Open Journal of Discrete Mathematics, 2016, 6, 238-247
http://www.scirp.org/journal/ojdm
ISSN Online: 2161-7643
ISSN Print: 2161-7635

Near-Optimal Placement of Secrets in Graphs

Werner Poguntke

Fachbereich Technische Betriebwirtschaft, Fachhochschule Südwestfalen, Hagen, Germany
Email: poguntke.werner@fh-swf.de

How to cite this paper: Poguntke, W. (2016) Near-Optimal Placement of Secrets in Graphs. *Open Journal of Discrete Mathematics*, **6**, 238-247.
http://dx.doi.org/10.4236/ojdm.2016.64020

Received: January 5, 2016
Accepted: August 30, 2016
Published: September 2, 2016

Abstract

We consider the reconstruction of shared secrets in communication networks, which are modelled by graphs whose components are subject to possible failure. The reconstruction probability can be approximated using minimal cuts, if the failure probabilities of vertices and edges are close to zero. As the main contribution of this paper, node separators are used to design a heuristic for the near-optimal placement of secrets sets on the vertices of the graph.

Keywords

Graph Algorithm, Cut, Secret Sharing, Approximation, Network Design

1. Introduction

We consider a scenario where a set of secrets is shared among individuals connected by a communication network, in such a way that no individual holds all the secrets. In other words, several individuals have to cooperate in order to reconstruct the whole secret set.

Secret sharing schemes were first introduced and investigated in [1] and [2]. In an (m, k)-*threshold scheme*, a secret is divided into k shares in such a way that the secret can be reconstructed whenever at least m of the shares have been collected. Survey papers on secret sharing schemes and threshold schemes are [3] and [4].

In this paper, we always assume $m = k$, *i.e.* it is necessary to collect all of the shares in order to reconstruct the secret. Subsets of the set of all secrets (called *shares*) are stored in the nodes of a communication network whose nodes and links are subject to failure with certain probability. One vertex is assumed to be the *user node*.

We consider two main problems:

- to calculate the reconstruction probability of the secrets, given an assignment of shares to vertices, and
- to assign shares to vertices such that the reconstruction probability of secrets gets as large as possible.

Papers closely related are [5]-[7].

As the main contribution of this paper, we present an approximation algorithm for the determination of the reconstruction probability, as well as a heuristic for the near optimal placement of shares.

2. Shared Secret Schemes in Networks with Failures

2.1. The Model

In this paper, a communication network is modelled by a finite undirected graph $G = (V,E)$, where V consists of n vertices, and E is the set of edges. Let a finite set $S = \{s_1, s_2, \cdots, s_k\}$ of *secrets* be given, and let Σ be a set of nonempty subsets (*shares*) of S. One node v_u in V is supposed to be the *user node*. A *shared secret scheme* or *secret sharing scheme* on G is a 1-1 mapping

$$\sigma : \Sigma \to V - \{v_u\},$$

i.e. each of the selected shares is placed on some node of the graph other than the user node.

It is further assumed that the vertices as well as the edges of G may possibly fail, *i.e.* they work with a certain probability only, and that the states of all single vertices and edges are independent from each other. In this paper, this probability is assumed to equal a fixed p ($0 \le p \le 1$) for all vertices and all edges. The only exception is the user node v_u; for technical reasons which will become clear later, it is assumed that v_u always works.

The *reconstruction probability* of (G, S, σ) is the probability that the complete secret set S can be reconstructed by the user node, *i.e.* the probability that along paths using vertices and edges not having failed and starting from node v_u, it is possible to collect all the secrets $s_1, s_2, \cdots, s_k$. It is obvious that as a function of p, the reconstruction probability is a polynomial. We denote this polynomial by $r(p)$.

More formally, we call any subset $X \subseteq V \cup E$ a *state*. A state X is *operational* if the secrets can be reconstructed provided each element of X works. In these terms, the reconstruction probability is the probability that the vertices and edges not having failed constitute an operational state.

One problem is to determine the polynomial $r(p)$. It can also be of interest to merely find the value $r(p_0)$ for a given p_0. Given the graph G, set Σ of shares and probability value p_0 for all vertices and edges, another problem is to design a shared secret scheme (*i.e.* placement of shares on the vertices) such that $r(p_0)$ becomes maximum.

2.2. Introductory Examples and Previous Results

The diagram in Figure 1 shows a graph G_1 consisting of eight vertices and twelve edges. As in all the examples of this paper, the node labelled "1" is the user node v_u. Four secrets s_1, s_2, s_3, s_4 are given, and Σ consists of the following six shares:

$$S_1 = \{s_1\},\ S_2 = \{s_2\},\ S_3 = \{s_3\},\ S_4 = \{s_4\},\ S_5 = \{s_1, s_3\},\ S_6 = \{s_2, s_4\}.$$

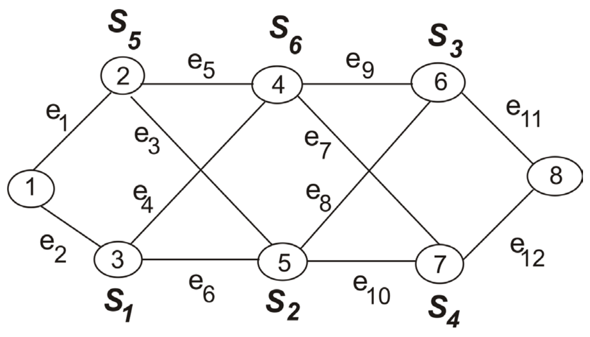

Figure 1. Six shares on graph G_1.

A shared secret scheme (*i.e.* placement of shares on vertices) is also shown in the diagram. The reconstruction probability polynomial turns out to be:

$$r(p) = p^5 + 4p^7 - 2p^8 + 6p^9 - 10p^{10} - 4p^{11} + p^{12} - 6p^{13} + 30p^{14}$$
$$-37p^{15} + 58p^{16} - 83p^{17} + 62p^{18} - 22p^{19} + 3p^{20}$$

Hence, e.g., $r(0.9) \approx 0.9242$[1].

Different variants of the model and related problems have been considered by many authors. Nearly all of the problems turn out to be NP-hard. In particular, it is easy to see that determining what we have called $r(p)$ is a generalization of the *graph reliability problem*. For the basic results in this field, we refer to [8] and [9]; a lattice-theoretic approach described in [10] and [11] is the basis of [7].

In [6], $r(p)$ is calculated by constructing *minimal share spanning trees*. Also, a simple share assignment algorithm is presented providing near-optimal share assignments efficiently. In this algorithm, the main strategy is to place large shares on vertices close to v_u. As the following example shows, this does not always lead to optimal results:

For the same graph G_1 as in **Figure 1**, consider the share assignment shown in **Figure 2**. The two-element shares are placed on the neighbours of node 1. Surprisingly, this scheme is slightly less reliable than the one we considered in **Figure 1**. (An explanation for this will be given below.) In particular, it turns out that

$$r(p) = p^5 + 6p^7 - 4p^8 - 4p^{10} + 2p^{11} - 4p^{12} - 10p^{13} + 34p^{14} - 56p^{15} + 118p^{16} - 164p^{17} + 118p^{18} - 42p^{19} + 6p^{20}$$

with $r(0.9) \approx 0.9223$.

As usual, a subset C of $V \cup E$ is called a *cut* if $(V \cup E) \setminus C$ is not operational, *i.e.* failure of all the elements of C makes it impossible for v_u to reconstruct the complete secret set. A cut C is a *mincut* if it is inclusion minimal as a cut, *i.e.* no proper subset of C is a cut.

Mincuts play a central role in the rest of this paper. The dual approach based on inclusion-minimal operational sets (sometimes called *minpaths*) is used in [6]. For a survey on the roles of cuts and paths in network reliability, see [8].

For s in S, call a subset C of $V \cup E$ an *s-separator* if failure of all the elements of C makes it impossible for v_u to collect s. In this terminology, a cut is a subset which is an

[1]In this paper, all calculations were done using Matlab.

s-separator for at least one *s* in *S*. In [12], an algorithm is described generating all minimal *s*-separators.

In the following, to make a clear distinction, and following the terminology of [13], we call a subset $C \subseteq V$ of nodes a *node separator* if removing *C* disconnects *G*, *i.e.* the graph G_C induced by $V \setminus C$ is not connected; in this case, if *s* and *t* are nodes belonging to different components of G_C, *C* can also be viewed as an *s-t-separator*.

To illustrate the notion of cuts for secret sharing schemes, we look at another example which was also considered in [6]. **Figure 3** shows a graph G_2 consisting of eight vertices and eleven edges. As in the preceding examples, four secrets s_1, s_2, s_3, s_4 are given, and Σ consists of the following six shares.

$$S_1 = \{s_1\},\ S_2 = \{s_2\},\ S_3 = \{s_3\},\ S_4 = \{s_4\},\ S_5 = \{s_1, s_3\},\ S_6 = \{s_2, s_4\}$$

A shared secret scheme is also shown in the diagram, with reconstruction polynomial

$$r(p) = 3p^5 - 2p^6 + p^7 + 7p^9 - 8p^{10} - 6p^{11} + 3p^{12} - 7p^{13} + 24p^{14} - 27p^{15} + 38p^{16} - 45p^{17} + 25p^{18} - 5p^{19},$$

and $r(0.9) \approx 0.9329$.

In this example, there are no one-element mincuts. The mincuts consisting of two elements turn out to be the following: $\{2,4\}$, $\{2,6\}$, $\{3,4\}$, $\{3,5\}$, $\{3,7\}$, $\{3,e_4\}$. Furthermore, there are 21 three-element mincuts, 33 four-element mincuts, 25 five-element mincuts, and only few mincuts containing six or more elements.

Figure 4 presents a slightly better placement of the shares on the nodes of G_2 (this example was also presented in [14]). The reconstruction polynomial turns out to be

$$r(p) = 3p^5 - 2p^6 + 2p^7 + 4p^9 - 7p^{10} - 5p^{11} + 3p^{12} - 6p^{13} + 23p^{14} - 25p^{15} + 33p^{16} - 41p^{17} + 24p^{18} - 5p^{19},$$

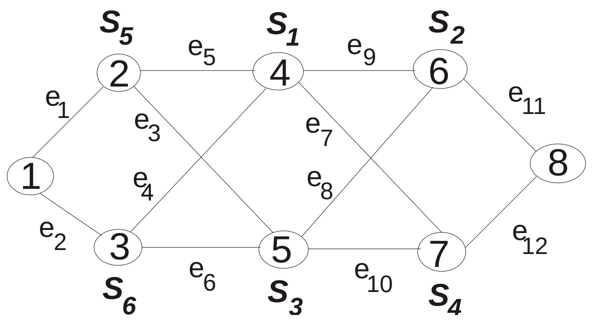

Figure 2. Another share assignment on graph G_1.

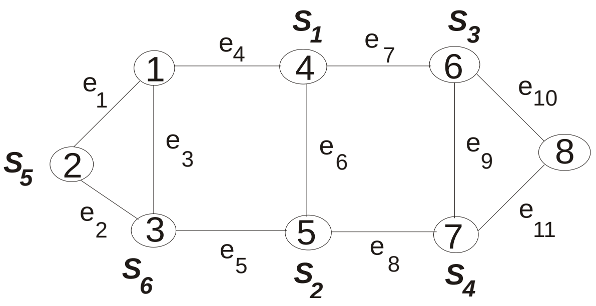

Figure 3. Six shares on graph G_2.

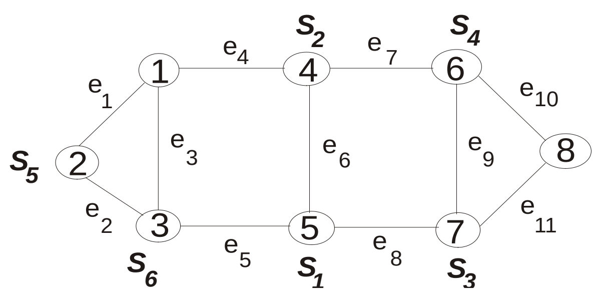

Figure 4. Another share assignment on graph G_2.

with $r(0.9) \approx 0.9398$. There are 5 two-element mincuts, 20 three-element mincuts, 32 four-element mincuts, 28 five-element mincuts, and only few larger ones.

3. Using Mincuts for Approximations

The following obvious fact is the basis of our approximation to $r(p)$.

Remark 1.

A state $X \subseteq V \cup E$ is not operational if and only if its complement contains at least one mincut, *i.e.* there is a mincut $C \subseteq (V \cup E) \setminus X$.

In other words, this means that $1 - r(p)$ equals the probability that all the elements of at least one mincut fail. Applying the inclusion-exclusion principle, this leads to a well-known formula which we rephrase as follows:

Theorem 1.

Let $\{C_1, C_2, \cdots, C_t\}$ be the collection of all the mincuts of a shared secret scheme. Then

$$1 - r(p) = \sum_{1 \le j \le t} \left[(-1)^{j+1} \sum_{\substack{T \subseteq \{1,2,\cdots,t\} \\ |T| = j}} (1-p)^{\left|\bigcup_{i \in T} C_i\right|} \right]$$

Proof:

Let f_i represent the statement "C_i fails" (*i.e.* each of its elements fails). Then by the above observation, we get:

$$1 - r(p) = prob(f_1 \text{ or } f_2 \text{ or} \cdots \text{or } f_t)$$

By inclusion-exclusion, this leads to:

$$1 - r(p) = \sum_i prob(f_i) - \sum_{i,j} prob(f_i \text{ and } f_j) + \sum_{i,j,k} prob(f_i \text{ and } f_j \text{ and } f_k) - \cdots$$

Using independence of the states of single elements, one finally gets the formula of the theorem.

If we now set

$$r_s(p) = 1 - \sum_{1 \le j \le s} \left[(-1)^{j+1} \sum_{\substack{T \subseteq \{1,2,\cdots,t\} \\ |T| = j}} (1-p)^{\left|\bigcup_{i \in T} C_i\right|} \right]$$

for $1 \le s \le t$, then obviously, $r_1(p), r_2(p), \cdots, r_t(p)$ is a sequence of approximations to $r(p)$, with $r_t(p) = r(p)$. To be more precise,

$$r_1(p) \le r_3(p) \le \cdots \le r(p) = r_t(p) \le \cdots \le r_4(p) \le r_2(p).$$

At this point, it is certainly plausible that if p_0 is close to 1.0, then $r(p_0)$ tends to strongly depend on the number of mincuts of small cardinality.

As usual, we define $q := 1 - p$.

For the above examples, let M denote the set of mincuts with two or three elements. As approximation $r_{app}(p)$ to $r(p)$ we define

$$r_{app}(p) := 1 - m_2 \cdot q^2 - m_3 \cdot q^3 + u_3 \cdot q^3 + 0.5 \cdot u_4 \cdot q^4,$$

where the following notation is used: m_2 is the number of two-element mincuts in M, m_3 is the number of three-element mincuts in M, u_3 is the number of unions of two elements of M that contain three elements, and u_4 is the number of unions of two elements of M consisting of four elements.

Table 1 gives an overview on the secret sharing schemes considered in the examples. As can be seen, for these graphs of modest size, $r_{app}(0.9)$ is quite a good approximation to the reconstruction probability, $r(0.9)$.

4. A Heuristic for Share Assignment Based on Node Separators

Once the relevance of mincuts for the reconstruction probability has become clear, we now turn to the question what makes a shared secret scheme have few mincuts. It turns out that, basically, there are two different effects that make a set X of vertices and edges a cut:

- X is a node separator, and the complete secret set S cannot be reconstructed by v_u only visiting vertices in its connected component
- X contains all vertices that carry one specific secret

To illustrate this, let us look at the example of Figure 3 again. $X = \{3,4\}$ is a cut, since failure of the two vertices 3 and 4 disconnects the graph, and the complete secret set cannot be reconstructed by v_u visiting only vertices in its connected component. Observe that $Y = \{6,7\}$ is not a cut, although it disconnects the graph. On the other hand, $Z = \{2,4\}$ does not disconnect the graph, but nevertheless is a cut, since none of the remaining vertices carries secret s_1.

It is now possible to identify the reason why the shared secret scheme of Figure 4 has fewer two-element mincuts than the scheme shown in Figure 3: In the scheme shown in Figure 4, $\{3,4\}$ is a mincut "for two reasons", namely it is a node separator, but it also constitutes a mincut since it contains all vertices carrying secret s_2; opposed to this, the example of Figure 3 has the "additional" mincut $\{3,5\}$.

Table 1. Reconstruction probabilities $r(0.9)$ and approximations $r_{app}(0.9)$ for the four examples.

Graph	Secret scheme	m_2	m_3	$r_{app}(0.9)$	$r(0.9)$
G_1	Figure 1	7	19	0.9230	0.9242
G_1	Figure 2	8	14	0.9243	0.9223
G_2	Figure 3	6	21	0.9322	0.9329
G_2	Figure 4	5	20	0.9385	0.9398

From these observations, we conclude that when designing a shared secret scheme, it is advantageous to place “secret mincuts” on node separators of the graph.

The heuristic presented also tries to avoid assigning a share to a node lying “behind” another node which carries a larger share. This point is illustrated via the small example presented in **Figure 5**: the assignment in diagram (a) is obviously more reliable than the one in diagram (b), since placing share $\{s_2\}$ on node 5 in (b) “behind” $\{s_1, s_2\}$ on node 3 does not make any sense.

We are now ready to present our heuristic for share assignment. The main idea is to place large shares on nodes close to v_u (as in [6]), but to also take into account node separators close to v_u. The restriction to node separators close to v_u keeps the algorithm polynomial in the size of G, independent of the size and structure of Σ.

We assume a graph $G = (V, E)$ with user node v_u as well as a set of shares $\Sigma \subseteq \mathrm{P}(S)$ are given, with $V = \{v_1, \cdots, v_n\}$, $S = \{s_1, \cdots, s_k\}$, and $|\Sigma| \le n-1$. A secret sharing scheme (1-1 mapping)

$$\sigma : \Sigma \to V - \{v_u\},$$

is defined by iteration, according to the following algorithm.

We begin with precomputations consisting of three algorithms:

- Algorithm *BFS-tree*
- Algorithm separators
- Algorithm *list of* shares

We next describe the algorithm for share assignment.

It is assumed that the algorithms *BFS-tree*, *separators*, and *list of shares* have been executed and have produced the numbering $V = \{u_1, \cdots, u_n\}$ of nodes with $u_1 = v_u$, the set *Minsep* of separators close to v_u, and the list *Share list* of all shares.

Algorithm *BFS-tree*
- builds up a breadth-first-search tree with renumbered nodes $V = \{u_1, \cdots, u_n\}$

Input: undirected and connected graph $G = (V, E)$, user node $v_u \in V$
Output: tree $T = (V, E')$ with $E' \subseteq E$ and renumbering $V = \{u_1, \cdots, u_n\}$ with $u_1 = v_u$

```
begin
  $i := 0$;
  $E' := \varnothing$;
  mark $v_u$;
  $List := [v_u]$;
  while $List \ne \varnothing$ do
      remove the first element $v$ of List;
      $i := i + 1$;
      $u_i := v$;
      while $u_i$ has an unmarked neighbor do
          choose an unmarked neighbor $w$ of $u_i$;
          mark $w$;
          add $w$ to the end of List;
          $E' := E' \cup \{\{u_i, w\}\}$
      end_while
  end_while
end
```

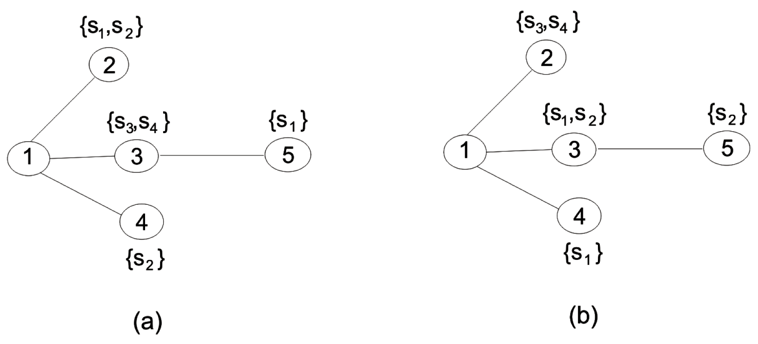

Figure 5. Two share assignments.

```
Algorithm separators
-    finds the minimal node separators close to $v_u$

Input:     undirected and connected graph $G=(V,E)$, user node $v_u \in V$
Output:    $Minsep \subseteq \mathrm{P}(N_u)$, where $N_u$ is the set of neighbors of $v_u$, and Minsep
consists of all minimal separators close to $v_u$ (following the terminology of [5])

begin
    determine $\mathrm{X} := \{C \mid C$ is a component in the graph induced by $V \setminus N_u\}$;
    $Minsep := \varnothing$;
    for each $C \in \mathrm{X}$ do
        determine set $D \subseteq N_u$ of all neighbors of $v_u$ with at least one neighbor in C;
        $Minsep := Minsep \cup \{D\}$
    end_for
end
```

```
Algorithm list of shares
-    builds up a list of all shares for the order of assignment

Input:     set $\Sigma \subseteq \mathrm{P}(S)$ of all shares, with set of secrets $S=\{s_1,s_2,\cdots,s_k\}$
Output:    a list Sharelist of all shares

begin
    choose an element $U \in \Sigma$ of maximal cardinality;
    $Sharelist := [U]$;
    $\Sigma := \Sigma \setminus \{U\}$;
    $union := U$;
    if $union = S$ then $union := \varnothing$ end_if;
    while $\Sigma \neq \varnothing$ do
        choose $T \in \Sigma$ such that $|union \cup T| = \max_{V\in\Sigma} |union \cup V|$
            and T is isof maximum size;
        add T to the end of Sharelist;
        $union := union \cup T$;
        if $union = S$ then $union := \varnothing$ end_if
    end_while
end
```

We next describe the algorithm for share assignment.

The algorithm *share assignment* assigns a share to the next node according to the following principles:

- (1st priority) nodes inside node separators should carry common secrets
- (2nd priority) nodes along paths in the BFS-tree should not carry common secrets

Algorithm *share assignment*

```
begin
    let $U$ be the first share in Sharelist;
    assign $U$ to $u_2$, i.e. $\sigma(U) := u_2$;
    remove $U$ from Sharelist;
    for i from 3 to $|\Sigma|+1$ do
        if $u_i$ belongs to some $X \in Minsep$ with $X \cap \{u_2, \cdots, u_{i-1}\} \neq \varnothing$
            then
                pick the smallest $j$ such that $\{u_j, u_i\}$ is contained in some $X \in Minsep$;
                pick the first share $T$ in Sharelist with ($|T \cap \sigma^{-1}(u_j)| = \max_{S \in Sharelist} |S \cap \sigma^{-1}(u_j)|$
                    and $T$ is of maximum size);
                assign $T$ to $u_i$, i.e. $\sigma(T) := u_i$;
                remove $T$ from Sharelist
            else
                if $u_i \in N_u$
                    then
                        pick the first share $T$ from the Sharelist;
                        assign $T$ to $u_i$, i.e. $\sigma(T) := u_i$;
                        remove $T$ from Sharelist
                    else
                        pick the predecessor $u_j$ ($j < i$) of $u_i$ in the BFS-tree;
                        pick the first share $T$ in Sharelist with ($|T \cap \sigma^{-1}(u_j)| = \min_{S \in Sharelist} |S \cap \sigma^{-1}(u_j)|$
                            and T is of maximum size);
                        assign $T$ to $u_i$, i.e. $\sigma(T) := u_i$;
                        remove $T$ from Sharelist
                end_if
        end_if
    end_for
end
```

This algorithm (including the precomputations) is polynomial in n, the number of vertices of the graph.

It can be easily checked that for the examples considered above, the algorithm produces the share assignments with higher reconstruction probability.

5. Conclusion

The presented algorithm for share assignment in communication networks uses node separators of the underlying graphs. This algorithm produces better results than simply placing large shares close to the user node, as is suggested in previous publications. One interesting question for further research is that under which assumptions concerning the underlying graph and set of shares, the heuristic presented here results in an optimal placement of the shares on the nodes.

Acknowledgements

This research was supported by Fachhochschule Südwestfalen—University of Applied Sciences.

References

[1] Blakley, R. (1979) Safeguarding Cryptographic Keys. *Proceedings of the AFIPS* 1979 *National Computer Conference*, **48**, 313-317.

[2] Shamir, A. (1979) How to Share a Secret. *Communications of the ACM*, **22**, 612-613.

http://dx.doi.org/10.1145/359168.359176

[3] Simmons, G.J. (1991) An Introduction to Shared Secret and/or Shared Control Schemes and Their Application. Contemporary Cryptology, IEEE Press, New York, 441-497.

[4] Stinson, D.R. (1992) An Explication of Secret Sharing Schemes. *Designs, Codes, and Cryptography*, **2**, 357-390. http://dx.doi.org/10.1007/BF00125203

[5] Blundo, C., de Santis, A., Stinson, D.R. and Vaccaro, U. (1995) Graph Decomposition and Secret Sharing Schemes. *Journal of Cryptology*, **8**, 39-64. http://dx.doi.org/10.1007/BF00204801

[6] Lee, C.-Y., *et al.* (1999) A Probability Model for Reconstructing Secret Sharing under the Internet Environment. *Information Sciences*, **116**, 109-127. http://dx.doi.org/10.1016/S0020-0255(98)10104-4

[7] Pönitz, A. and Sans, O. (2001) Computing the Reconstruction Probability of Distributed Secret Sharing Schemes Using the Composition Method. (Unpublished Manuscript)

[8] Colbourn, J. (1987) Combinatorics of Network Reliability. Oxford University Press, Oxford.

[9] Shier, D.R. (1991) Network Reliability and Algebraic Structures. Clarendon Press, Oxford.

[10] Bienstock, D. (1988) Some Lattice-Theoretic Tools for Network Reliability Analysis. *Mathematics of Operations Research*, **13**, 467-478. http://dx.doi.org/10.1287/moor.13.3.467

[11] Tittmann, P. and Blechschmidt, A. (1991) Reliability Bounds Based on Network Splitting. *Elektronische Informationsverarbeitung und Kybernetik*, **27**, 317-326.

[12] Poguntke, W. and Simonet, G. (2004) Secrets and Separators. ALGCO—Algorithmes, Graphes et Combinatoire, LIRMM—Laboratoired'Informatique de Robotique et de Microélectronique de Montpellier, Report 04001.

[13] Kloks, T. and Kratsch, D. (2006) Listing All Minimal Separators of a Graph. *Proceedings of the* 11*th Annual Symposium on Theoretical Aspects of Computer Science*, Springer, LNCS 775, 759-768. http://dx.doi.org/10.1137/s009753979427087x

[14] Poguntke, W. (2003) Using Mincuts to Design Secret Sharing Schemes in Graphs. Extended Abstract. *Electronic Notes in Discrete Mathematics*, **13**, 97-102. http://dx.doi.org/10.1016/S1571-0653(04)00447-0