the reader of the following two definitions.

Definition 2

A secret sharing scheme is called *perfect* if each group of users not in F has the same knowledge about the secret.

Definition 3

The information *rate* of a secret sharing scheme is defined as the ratio between the size of the secret and the size of the shares. A perfect scheme with information rate one is called *ideal*.

3 The Construction

We assume that we have a secret that we want to distribute among a set of participants. To do this we have to make the following construction. The construction is divided into two steps. First we use the access structure to construct a generator matrix, G, over $GF(q)$. Then we use the secret to construct the information vector. We choose the information vector so the first component of the codeword will be the equal to the secret. Multiplying the generator matrix and the information vector we get a codeword. The components of the codeword are distributed as shares to the participants. The first component of the codeword can not be distributed to any participant since it is equal to the secret.

Definition 4

Denote a codeword $c = (c_0, c_i, \ldots, c_{n-1})$. The share of user P_i is the set of components $\{c_j\}$ for all j in the index set p_i. Let A be a subset of participants then

$$J_A = \bigcup_{P_i \in A} p_i \qquad \textbf{(3.1)}$$

J_A is a union of index sets and is of course itself an index set.

The following theorem tells us how we should choose the columns of the generator matrix. We remind you that the secret is equal to c_0 and is thus associated with the index set $\{0\}$.

Theorem 2

A subset of users, A, can retrieve the secret, i.e. $A \in F$, if and only if the index sets J_A and $\{0\}$ fulfills the following requirement

$$rank(G(J_A \cup \{0\})) = rank(G(J_A)) \text{ for } A \in F. \qquad \textbf{(3.2)}$$

A subset of users, B, that can not retrieve the secret, i.e. $B \in \overline{F}$, must fulfill the following requirement

$$rank(G(J_B \cup \{0\})) = rank(G(J_B)) + 1 \text{ for } B \in \overline{F}. \qquad \textbf{(3.3)}$$

Proof

Equation (3.2) means that $G(\{0\})$ is linearly dependent of $G(J_A)$ and from Lemma 1 we know that this means that knowing $c(J_A)$ determines c_0 and we

2.2 Linear Codes

We use the following standard notation. Let G be the $k \times n$ generator matrix, H the $(n-k) \times n$ parity check matrix. G is a linear mapping from the information space U to the code space C. Both U and C are over $GF(q)$. As distance measure we use Hamming distance, denoted d. Hamming distance is defined as the number of positions in which two words differ. The weight, w, of a word is the number of non zero symbols in the word. d_{min} for a code is the minimum distance between any two code words. The minimum distance is equal to the weight of the code words with least weight.

H and G have the following relationship

$$GH^T = 0. \tag{2.6}$$

The parity check matrix also generates a code which is called the dual code. Let $c^{\perp}$ be a codeword in the dual code then

$$Gc^{\perp^T} = \bar{0} \tag{2.7}$$

for all codewords in the dual code. The same is also true for the parity check matrix and all codewords in the code.

We also need the following notation. Let A be an index set with values between 0 and $n-1$. $G(A)$ is a subset of the columns in G corresponding to A and $c(A)$ is a subset of the components in c corresponding to A.

We are interested in what happens when we add new columns from the generator matrix to our set of columns. Let A be an index set and let m be an index not in A. Let g be the column of G corresponding to index m. If $c_i(A) = c_j(A)$, what is the value of $c_i(m)$ and $c_j(m)$? There are two different cases to be studied, either g is linearly dependent of the columns in $G(A)$, or it is linearly independent. The following two lemmas are given without proof.

Lemma 1

Let g be linearly dependent of $G(A)$. If $c_i(A) = c_j(A)$ then $c_i(m) = c_j(m)$.

Lemma 2

Let g be linearly independent of $G(A)$. Choose a fix c_j. Then in the set $\{c_i, c_i(A) = c_j(A)\}$ there are exactly

$$q^{k - rank(G(A)) - 1} \tag{2.8}$$

codewords with $c_i(m) = a$ for every $a \in [0, ..., q-1]$.

know the secret. Equation (3.3) means that $G(\{0\})$ is linearly independent of $G(J_B)$ and from Lemma 2 we know that this means that for all possible values in $c(J_B)$ the value in c_0 takes on all values in $GF(q)$ with equal probability and we know nothing about the secret.

We have now chosen G and are ready to choose the information vector, u, using the secret, s. s must be an element of $GF(q)$. We choose the components of u such that

$$ug_o = s, \tag{3.4}$$

where g_0 is the first column in the generator matrix. This means that the first component of the codeword will be equal to the secret.

Now we have constructed both the generator matrix and the information vector. We multiply the information vector and the generator matrix to get a codeword.

$$c = uG \tag{3.5}$$

We can know distribute the different components of this codeword as shares to the users. The sets p_i determine which user should have which components.

Just as a remark, if G generates the access structure Γ, then the parity check matrix H of G generates the dual access structure Γ^* of Γ.

A few comments about this construction to make it clearer. If we are able to use only one column per user we get a $k \times (N+1)$ generator matrix and the system will be ideal. This is not possible for all access structures. It is important to notice that the shares of a subset $A \in F$ does not necessarily determine a single codeword. The shares determines a set of codewords that are equal in the positions $J_A \cup \{0\}$.

3.1 The Algorithm Constructing the Generator Matrix

In Theorem 2 we gave the requirements for the construction of the generator matrix. We will use this to give an algorithm that generates a generator matrix for a given access structure. Theorem 2 gives us relationships between the subsets of F and $\bar{F}$ and the column subsets of G. As stated in section 2 it is usually impractical to work with F and $\bar{F}$. Instead we would like to get conditions on G from Γ and $\bar{\Gamma}$. This is easily done. If (3.2) is true for an index set, J_A, it is also true if we add indices to J_A. So if we find a G that fulfills (3.2) for Γ, it will also fulfill it for F. In a similar way it is enough to use $\bar{\Gamma}$ instead of $\bar{F}$ in (3.3).

The algorithm is divided into two parts. First we use $\bar{\Gamma}$ to construct a generator matrix with some free parameters and then we use Γ to decide the values of these parameters.

Equation (3.3) says that if $B \in \bar{\Gamma}$, the first column of G is linearly independent of $G(J_B)$. This means that the null-space of $G(J_B)$ is q times larger then the null-space of $G(J_B \cup \{0\})$ and this can be written as

$$\exists u \text{ such that } uG(J_B) = \bar{0} \text{ and } uG(0) \neq 0. \qquad \textbf{(3.6)}$$

If there are q^l codewords with zeros in the positions corresponding to J_B, q^{l-1} are equal to a in the first position for all $a \in GF(q)$.

From equation (3.8) and the comment following it we know that for each subset $B \in \bar{\Gamma}$ there should be at least one codeword with zeros in the positions corresponding to J_B and a non-zero symbol in the first position. (We actually get many such codewords.) One way to get such a codeword is to let it be a row in the generator matrix. This means that we will choose the entries of the generator matrix as follows.

Step 1 First we let the number of rows in G be equal to the number of terms in $\bar{\Gamma}$ and demand that all the entries in the first column of G are non-zero. Now we choose the rest of the entries in G using the terms in $\bar{\Gamma}$. Each term in $\bar{\Gamma}$ is associated with a row in G. Let g_i be the i:th row of G and let $g_i(J_B)$ be the index positions corresponding to the participants in B. Then we let $g_i(J_{\bar{\gamma}_i}) = 0$ and let the rest of the coefficients of G be undetermined, but non-zero.

Now we have a G with some unknown entries. This will be used in the next step where we will determine these unknown. Equation (3.2) says that if $A \in \Gamma$, the first column of G is linearly dependent of $G(J_A)$. This means that the null-space of $G(J_A)$ is equal to the null-space of $G(J_A \cup \{0\})$ and every codeword which is zero in the positions corresponding to J_A must also be zero in the first position. From this we can choose the values of the undecided g_{ij} as follows.

Step 2 Given the generator matrix constructed in the first step we will use Γ to choose the values of the undetermined coefficients of G. The following equation gives us demands on the undetermined coefficients of G.

$$\forall u, \text{ such that } uG(J_{\gamma_i}) = \bar{0}, uG(\{0\}) = 0 \qquad \textbf{(3.7)}$$

3.2 Examples

We give three examples that are typical in that they show the problems that arise in the above algorithm. In the following examples we have chosen q small. Since q is the size of the secret q must in practice be made large. It should be large enough to make it improbable to be able to guess the value of the secret.

Example 3.1

Let $P = \{A, B, C\}$ and $\Gamma = A + BC$ then $\bar{\Gamma} = B + C$. From $\bar{\Gamma}$ we get $k = 2$ and

$$G = \begin{bmatrix} s_1 & a_1 & 0 & c_1 \\ s_2 & a_2 & b_2 & 0 \end{bmatrix} \tag{3.8}$$

Now we can use Γ. From $\gamma_1 = A$ and the first column we get

$$u_1 a_1 + u_2 a_2 = 0 \text{ and } u_1 s_1 + u_2 s_2 = 0. \tag{3.9}$$

Combining these we get

$$\left(s_1 - s_2 \frac{a_1}{a_2}\right) u_1 = 0 \Rightarrow s_1 - s_2 \frac{a_1}{a_2} = 0 \tag{3.10}$$

since u_1 could be non-zero. From $\gamma_2 = BC$ we get

$$u_2 b_2 = 0 \text{ and } u_1 c_1 = 0 \Rightarrow u_1 = u_2 = 0 \tag{3.11}$$

and we can choose b_2 and c_1 arbitrarily. We do not get any new demands on s_1 and s_2 either. In this case we can choose $q = 2$ and $s_1 = s_2 = a_1 = a_2 = b_2 = c_1 = 1$. We then get the following generator matrix

$$G = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}. \tag{3.12}$$

Example 3.2

Let $P = \{A, B, C\}$ and $\Gamma = AB + AC + BC$ then $\bar{\Gamma} = A + B + C$. From $\bar{\Gamma}$ we get $k = 3$ and

$$G = \begin{bmatrix} s_1 & 0 & b_1 & c_1 \\ s_2 & a_2 & 0 & c_2 \\ s_3 & a_3 & b_3 & 0 \end{bmatrix}. \tag{3.13}$$

We will not go through all the steps in this example. We just state that Γ gives us

$$-s_1\frac{b_3}{b_1}-s_2\frac{a_3}{a_2}+s_3 = 0 \tag{3.14}$$

$$-s_1\frac{c_2}{c_1}+s_2-s_3\frac{a_2}{a_3} = 0 \tag{3.15}$$

$$s_1-s_2\frac{c_1}{c_2}-s_3\frac{b_1}{b_3} = 0. \tag{3.16}$$

We now have three equations and nine unknowns. We start by choosing $s_i = 1$ and from equation (3.16) we see that we have to choose $a_2 \neq a_3$, otherwise $b_3 = 0$. This means that $q>2$. In a similar way $b_1 \neq b_3$ and $c_1 \neq c_2$. If we now choose $a_2 = b_1 = c_1 = 1$ and $a_3 = 2$ we get $b_3 = -1$ and $c_2 = 1-2^{-1}$. Finally we choose $q = 3$ and get

$$G = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 2 \\ 1 & 2 & 2 & 0 \end{bmatrix}. \tag{3.17}$$

If we study this generator matrix we see that the last row is a linear combination of the first and the second row. This means that we can skip the last row and still get the same code. We can now write the generator matrix as

$$G = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 2 \end{bmatrix}. \tag{3.18}$$

Example 3.3

Let $P = \{A, B, C, D\}$ and $\Gamma = AB+BC+CD$ then $\bar{\Gamma} = AC+AD+BD$. From $\bar{\Gamma}$ we get $k = 3$ and

$$G = \begin{bmatrix} s_1 & 0 & b_1 & 0 & d_1 \\ s_2 & 0 & b_2 & c_2 & 0 \\ s_3 & a_3 & 0 & c_3 & 0 \end{bmatrix}. \tag{3.19}$$

From Γ we get

$$\gamma_1 = AB \Rightarrow s_2 = s_1\frac{b_2}{b_1} \tag{3.20}$$

$$\gamma_2 = BC \Rightarrow -s_1\frac{b_2}{b_1} + s_2 - s_3\frac{c_2}{c_3} = 0 \tag{3.21}$$

$$\gamma_3 = CD \Rightarrow s_2 = s_3\frac{c_2}{c_3} \tag{3.22}$$

Combining these three equations gives

$$-s_2 + s_2 - s_2 = -s_2 = 0. \tag{3.23}$$

This means that $s_2 = 0$, but we have a demand that $s_i \neq 0$. The only solution to the three equations is not allowed. We can not construct a generator matrix with one column per user for this access structure. This means that we in our model can not create an ideal secret sharing scheme for the access structure $\Gamma = AB + BC + CD$. To find a solution to this access structure we must add another column to one of the participants. We choose to add a column to participant B and then $\bar{\Gamma}$ gives us the following generator matrix

$$G = \begin{bmatrix} s_1 & 0 & b_{11} & b_{12} & 0 & d_1 \\ s_2 & 0 & b_{21} & b_{22} & c_2 & 0 \\ s_3 & a_3 & 0 & 0 & c_3 & 0 \end{bmatrix}. \tag{3.24}$$

We must choose the b_{ij} so that the two columns are linearly independent otherwise it is useless to have two columns. This means that the only u that is orthogonal to the A column and the two B columns is $u = (0, 0, 0)$. We do not get any restrictions on the b_{ij} and a_i from $\gamma_1 = AB$. Neither does $\gamma_2 = BC$ give any restrictions on the b_{ij} and c_i. The only restrictions left are equation (3.24) and that the columns of B must be independent. For $q = 3$ we can choose

$$G = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 2 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}. \tag{3.25}$$

In the first example we get the generator matrix directly. In the second we can construct the generator matrix but some of the rows are linearly dependent and we can remove them from the generator matrix. In the last example we are not able to construct the generator matrix with only one column per participant. Instead we are forced to add extra column(s) to one or more of the participants. These three examples cover all possible cases.

4. Retrieving the Secret

We will regard the retrieval of the secret as a special case of erasure decoding of linear block codes. It is important to notice that the number of erased positions is usually more than $d_{min} - 1$ which means that we will not be able to determine all erased symbols. Due to the given construction we are always able to determine the secret, i.e. the first component of the codeword.

Let us look at the shares as a part of a received codeword that is erased in some of the positions. Use the received vector, v, and the error vector, e, as follows

$$v = (0, 0, a_i, \ldots) \tag{4.1}$$

$$e = (c_0, c_1, 0, \ldots). \tag{4.2}$$

v is zero in all positions where there are not any known shares and has values according to the shares in the others. e is zero in all positions corresponding to the known shares and is assumed to be equal to the codeword in the others. Let B denote the index set of the unknown shares. Observe especially that index 0 is in B. We can now set up the following equations

$$s = Hv^T = H(c+e)^T = He^T = H(B)e(B)^T, \tag{4.3}$$

since $Hc^T = 0$ by definition. Since both v and H are known s is also known. $H(B)$ is in general a rectangular matrix with more columns than rows. To be able to reconstruct *all* the unknown c_i, which are in $e(B)$, we have to find the pseudo-inverse of $H(B)$. I.e. find

$$H(B)^{-1}H(B) = I_{|B|}, \tag{4.4}$$

where $I_{|B|}$ is the identity matrix of size $|B|$. If we could find $H(B)^{-1}$ we could get the missing values in e from s in the following way

$$e(B)^T = H(B)^{-1}s. \tag{4.5}$$

The problem is that we can not always find a $H(B)^{-1}$ fulfilling the demands in equation (4.4). However we are in a better position in that we only have to find c_0, and it can be found if we are able to find the first row of $H(B)^{-1}$. Denote this first row of $H(B)^{-1}$ l then we have

$$c_0 = e_0 = -\sum_{i=0}^{k-1} l_i s_i. \tag{4.6}$$

l has the following properties, $lh_0 = 1$ and $lh_i = 0, i \in B$, where h_i are the columns of H.

This means that l is not orthogonal to the first column of $H(B)$, but is orthogonal to the rest of the columns. We can find such an l if and only if h_0 is linearly independent of the rest of the h_i, $i \in B$. To prove that this is true in our construction we first state the following lemma about G and H.

Lemma 3

Let A be an index set with values from the set $\{0, \ldots, n-1\}$ and let $\backslash A$ be all indices not in A. Then

$$rank(G(A)) = |A| \Rightarrow rank(H(\backslash A)) = n-k \qquad \textbf{(4.7)}$$

$$rank(G(A)) < |A| \Rightarrow rank(H(\backslash A)) < n-k \qquad \textbf{(4.8)}$$

Proof

Remember that $Gc^{\perp^T} = \bar{0}, \forall c^{\perp}$ in the dual code. We first prove (4.7). We take any codeword in the dual code such that $c^{\perp}(\backslash A) = \bar{0}$. Then $c^{\perp}(A)$ must be equal to $\bar{0}$ for all these codewords since the columns in $G(A)$ are linearly independent. So if $c^{\perp}(\backslash A) = \bar{0}$ we have the all zero codeword. This means that $uH(\backslash A) = \bar{0}$ if and only if $u = \bar{0}$ and $rank(H(\backslash A)) = n-k$.

Now we prove (4.8). Since the columns in $G(A)$ are linearly dependent there exist codewords in the dual code, such that $c^{\perp}(\backslash A) = \bar{0}$ and $c^{\perp}(A)$ are not all zero. This means that for some vector $u \neq \bar{0}$, $uH(\backslash A) = \bar{0}$ and the rank of $H(\backslash A)$ must be less than the number of rows.

Theorem 3

h_0 is linearly independent of all h_i, $i \in B$.

Proof

Let $\gamma \in \Gamma$, then the following is always true, $rank(G(J_\gamma)) \leq |J_\gamma|$. $rank(G(J_\gamma)) = |J_\gamma|$ follows as a special case so lets assume that $rank(G(J_\gamma)) < |J_\gamma|$. Choose a set $J \subset J_\gamma$ such that $rank(G(J)) = |J| = rank(G(J_\gamma))$. From Lemma 3 and Theorem 2 $rank(H(\backslash J)) = n-k$ and $rank(H(\backslash(J \cup \{0\}))) < n-k$. This means that h_0 is independent of h_i, $i \in \backslash J$. If we add indices to J to get J_γ we will remove indices from $\backslash J$. If h_0 is independent of all columns h_i, $i \in \backslash J$ then it must still be independent when we have removed some of the columns.

5 Conclusions

We have constructed a secret sharing scheme using linear block codes. We show how to choose the columns of a generator matrix so that we get a desired access structure. We have also given an algorithm that use the access structure Γ and the negative access structure $\bar{\Gamma}$ to construct the generator matrix of the code. The size of this generator matrix is only $\sim N \times N$ for N users. Finally we show how to retrieve the secret given the shares of a set of participants that should be able to retrieve the secret.

References

[1] F.J. MacWilliams, N. J. A. Sloane, *The Theory of ErrorCorrecting Codes*, North-Holland, First Edition 1977

[2] A. Shamir, *How to Share a Secret*, Comm. ACM, Vol.22, pp612, Nov 1979

[3] G. R. Blakely, *Safeguarding Cryptographic Keys*, Proc. AFIPS 1979 Natl. Computer Conference, New York, Vol.48, pp 313-317, June 1979

[4] G. Simmons Ed, *Contemporary Cryptology*, IEEE Press, 1992

[5] G. Simmons, W. Jackson, K. Martin, *The Geometry of Shared Secret Schemes*, Bulletin of the Institute of Combinatorics and its Application (ICA) to appear 1991

[6] R. J. McEliece, D. V. Sarwate, *On Sharing Secrets and Reed-Solomon Codes*, Comm. ACM, Vol.24, pp 583-584, Sep 1981

[7] E.D Karnin, J.W. Green, M.E. Hellman, *On Secret Sharing Systems*, IEEE Trans. Information Th., Vol. IT-29, No 1, pp. 35-41, Jan 1983

[8] K.M. Martin, *Discrete Structures in the Theory of Secret Sharing*, Doct. Thesis, Royal Holloway and Bedford New College, University of London, 1991