On Generalized Greedy Algorithms in Broadcast Key Distribution Schemes

J. Wroblewski*, V. Korjik**, M. Ivkov**

* Institute of Mathematics Warsaw University
ul. Banacha 2, 00-097 Warsaw, Poland
E-mail: jakubw@mimuw.edu.pl
** University of Telecommunications,
Moika 61, St. Petersburg, 191186, Russia.
E-mail, valkor@pds.sut.ru.

Abstract. In [1] a key distribution scheme for broadcast encryption based on 2-designs was proposed. The connectivity of this scheme is the maximal number of messages which should be broadcasted by the center to the non-compromised users to generate a common key. This value was upper bounded in [1] with the use of the greedy algorithm. Unfortunately it is a very crude bound. We propose the generalized greedy algorithm and some randomized modifications of it (a hybrid genetic-greedy algorithm among them) to improve an achievable connectivity of broadcast key distribution schemes. We present the results of simulations for different 2-design matrices to prove this property and to find a computation time of the covering when different algorithms are used. We discuss also how close are the schemes based on block designs to optimal broadcast key distribution scheme given some natural parameters.

Key words: key distribution, broadcast encryption, dynamical key management, resilience, connectivity, block design, covering, greedy algorithm, genetic algorithm.

1 Introduction

Let us consider a typical key distribution problem for broadcast encryption. In such situation, a center should broadcast secure transmissions to a set **V** of users. The size ν of this set can be large enough. For the encryption, the center uses a key k_o, which should be known to the users of the system. Later some users may no longer be entitled to the messages (for instance, if their subscription for pay television has expired). Let **C,** $|\mathbf{C}|=\sigma$ be the set of illegitimate users. To continue broadcasting, the center has to replace k_0 by another key k_1 known to the users in **V-C** only. This situation can be iterated, i.e. some additional users can become unauthorized and join set **C.** Then k_1 has to be replaced by k_2, and so on. We assume also that all illegitimate users are known to the center. (It is no wonder if the center verifies subscriptions of users).

It is logical to assume that the only channel available for the transmission of the keys is broadcast channel itself since otherwise the center has to maintain a host of private channels toward each user. Therefore, we believe that during the installation of the system, each user is supplied with a collection of private keys, which can be used to broadcast master keys ki. We allow the center to use any of the private keys of user for this purpose. We call the procedure described above the *dynamical key management.*

There are two extreme approaches to this problem. Under the first approach every user u has his own unique key k_u, which is used by the center to transmit to this user the key k_i that the broadcaster is going to use. In the context of pay television we have the broadcast message block which can be encrypted by some symmetric cipher under running key k_i and the enabling block consisting of the subblocks ki encrypted under all keys ku of legitimate (at this juncture) users. This means that the total length of the enabling block is at least | **V-C** | times the length of the running key k_i. This is clearly ineffective approach in any broadcast environment.

Under the second approach each group of legitimate (at this juncture) users can compute a running key k_i with the use only their private keys distributed by the center at the initial stage. It is obvious that the center can also compute the same key k_i and uses it to encrypt the broadcast message. This approach can be called *zero message scheme* because no enabling block is necessary. In [2] k-secure, t-conference, zero message key distribution schemes, where t is the size of the group of legitimate users, k is the size of the group of illegitimate users were considered. It has been proved in [1] that $\binom{k + t - 1}{t + 1}$ is the lower bound for the number of private keys which should be distributed initially by the center. What is more the scheme based on symmetrical polynomial meets this bound. In our case this scheme requires $\binom{v - 1}{\sigma}$ initial keys to be distributed to each user, that can be impractical. To reduce the number of private keys and messages transmitted by the center, the private keys can be communicated to subsets of users rather than to individual users. Each subset can be reached during a single transmission round (say a single enabling subblock), thus reducing the number of rounds (subblocks into enabling block). Let us call this scheme a key distribution scheme based on *incident structure.* This is a collection **B**, $|\mathbf{B}|=b$, of subsets of **V**. To every $B \in \mathbf{B}$ we associate a key k_b, which is given by the center to all the users contained in B. The key management can be arranged as follows. Let $\mathbf{C} \subset \mathbf{V}$ be a coalition of illegitimate users known to the center. Then all keys k_b for which B contains at least one element from the coalition are compromised and should be removed. The remaining keys should be used in most effective way to communicate a new broadcast key k_i to all users outside the coalition. Any incident structure can be described by a binary incident $v \times b$ matrix **A**, whose element $a_{ij}=1$ if and only if the i-th user has the j-th key, otherwise - it is equal to zero. Thus the key management in terms of an incident matrix have to be the following. We should remove both all rows corresponding to the users of compromised coalition and all columns which are incident with them (i.e. have "1" in the chosen columns). After that we should choose a set of columns belonging to a new matrix, which *covers* this matrix. (This means that all rows of the new matrix have ones at least in one of the columns of chosen set). Then we can use the keys corresponding to the chosen columns to encrypt broadcast key k_i and place the obtained ciphertexts into enabling block. An example of incident matrix for seven users and seven keys is presented below

	k1	k2	k3	k4	k5	k6	k7
u1	1	0	0	0	1	0	1
u2	1	1	0	0	0	1	0
u3	0	1	1	0	0	0	1
u4	1	0	1	1	0	0	0
u5	0	1	0	1	1	0	0
u6	0	0	1	0	1	1	0
u7	0	0	0	1	0	1	1

The key distribution scheme based on an incidence structure is called s-*resilient* if s is the maximum number σ such that the following relation holds

$$\cup B = \mathbf{V}\text{-}C$$
$$B \in \mathbf{B},\ B \cap \mathbf{C}=0,\ |\mathbf{C}|=\sigma$$

Thus, if $|\mathbf{C}|>s$, then key management based on s-resilient incident structure is impossible since some users cannot be reached by enabling blocks. For $|\mathbf{C}|=\sigma \le s$, we are interested in the minimum number of messages (subblocks into enabling block) that the center has to transmit to legitimate users for any coalition **C** of size σ. This value can be called *the connectivity* χ_σ of the broadcast key distribution scheme based on incident structure. In terms of incident matrix the connectivity is the minimal size of covering to this matrix by its columns. In [1] the incidence structures has been considered, which are based on 2-designs - a special case of the more general structure-*balance incomplete block designs* (BIB-design) [3]. Let us recall some of their properties. A 2-(v,k, λ)

design is a finite set **V**, |**V**|=v, and a collection **B** of its k-subsets (blocks) such that every pair of elements of **V** is contained in λ blocks. We assume that $0 \le \lambda \le k \le v-1$. Every element of **V** occurs in one and the same number of blocks r. The following relation should be true for 2-design: bk=vr, λ(v-1)=r(k-1), b≥v, k≤r. If b=v and k=r, the design is called symmetric. The previous relations imply for symmetric designs: $v=1+k(k-1)/\lambda$. A symmetric design with the parameters 2-(k^2-k+1, k,1) is called a *finite projective plane* of order d=k-1**.** It is the most important 2-design for construction of key distribution schemes. We will consider in the sequel this case only. It has been shown in [1] that resilience of broadcast key distribution schemes based on 2-designs with λ=1 is s=r-1=k-1 whereas the connectivity of such schemes given size of the coalition of compromised users σ can be upper bounded as follows

$$\chi_\sigma \le \frac{1}{k-\sigma}(1+\log k)(v-\sigma) \quad \text{for any } \sigma<k \quad (1)$$

Our contribution in this paper may be viewed in the following manner:

- Improve the bound (1) for χ_σ and especially in the cases of large σ, when (1) is too crude (if not trivial).
- Present the algorithm of matrix covering and estimate its complexity, because the greedy algorithm used in [1] to obtain (1) belongs to *hard problems* (See[4] for detail) (We note that the lack of such algorithm depreciates key distribution scheme completely because it cannot be put into practice).

To solve these problems we use some algorithms of covering applied to 2-design incident matrices with parameters λ=1, v=7, 13, 21, 31, 57, 73, 91, 133. The description of these algorithms and the experimental results of computer simulations (the average size of covering, minimal and maximal covering length, a computation time of covering) are given in the section 2. (We take 2-designs because they can be constructed from difference sets (See [1] and [3] for detail)). In the section 3 we discuss an optimality 2-designs as broadcast key distribution schemes. The last section contains some open problems in a topic of the broadcast key distribution schemes.

2 Algorithms of Covering for 2-Design Schemes

Our problem can be formulated in terms of row covering problem: for a given binary matrix, find minimal set of columns which are incident with each row. In general, this is NP-hard problem. In our case, we have a special form of the matrix: this is a BIB matrix with some columns and rows removed; we know for sure, that a covering exists if σ < s. Our goal is to find the shortest one (i.e. with minimal number of columns).

2.1 Weighted greedy algorithm - method 1 (WGA1)

Our algorithm bases on a “greedy” paradigm: in each step achieve as much as possible. A general scheme is as follows:

1. Calculate coefficients (weights) C_1 and C_2 for all columns of the given matrix.
2. Choose a column with the largest value of C_1.
3. If there are more than one such a column, use the least value of C_2 as a secondary criterion. If more than one column has the both parameters equal, choose the first of them.
4. Add a column chosen as above to a set of covering columns.
5. Remove the column from the matrix. Remove all rows which have “1” in that column.
6. Continue from step 1 with the reduced matrix. Stop when all rows are covered.

Because of stopping criterion, the algorithm will always produce a covering.

The first coefficient, $C_1(i)$, depends on a number of “1” in the i-th column:

$$C_1(i) = \sum_{j=0}^{n-1} a_{ij} \quad (2)$$

$$C_2(i) = \sum_{j=0}^{n-1} a_{ij} \sum_{k=0}^{m-1} a_{kj} C_1(k) \qquad (3)$$

where m - number of columns in the matrix.

Now, we choose to our covering a column with the largest value of C_1 - so we are covering many rows - and with possibly small value of C_2 - so we are covering these rows, which are "hard to cover" by other columns, therefore the rest of a covering process should be relatively easy. If more than one column has the same value of C_1 and C_2, the algorithm chooses the first (in particular order) of them.

The experiment was performed as follows: the BIB matrix of size v×v was generated and σ random rows (together with the incident columns) were replaced. Then, a covering was calculated for the rest of the matrix using the following methods: random (RAND: we choose a random column to our covering until all matrix is covered), where n - number of rows in the matrix (equal to ν in the first step).

The large value of C_1 means, that the column i covers many rows. Unfortunately, this coefficient is not sufficient to distinguish between "good" and "bad" columns, because many of them have the same value of C_1 (actually, in the first step of the algorithm all the values are the same).

The second coefficient, $C_2(i)$, is calculated as follows: first, a sum of C_1 is calculated for each row (i.e. C_1 of these columns, which have "1" in the row); the large sum means, that the row is "easy to cover". Then, for each column sum of these sums is calculated (once again, for these rows, which have "1" in the column); the small value means, that the column covers rows that are "hard to cover" by other columns therefore the rest of a covering process should be relatively easy. If more than one column has the same value of C_1 and C_2, the algorithm chooses the first (in particular order) of them..

Example 1. A simple 10×10 matrix with calculated coefficients is presented below:

Col.	1	2	3	4	5	6	7	8	9	10	Σ C_1
	1	0	0	0	1	0	0	1	0	0	16
	0	0	1	0	0	1	1	0	0	0	13
	0	1	1	1	1	1	0	1	0	0	29
	1	0	0	1	0	1	0	0	1	0	18
	1	0	0	0	1	0	0	1	0	0	16
	1	1	1	0	1	1	0	1	0	1	33
	0	1	0	0	0	0	1	0	0	1	8
	0	0	1	0	1	1	0	0	1	0	20
	1	0	1	1	1	0	0	0	0	0	22
	1	1	1	1	0	0	0	0	1	0	23
C_1	6	4	6	4	6	5	2	4	3	2	
C_2	92	77	104	76	100	88	17	78	52	37	

In this case, columns 1, 3 and 5 have the largest value of C_1; on the other hand, column 1 has the lowest value of C_2 (it covers rows 1, 4 and 5, which are "hard to cover"). So we should choose column 1 to our covering. We remove column 1 and rows 1, 4, 5, 6, 9 and 10 and we continue procedure with the reduced 9×4 matrix.

The coefficients C_1 and C_2 can be easily calculated for entire matrix by approximately 3(n×m) operations, so the algorithm is relatively fast.

2.2 Weighted greedy algorithm - method 2 (WGA2)

Unfortunately, for the special case of BIB matrices method WGA1 is not sufficient: the most of columns has the same value of C_1 and C_2. Thus, another type of weighted greedy algorithm was proposed:

1. Calculate coefficients (weights) D_1, D_2, D_3 and D_4 for all columns of the matrix.
2. Choose a column with the largest value of D_1.
3. If there are more than one such a column, choose a column with the largest value of D_2 from among them.

4. If there are more than one such a column, choose a column with the largest value of D_3 from among them.
5. If there are more than one such a column, choose a column with the least value of D_4 from among them. If we still have more than one column, choose the first of them.
6. Add a column chosen as above to a set of covering columns.
7. Remove the column from the matrix. Remove all rows which have "1" in that column.
8. Continue from step 1 with the reduced matrix. Stop when all rows are covered.

Now we have four coefficients to calculate for each column. First of them, D_1, is equal to C_1, and reflects to the number of "1" in the i-th column.

The second coefficient, D_2, is calculated only for these columns, which have maximal value of D_1. To calculate this coefficient, we remove the i-th column from the matrix (together with adjacent rows) and calculate D_1 for this reduced matrix. The value of $D_2(i)$ is equal to the maximal value of D_1 calculated for the reduced matrix.

$$D_2(i) = \max_k \left(\sum_{j=0}^{n-1} a_{kj}^{(i)} \right) \tag{4}$$

where $\{a^{(i)}\}$ denotes the matrix with removed i-th column. This coefficient should be maximized, because we are interested in a situation, that after removing i-th column we still are able to remove another column with many "1"-s.

The third coefficient is concerned with the second one. D_3 is defined as a number of columns in the reduced matrix with maximal value of D_1:

$$D_3(i) = card\left\{ k: \sum_{j=0}^{n-1} a_{kj}^{(i)} = D_2(i) \right\} \tag{5}$$

The large value of D_3 means, that if we choose this column for our covering, in the next step we will have many columns with high value of D_1 to choose. Thus, we tend to maximize D_3.

These three coefficients works quite good and are in most cases efficient enough to create a short covering. Unfortunately, because of the special form of BIB matrices, many different columns have the identical values of all three coefficients (especially in the first and the second step of algorithm). Thus, the fourth coefficient was introduced:

$$D_4(i) = \min_{j:\, a_{ij}=1} \left(\sum_{k=0}^{m-1} a_{kj} \right) \tag{6}$$

i.e. the sum of minimal adjacent row. We want D_4 to be minimal, because it means, that the i-th column covers a row which is covered by few other columns, so it is "hard to cover".

In a rare case when there are more than one column with all these coefficients optimal, we have to choose the first of them (in particular order).

Example 2. Let us consider a matrix from the Example 1. The values of D_1 are equal to C_1, so we have three columns with maximal value of D_1 (columns 1, 3 and 5) and for these columns we will calculate the rest of coefficients. Let us consider column 1. Suppose we remove this column from the matrix, together with six adjacent rows. We calculate sums of columns in such a reduced matrix and see, that two of them (column 3 and 6) will have maximal value of sum equal to 3. Thus, $D_2(1)=3$ and $D_3(1)=2$. We calculate coefficients for the column 3 and 5 in similar way: $D_2(3)=3$, $D_3(3)=1$, $D_2(5)=2$, $D_3(5)=7$. Our algorithm chooses the first column, because of maximal D_2 and (secondary criterion) better D_3. We do not need to calculate D_4, because there is only one column with maximal D_2 and D_3. We remove column 1 from the matrix (together with adjacent rows) and continue with reduced 9×4 matrix.

2.3 Randomized weighted greedy algorithm (RWGA)

The algorithms described above produce good (in a sense of columns number) coverings in short time. However, the coverings are not always optimal; sometimes we could find a shorter one using another method. Thus, a modification of both WGA1 and WGA2 was introduced - a modification based on a random search near the result obtained by the pervious algorithm.

Our new algorithm acts as follows:

1. Calculate coefficients (weights) as in algorithm WGA1 or WGA2 for all columns of the given matrix.
2. For a given parameter (small integer) K, choose a random value k uniformly from [1..K].
3. WGA1: Sort the columns of the matrix using values of C_1 (descending) and C_2 (secondary criterion, ascending). Now a column with the largest value of C_1 and the lowest value of C_2 (among these with equal C_1) is on the beginning of the list.

WGA2: Sort the columns of the matrix using values of D_1 (descending), then D_2 (descending), then D_3 (descending), then D_4 (ascending). Now a column with the largest value of D_1 and D_2 and D_3 and the lowest value of D_4 is on the beginning of the list.
4. Choose the k-th column from among the sorted ones.
5. Add the column chosen as above to a set of covering columns.
6. Remove the column from the matrix. Remove all rows which have "1" in that column.
7. Continue from step 1 with the reduced matrix. Stop when all rows are covered.

The algorithm based on WGA1 we will refer to as RWGA1, the one based on WGA2 we will refer to as RWGA2.

It is easy to see, that our new algorithm is very similar to the pervious ones. The difference is, that we do not choose the best (in a sense of coefficients) column to our covering, but the k-th one. Thus, our algorithm becomes nondeterministic, and it may be executed many times to test another solutions. The more time we have, the better result we obtain.

Example 3. We will use RWGA1 in this example. Let us consider Example 1. We sort columns due to C_1 and then due to C_2 and obtain the following order: column 1, column 5, column 3, column 6, column 4, column 2, column 8, column 9, column 7, column 10. Suppose we have K=5 and we have chosen randomly k=2. So we have to get the second column in sorted order, i.e. the column 5 (the best value of C_1, the second value of C_2). If we choose k=4, we get column 6.

2.4 Genetic algorithm (GA)

Results obtained by RWGA in comparison with these obtained by WGA suggests, that a further improvement is possible. As a next approach to the covering problem, a genetic algorithm (see [6], [7]) was used to generate solutions.

In a fact, the algorithm described below is a hybrid genetic-greedy algorithm (see [8]), where an evolution process is used as a metaheuristics. The algorithm bases on the RWGA1 (denoted by GA1) or RWGA2 (denoted by GA2) controlled by a sequence $\boldsymbol{\kappa} = \{ \kappa_1, \kappa_2, \ldots \kappa_v \}$, $\kappa_i \in [1, \ldots \nu]$, and acts as follows:

1. Calculate coefficients (weights) for all columns of the given matrix.
2. Choose $k = \kappa_i$, where i = 1 for the first column in the covering, i = 2 for the second one (the second turn of algorithm) etc.
3. Sort the columns of the matrix using values of coefficients as in RWGA1 or RWGA2 algorithm.
4. Choose the k-th column from among the sorted ones.
5. Add the column chosen as above to a set of covering columns.
6. Remove the column from the matrix. Remove all rows which have "1" in that column.
7. Continue from step 1 with the reduced matrix. Stop when all rows are covered.

Let WGA($\boldsymbol{\kappa}$) denotes the modified algorithm. Now, the value k is not chosen randomly, but it comes from a given sequence $\boldsymbol{\kappa}$. So, the result depends on the values of $\boldsymbol{\kappa}$. These values are optimized by a genetic algorithm.

Example 4. Once again let us consider Example 1. We will use GA1 method. Suppose we have an individual with chromosome { 2, 1, 1, 3, 2, 1, 2, 2, 1, 1 }. We have sorted columns as in RWGA1 algorithm: column 1, column 5, column 3, column 6, etc. In the first step we get the first value $\kappa_1=2$ from our chromosome, so k=2 and we get the second column in the order calculated due to coefficients (i.e. the column 5). We reduce the matrix by removing column 5 and rows 1, 3, 5, 6, 8 and 9. Next we calculate coefficients for the reduced 9×4 matrix and sort columns due to these new coefficients. Next, we get $\kappa_2=1$ from our chromosome, so we have k=1 and we will select the first column in the new order (it will be column 1, because only this column covers 3 rows among these 4

rows not covered by column 5). The matrix will be reduced to trivial 8×1 case, but formally we have to calculate coefficients once again and sort columns due to these new values. Finally we use $\kappa_3=1$ and select the first column due to this new order to our covering. We stop, because the whole matrix is covered. The rest of our chromosome (from κ_4 to κ_{10}) is not used.

The genetic algorithm used in our problem (**ordinal-based genetic algorithm**) works on the individuals of the form $\boldsymbol{\kappa} = \{ \kappa_1, \kappa_2, \ldots \kappa_v \}$, $\kappa_i \in [1, \ldots v]$ - this is the most natural way of coding in our problem. A genetic operators, however, have to be redefined, because the form of individuals is not classical. A mutation operator affects random position of random individual, increasing (with probability of P_+) or decreasing it by one. E.g.:

$$\{\ldots,\kappa_i,\ldots\} \xrightarrow{\text{mutation on pos. } i} \begin{cases} \{\ldots,\kappa_i+1,\ldots\} & \text{prob. } P_+ \\ \{\ldots,\kappa_i-1,\ldots\} & \text{prob. } 1-P_+ \end{cases}$$

In our experiments, $P_+ = 0.5$. If $\kappa_i = 1$, it cannot be decreased (mutation is ignored). An additive technique was used to increase efficiency: the mutation probability was increased when the best individual does not increases its fitness during some steps.

A crossover operator is similar to the classical one (see [6]). First, a random location i is chosen uniformly from $[1, \ldots j]$, $j \le v$, then the first i genes from the parents are exchanged. E.g:

$$\begin{matrix} \{\kappa_1^1\ldots,\kappa_i^1,\kappa_{i+1}^1\ldots\} \\ \{\kappa_1^2\ldots,\kappa_i^2,\kappa_{i+1}^2\ldots\} \end{matrix} \xrightarrow{\text{crossover to pos. } i} \begin{matrix} \{\kappa_1^2\ldots,\kappa_i^2,\kappa_{i+1}^1\ldots\} \\ \{\kappa_1^1\ldots,\kappa_i^1,\kappa_{i+1}^2\ldots\} \end{matrix}$$

In our problem, the large part of every individual is "inactive": when we find a covering, we stop the WGA($\boldsymbol{\kappa}$) without using the rest of κ_i values. If the crossover point is chosen to be outside the "active" part of individuals, the operator in a fact will do nothing. Therefore, the parameter j (the maximal position of crossover point) was set to a value of expected covering length.

When we have to calculate a fitness value for an individual $\boldsymbol{\kappa}$, we find a covering concerned with it by executing the WGA($\boldsymbol{\kappa}$). Let L($\boldsymbol{\kappa}$) be a length of this covering. Because our goal is to find a sequence $\boldsymbol{\kappa}$ which generates the shortest covering, we can consider a fitness function F($\boldsymbol{\kappa}$) as follows:

$$F(\boldsymbol{\kappa}) = v - L(\boldsymbol{\kappa}) + 1 \qquad (7)$$

where v is a size of matrix. Unfortunately, the results obtained for this fitness function was poor. The reason was, that this function does not distinguish between two solutions of the same size. When the entire population consists of individuals generating coverings of equal size (it was quite frequent situation), we should can find and promote "more promising" ones. So, the additional quality measure was introduced. Let L'($\boldsymbol{\kappa}$) be a number of steps of WGA($\boldsymbol{\kappa}$), after which the half of rows was covered. We have L'($\boldsymbol{\kappa}$) < L($\boldsymbol{\kappa}$). When we have two coverings of equal length, the one with smaller L'($\boldsymbol{\kappa}$) is "more promising", because it has "good beginning" and may be easier to improve (by mutation or crossover). Now, our fitness function receives its final form:

$$F(\boldsymbol{\kappa}) = \left(v - L(\boldsymbol{\kappa}) + 1\right) + \frac{L(\boldsymbol{\kappa}) - L'(\boldsymbol{\kappa})}{L(\boldsymbol{\kappa})} \qquad (8)$$

We use the standard selection method of the "roulette wheel" (see [6]). A linear scaling was applied to the fitness values before selection.

2.5 Experimental results

The experiment was performed as follows: the BIB matrix of size v×v was generated and σ random rows (together with the incident columns) were replaced. Then, a covering was calculated for the rest of the matrix using the following methods: random (RAND: we choose a random column

to our covering until all matrix is covered), WGA1, RWGA1, GA1, WGA2, RWGA2 and GA2. A computation time and average size of covering for about 1000 trials are presented in Table 1.

Matrix		RAND	WGA1		RWGA1		GA1		WGA2		RWGA2		GA2		range
ν	σ	result	time [s]	result	time [s]	result	time [s]	result	time [s]	result	time [s]	result	time [s]	result	
21	4	**5.98**	0.0005	**5.64**	0.03	**5.53**	0.51	**5.53**	0.0006	**5.53**	0.04	**5.53**	0.52	**5.53**	4 - 6
31	5	**8.71**	0.0012	**7.56**	0.08	**7.13**	5.59	**7.05**	0.0017	**7.05**	0.09	**7.05**	1.39	**7.05**	5 - 8
57	7	**15.58**	0.0051	**11.62**	0.33	**11.02**	8.14	**10.77**	0.009	**10.87**	0.45	**10.70**	7.01	**10.73**	10-12
73	8	**19.67**	0.0096	**13.76**	0.62	**12.99**	11.1	**12.67**	0.017	**12.81**	0.92	**12.34**	13.3	**12.37**	11-15
91	9	**23.90**	0.0156	**15.94**	1.02	**15.09**	17.5	**14.62**	0.033	**14.90**	1.49	**14.21**	22.5	**14.26**	13-16
133	11	**33.67**	0.0381	**20.43**	2.48	**19.35**	46.5	**18.76**	0.075	**19.11**	4.01	**18.20**	60.0	**18.21**	18-21

Table 1. Results of experiments for different BIB matrices and methods. All computations performed on Pentium-200 machine.

We use RWGA1 method with K=3, whereas in case of RWGA2 we put K=5. These algorithms were passed 50 times on each randomly reduced BIB matrix and the best result was taken.
GA1 parameters: population size: 20 individuals, evolution time: 68 generations, prob. of crossover: 0.8, initial prob. of mutation: about 0.01 (per gene).
GA2 parameters: population size: 20 individuals, evolution time: 40 generations, prob. of crossover: 0.6, initial prob. of mutation: about 0.01 (per gene).
The column "range" contains minimal and maximal covering length found by RWGA2.

The parameters of RWGA and GA was chosen to optimize program efficiency: with another value of parameter K, fitness functions, values of mutation or crossover probabilities the results was slightly worse. The fewer number of generations in case of GA2 is concerned with the faster convergence of this version of GA.

The results presented in Table 1 are calculated for the maximal value of σ. On the other hand, in practice we will use these methods for the smaller values as well. Results for average size of coverings for the 133×133 matrix (average for 500 cases) are presented in Table 2. These results show, that the difference between RWGA and WGA rises when a problem becomes harder.

σ	WGA 1	WGA2	RWGA2
1	21.00	21.00	21.00
2	20.00	20.00	20.00
3	20.12	19.91	19.91
4	20.08	19.34	19.26
5	20.09	19.05	19.02
6	20.09	19.03	18.94
7	20.10	19.01	18.59
8	20.07	18.94	18.25
9	20.22	18.96	18.19
10	20.27	19.06	18.22
11	20.43	19.11	18.20

Table 2. Average size of covering for ν=133.

3 About Optimality of Broadcast key distribution schemes based on 2-designs

Let us consider a family of (**V**, **B**) key distribution schemes based on incident structure, where **V**, |**V**|=v a set of users and B, |B|=b is a set of some subsets **V**. Then the total number of keys which should be generated by the center and distributed to users during the installation of the system will be equal to b.

Another important parameter is the number of keys belonging to each of users. For simplicity assume that all users have the same number of keys N and any of keys belongs to the same number of users k at once after the installation of the system). This means in terms of incident matrix that both all rows and columns have the same number of ones. This assumption results in the following relation

$$v\,N=b\,k \qquad (9)$$

(Recall that for an incident structure based on symmetric 2-designs we always obtain v=b and N=k).

Claim. For any broadcast key distribution scheme based on incident structure the following inequalities for the connectivity and the resilience should be true

$$\chi_\sigma \geq \frac{v-\sigma}{k}, \quad \sigma \leq S \qquad (10)$$

$$S \leq N-1, k \geq 2 \qquad (11)$$

Proof. The first inequality follows immediately from the fact that we obtain the best case for a minimization of the connectivity, when supports of columns of incident matrix are disjoint. The second inequality follows if we take as illegitimate users such of them which have common keys with any given legitimate user. Since he has N keys we always can choose such the case M illegitimate users to remove all M keys belonging to any legitimate user.

Example 5. Let us consider 2-design presented in the table 1 (see also [1]) with parameters b=v=133, k=r=N=12, σ=11, $\chi_{11}\approx 18$ (on the average). On the other hand, we obtain from (9), (10) and (11) that for any broadcast key distribution scheme based on incident structure with parameters b=v=133, k=N=12, the following inequalities should be true: $s \leq 11$, $\chi_{11} \geq 10$. We can see that 2-design gives an optimal solution relative to a resilience but far removed enough from the lower bound (10) for the connectivity. (However, the last fact is rather due to the crude lower bound (10) but not due to bad properties of these schemes).

It is worth noting that there are some modifications of broadcast key distribution schemes based on 2-designs which have b<v, s=N-1 and k≠N, which are also close to optimal ones relative to a resilience.

Another problem is an optimization of parameters for 2-designs given the total number of users L. At first glance it would be seem that we should take 2-design with parameter v=L. In fact, the more is v for 2-design the less is the relative length of the enabling block χ_s/v. (For example χ_s/v= 0.28 if we use 2-design with v=7, and χ_s/v=0.13, if we take 2-design with v=133). This means that we could decrease the use of channel to transmit pay television if we increase v. But on the other hand, the more is v the less is the ratio s/v=r-1/v. (For example s/v=0.285 if we use 2-design with v=7 and s/v=0.083 if we take 2-design with v=133). This means that if we define a model of compromisations for users to be a binomial one with the probability of compromisation of each user equal to p then the probability p(L) that at least one legitimate user among L requires some extra distribution of keys outside of 2-design scheme, can be found as follows

$$P(L)=1-(1-p_0(v))^{L/v}, \qquad (12)$$

where $p_0(v)= \sum_{i=S+1} \binom{v}{i} p^i (1-p)^{v-i}$

Now we should take v to provide the maximal possible value p given L and p(L) and it is not necessary to be v_{opt}=L.

Example 6. Let us take L=133 and $p(L)\approx 10^{-2}$. (This means that the user descards his subscription of pay TV with the probability 10^{-2}). Then we obtain from (12) that $p\approx 0.03$ for v=7 and p=0.07 for v=133.

So we should take into account both the relative length of the enabling block and a maximization of p given p(L) and L when we choose an appropriate 2-design as a broadcast key distribution scheme.

4 Summary and open problems

A few methods for row covering problem in special case of reduced 2-design matrices were presented. Experimental results show, that relatively fast RWGA2 method is efficient for this problem. On the other hand, in some real-time applications we can use suboptimal, but very fast WGA2 method. All methods based on WGA1 was worse than these based on more complicated WGA2.

It is interesting to note, that GA1 gives better results than RWGA1, whereas GA2 is a bit worse than RWGA2. The cause is, that WGA2 itself (and its randomized version) is very effective, so the usage of genetic algorithm cannot improve results. We can see this especially for case of 21×21 and 31×31 matrices. For these matrices WGA2 in most cases gives the optimal covering - so no other method could give better results. All these methods give much better results than simple covering method based on random choice of columns.

The upper bound (1) for χ_σ obtained in [1] is also much worse than the experimental results. So for v=133, σ=11 we have the upper bound $\chi_{11}\approx 437$ (it is worse also than trivial bound $\chi_{11} \le 133$) whereas the best algorithm of covering RWGA2 gives on average $\chi_{11}\approx 18$.

An advance to solution of covering problem for 2-design incident matrices open a real perspective to put into practice broadcast key distribution schemes based on 2-designs. It is possible to choose parameters of 2-designs in such a way to provide the best reliability of them.

We mentioned also a possibility to construct schemes which have less number of total keys b at the cost of lesser resilience. An obvious open problem is to develop a regular theory of broadcast key distribution schemes based on incident structure for b>v to provide more large resilience and lesser connectivity.

In would be nice to modify the bounds (10) and (11) providing the use of “nonuniform” schemes where each of users can obtain different number of keys at the initial stage and each of keys can be distributed to different number of users. And at last, more “powerful” 2-design (v>133) having the known algorithm for their construction would be very useful for further investigations.

References

[1] V.Korjik, M.Ivkov, Y. Merinovich, A. Barg and Henk va van Tilborg, “A Broadcast key Distribution Scheme Based on Block Designs”, Proc. 5th IMA Conf. Cryptography and Coding., 1995.

[2] C. Blundo, A. De Santis, A. Herzberg, S. Kutten,U. Vaccaro, M. Yung, “Perfectly-Secure Key Distribution for Dynamic Conferences”, Proc. of Crypto’92, pp.471-485.

[3] E.F. Assmus and J.D.Key, “ Designes and Their Codes”, Cambr. Univ. Press,1993.

[4] G.Cohen, S.Litsyn, G. Zemor, “On Greedy Algorinhm in Coding Theory”, IEEE Trans an IT. Vol 42, N6,1996, pp.2053-2057.

[5] B.Chor, A. Fiat, M.Noar, “Tracing Traitors ”, Proc. Crypto’94, pp.257-270.

[6] Goldberg D.E., 1989. “Genetic Algorithms in Search, Optimisation, and Machine Learning”. Addison-Wesley.

[7] Holland J.H. 1975. “Adaptation in natural and artificial systems”. The MIT Press, Cambridge, 1992.
[8] Wróblewski J., 1996. “Theoretical Foundations of Order-Based Genetic Algorithms”. Fundamenta Informaticae, vol. 28 (3, 4), pp: 423-430. Kluwer, Dordrecht 1996.