Modeling System Threat Probabilities Using Mixed-Radix Multiple-Valued Logic Decision Diagrams

THEODORE W. MANIKAS[1]*, MITCHELL A. THORNTON[1], DAVID Y. FEINSTEIN[2]

[1] *Department of Computer Science and Engineering, Southern Methodist University, USA*
[2] *Innoventions, Inc., USA*

Received 29 September 2011; In final form 1 August 2012

Design for security has become an area of increasing importance. This includes securing systems from both natural and intentional threats. Extremely large and complex systems can have an accordingly large number of threat scenarios, thus simply listing the threats and devising countermeasures for each is ineffective and inefficient. The components of a large system may also have various states of failure, which cannot be captured by contemporary binary system models. To address this problem, we describe a threat cataloging methodology whereby a large number of threats can be efficiently cataloged and analyzed for common features. This allows countermeasures to be formulated that address a large number of threats that share common features. The methodology utilizes Mixed-Radix Multiple-Valued Logic for describing the state of a large system and a multiple-valued decision diagram (MDD) for the threat catalog and analysis.

Key words: large system security, threat cataloging, mixed-radix, MDD, threat probability analysis.

* email: `manikas@lyle.smu.edu`

1 INTRODUCTION

In recent years, more emphasis has been placed on design for security, security assessment, disaster recovery, and disaster tolerance in addition to ongoing efforts in the established area of fault tolerance. All of these areas require the identification and analysis of faults or threats. Most fault models and analysis methods assume that a fault causes a system to either function or fail and are thus modeled with binary conditions. In [9] we extended the model to allow the system to be modeled with a finite discrete set of states encompassing intermediate conditions of partial failure. Here, we further extend the model to allow for subsystem components to have differing numbers of state values resulting in a mixed-radix MDD representation.

Recent events have demonstrated our vulnerability to disasters, both natural and man-made. This motivates the need to incorporate disaster tolerance into large system designs. Disaster tolerance is a characteristic of a system that results from incorporating a superset of the more established design approaches for fault tolerant systems. Disaster tolerant systems differ from fault tolerant systems since they are designed with the assumption that failures can occur due to threats that can result in either single or massive numbers of individual faults. Furthermore, the faults can occur simultaneously or in a rapidly cascading manner. Traditional fault tolerance system design usually relies on fault models that represent single points of failure for the entire system. In contrast, a disaster-tolerant system can still function with some degree of normality even in the presence of multiple or cascading faults [8, 19].

Threats to a system can occur in either an accidental or intentional manner. Examples of accidental threats include random component failures, natural disasters, and fires. Examples of intentional threats include sabotage, or in the case of information processing systems, cyber threats. It is necessary to catalog and characterize anticipated system threats in order to formulate appropriate countermeasures during the design and implementation of a disaster tolerant system.

A variety of tree-like data structures have been developed to represent possible system threats. These tree structures are commonly called fault trees or attack trees. Fault trees are used to identify the effects of component failures on a system [21] while attack trees are intended to focus on the effects of cyber security breaches [18]. Several different variants of these types of trees have been devised and used interchangeably in past literature [3, 5, 12, 7, 14]. However, these trees are all based on a binary model whereby a system either operates in a fully functional or a complete failure mode. This assumption allows binary-valued trees, connected with Boolean AND and OR operators to be used as basic elements. The binary model limits the effectiveness of how such trees can be used to represent threats in a disaster tolerant system.

Modeling different operational modes other than the binary case of failure or normal operation are critical in analyzing large systems in the presence of threats. As an example, in the 2003 blackout of the US power grid, many complex interactions caused a blackout to occur in a large portion of the northeastern US [1]; however, it would be incorrect to state that the entire US power grid failed. In order to effectively catalog system threats, it is necessary to determine the probabilities of these threats based on various system stimuli, including input conditions and their probabilities.

As part of our preliminary research on large-scale system threat assessment, we developed threat tree models [13]. Threat trees are a superset of fault and attack trees since they are based on multiple-valued (MV) or radix-p valued algebras over a finite and discrete set of values [11]. When the radix $p = 2$, the threat tree reduces to a fault or attack tree depending on the nature of the disruptive events. Generally, threat trees have $p > 2$: these additional logic states allow for more complicated interactions to be modeled. In particular, these additional states can represent partial failures or degraded performance in a system, which are critical in analyzing large systems in the presence of threats.

Multiple-valued logic systems can either be fixed-radix, where all system components have the same radix p, or mixed-radix, where the system components have different radices. A preliminary version of the results in this paper was presented at the 41st IEEE International Symposium on Multiple-Valued Logic [9], and focused on threat probability analysis methods for fixed-radix systems. However, many practical systems may contain components that have different radices. This paper describes the extension of our methods to determine threat probabilities for mixed-radix systems.

The structure of this paper is the following: First, background information on decision diagram models is provided. Next, an approach is presented using decision diagram models to determine system threat probabilities. Finally, a system example is used to illustrate these concepts.

2 DECISION DIAGRAMS

Figure 1 shows an example of a simple binary fault tree where the circular nodes represent the event of a single component failure and the logic operators (AND/OR gates) show how the events combine to result in a subsystem

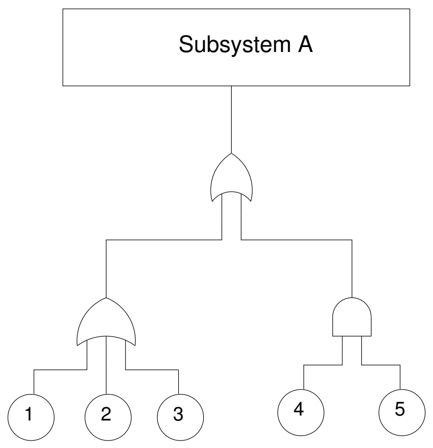

FIGURE 1
Fault Tree Example

failure. Referring to Figure 1, if one or more of events 1, 2, or 3 occur, subsystem A will fail. Alternatively, both events 4 and 5 must occur for subsystem A to fail. For practical systems, it is likely that a large number of components will need to be analyzed. As the system size grows (in terms of number of components and/or number of logic states), the system analysis process quickly becomes unwieldy. Therefore, an alternate system representation is required to make the analysis process more efficient.

Decision diagrams are well-suited for compact representation of a large number of threats and due to their canonical structure, efficient algorithms are formulated that analyze threats and identify those that pose the greatest threat to the system. Decision diagrams are rooted directed acyclic graphs (DAG) that can be used to represent large switching functions in an efficient manner. For binary-valued logic, the binary decision diagram (BDD) is a well-known structure [4] that has been applied to many areas including the representation of fault trees [17, 16, 15, 2, 22, 23]. Furthermore, efficient software is readily available to manipulate BDDs [16, 15, 2].

In the case of Multiple-Valued Logic (MVL), an extension to the BDD construct has been developed and implemented called the Multiple-Valued Decision Diagram (MDD). Consider a totally-specified p-valued function with n inputs, $f(x_0, x_1, \ldots, x_n)$ where each dependent variable $x_i \in \{0, 1, \cdots, p-$

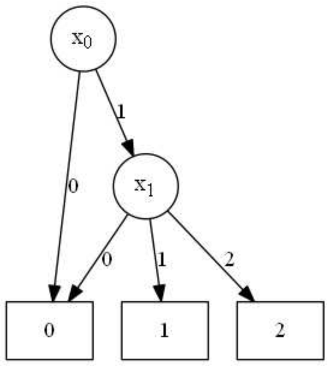

FIGURE 2
Mixed-radix MDD Example

$1\}$. $f(x)$ can be efficiently represented by an MDD. As is similar to BDD, the MDD is also a DAG and it contains a maximum of p terminal nodes, where each terminal node is labeled by a distinct logic value in the range $[0, p-1]$. Every non-terminal node is labeled by an input variable, and has p outgoing edges, where each edge corresponds to each logic value. MDD can be minimized using various techniques that were developed for BDD, thus allowing the representation of exceptionally large number of such functions [10].

For mixed-radix MVL, each non-terminal node on the MDD will have its own radix. Given a mixed-radix MDD with m non-terminal nodes, each node j has an associated radix p_j. The maximum radix of the MDD is denoted by $|p| = max\{p_1, p_2, \ldots, p_m\}$, and the MDD contains a maximum of $|p|$ terminal nodes [20]. Figure 2 shows an example of an MDD for a mixed-radix system. Input variable x_0 is radix-2, while input variable x_1 is radix-3. The truth table for this system is shown in Table 1. This is mixed-radix "min" function, where $f = min(x_0, x_1)$. For a mixed-radix *min* function, if both inputs are their maximum radix values, then the output is the maximum radix value of the system. Therefore $f = 2$ if $x_0 = 1$ and $x_1 = 2$.

3 DETERMINING SYSTEM OUTPUT PROBABILITIES

For binary tree structures (fault and attack trees), BDD's have been applied to simplify the modeling threats on complex systems [5, 17, 16, 15, 2, 22,

x_0	x_1	f
0	0	0
0	1	0
0	2	0
1	0	0
1	1	1
1	2	2

TABLE 1
Truth Table for Mixed-radix Example

23]. The common approach to determine probabilities on binary tree structures is to apply probability equations, such as those described in [12]. Assume we have n inputs, each with probability $P_{in}(x_i)\{1\}, i \in [1, n]$, where $P_{in}(x_i)\{1\}$ = probability that input x_i is 1. If the inputs are mutually independent, then the output probability (probability that the output is 1) of an n-input AND gate is the sum of all the edges pointing to the terminal-1 vertex.

As an example, assume we have a 2-input AND gate with the following input probabilities:

Input $x_0 : P_{x_0}[x_0 = 0] = 0.5, P_{x_0}[x_0 = 1] = 0.5$

Input $x_1 : P_{x_1}[x_1 = 0] = 0.25, P_{x_1}[x_1 = 1] = 0.75$

Then $P_{out}\{1\} = P_{x_0}[x_0 = 1]P_{x_1}[x_1 = 1] = (0.5)(0.75) = 0.375$ Since this is a binary system, $P_{out}\{0\} = 1 - P_{out}\{1\} = 1 - 0.375 = 0.625$.

While the separate application of probability equations has been commonly used for previous fault tree assessment methods, this approach can become inefficient for systems with large number of nodes. Since the decision diagram contains edges with logic value weights, a more efficient approach would be to incorporate probabilities as edge weights, then calculate output probabilities by traversing the graph edges.

The BDD for this gate is shown in Figure 3. The edge weights have the notation "V/P", where V = input value and P = probability of that input value. The output state probabilities $P_{out}\{1\}$ and $P_{out}\{0\}$ are calculated by traversing the edges of the BDD. This is done along all the paths from the root node to the output state nodes (0 and 1). The probabilities along each path are multiplied, and paths leading to the same output state node are added:

$P_{out}\{0\} = 0.5 + (0.25)(0.5) = 0.625$

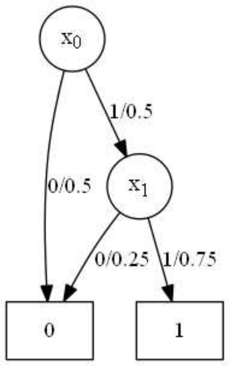

FIGURE 3
Binary BDD with Probabilities for AND Gate

$P_{out}\{1\} = (0.5)(0.75) = 0.375$

Since an MDD is an extension of a BDD, it should also hold that we can determine threat probabilities by applying the specified state probabilities as weights to the MDD edges and traversing the tree. Table 2 shows a system with three operational states and two components with given input probabilities. The system has the following truth table (Table 3), which becomes the MDD shown in Figure 4. Note that branch (1,2) from A to output state (2) adds the probabilities of the two events: 0.25 + 0.5 = 0.75. We get the following output probabilities by traversing the MDD. This is done along all the paths from the root node to the output state nodes (0, 1, and 2). The probabilities along each path are multiplied, and paths leading to the same output state node are added together:

$P_{out}\{0\} = (0.2)(0.25) = 0.05$

$P_{out}\{1\} = (0.2)(0.5) + (0.3)(0.25) = 0.175$

$P_{out}\{2\} = (0.2)(0.25) + (0.3)(0.75) + 0.5 = 0.775$

Using these concepts, we developed the algorithm shown in Figure 5 to calculate system output probabilities for a given MDD. Initially, the output probabilities are set to zero. The algorithm starts at the root vertex of the MDD, then traverses the MDD using a depth-first search. A standard depth-first search on a graph with V vertices and E edges has a complexity of $\mathcal{O}(V + E)$ [6]. Each edge is traversed, and the corresponding probabilities are multiplied to form a working probability value until a terminal node is

State	Component A Probability	Component B Probability
(2) Operational	0.25	0.5
(1) Degraded	0.5	0.3
(0) Offline	0.25	0.2

TABLE 2
Radix-3 System Example

A	B	f
0	0	0
0	1	1
0	2	2
1	0	1
1	1	2
1	2	2
2	X	2

TABLE 3
Truth Table for Radix-3 Example

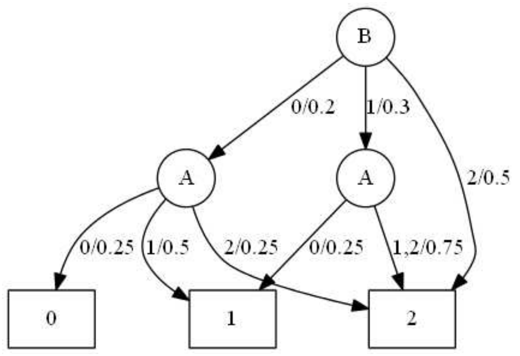

FIGURE 4
MDD for Radix-3 Example

reached. At this point, the output probability corresponding to the terminal node is updated by adding the working probability value. When all vertices have been traversed, the output probabilities have been updated to their final values.

4 EXAMPLE

Using the previously mentioned 2003 US power grid failure example, assume that we can represent a simple power system using the example from [13]. Figure 6 shows the MVL circuit for the power system example. The city's power condition P depends on two sub-systems:

1. Power generation plant sub-system G, which is a radix-3 system.
2. Power transmission line sub-system T, which is a radix-2 system.

System P is mixed-radix, so a mixed-radix MDD is developed to model this system. The analysis proceeds in a bottom-up manner: the output logic levels and probabilities of sub-systems G and T will become the input logic levels and probabilities for the MIN gate.

4.1 Power generation plant sub-system

The power generation plant sub-system G has three operational states: fully operational, partially operational, and non-operational. This is a radix-3 system, so it cannot be represented by the traditional attack tree structure. However, it can be represented by our threat tree structure: state 2 = fully operational, state 1 = partially operational (degraded) and state 0 = non-operational. Also assume that the simple power grid system has three generation plants: coal, hydro, and wind. The total power available on the power grid is the sum of the power output produced by the coal, hydro and wind plants. Table 4 shows the threshold power values for fully operational, degraded, and non-operational states for the total power grid, while Table 5 shows these values for each generation plant. Using this information, an MVL truth table for this system is created (Table 6), where f represents the total power grid operational state. Note that "X" indicates a "don't-care" state, where the value can be either 0, 1, or 2. For example, row 4 in Table 6 identifies the system state where the coal plant is non-operational (state 0: output = 0), the hydro plant is partially operational (state 1: output = 1500 MW) and the wind plant is fully operational (state 2: output = 200 MV). The total power grid output for this system state is 1700 MW, so the power grid is partially operational ($G = 1$).

Given MDD as graph **G(V,E)** with vertices **V** and edges **E**:

```
begin Calc_Prob(G)
    let p = 1.0    // current probability value
    for j = 0 to (radix-1) do
        $P_{out}[j] = 0.0$    // output probability values
    end for
    let u = root vertex of G
    DFS(u,p)
end Calc_Prob


begin DFS(u,p)
    if u is a terminal node then
        k = value of terminal node $(0, 1, \ldots, radix - 1)$
        // update output probability value
        $P_{out}[k] = P_{out}[k] + p$
    else
        for each child vertex v of u do
            x = probability value on edge(u,v)
            // multiply edge probability by current probability
            $y = x \cdot p$
            // do depth-first search starting at child v
            DFS(v,y)
        end for
    end if
end DFS
```

FIGURE 5
Algorithm for Output Probability Calculations

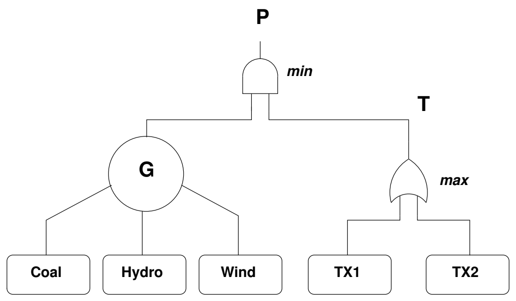

FIGURE 6
Power System Example with Power Generation and Transmission Line Systems [13]

State	Total Power Output Available (MW)
(2) Fully Operational	≥ 2400
(1) Degraded	1600 - 2400
(0) Non-operational	< 1600

TABLE 4
Operational States for Power Grid Subsystem

State	Coal Plant		Hydro Plant		Wind Plant	
	MW	Probability	MW	Probability	MW	Probability
(2)	1000	0.98	2000	0.99	200	0.6
(1)	600	0.018	1500	0.009	100	0.3
(0)	0	0.002	0	0.001	0	0.1

TABLE 5
Operational States for Each Power Plant (column *MW* indicates output in MW)

Coal	Hydro	Wind	G
0	0	X	0
0	1	0	0
0	1	1	1
0	*1*	*2*	*1*
0	2	X	1
1	0	X	0
1	1	X	1
1	2	X	2
2	0	X	0
2	1	X	2
2	2	X	2

TABLE 6
Truth Table for Power Grid Subsystem

The MDD for our example power grid system is shown in Figure 7. The path along the MDD that corresponds to row 4 in Table 6 is indicated by the dashed arrows in Figure 7.

Using this MDD, we can now calculate the output probabilities for this radix-3 system. Note that the input probabilities from Table 5 are denoted in the MDD of Figure 7. Applying the algorithm of Figure 5, we obtain the following output probabilities:

$P_{out}\{0\} = (0.002)(0.009)(0.1) + (0.002)(0.001) + (0.018)(0.001) + (0.98)(0.001) = 0.001002$

$P_{out}\{1\} = (0.002)(0.009)(0.3) + (0.002)(0.009)(0.6) + (0.018)(0.009) + (0.002)(0.99) = 0.002158$

$P_{out}\{2\} = (0.98)(0.009) + (0.018)(0.99) + (0.98)(0.99) = 0.99684$

4.2 Power transmission line sub-system

The power transmission line sub-system T has two states: either fully operational or non-operational. Therefore, this is a radix-2 system. As shown in Figure 6, $T = max\{TX_1, TX_2\}$. This means that if at least one line is fully operational, system T is fully operational. Figure 8 shows the MDD for this system. For this example, we assume that line TX_1 has a probability of 0.9 operational and 0.1 non-operational, while the respective values for line TX_2 are 0.95 and 0.05. The resulting output probabilities are:

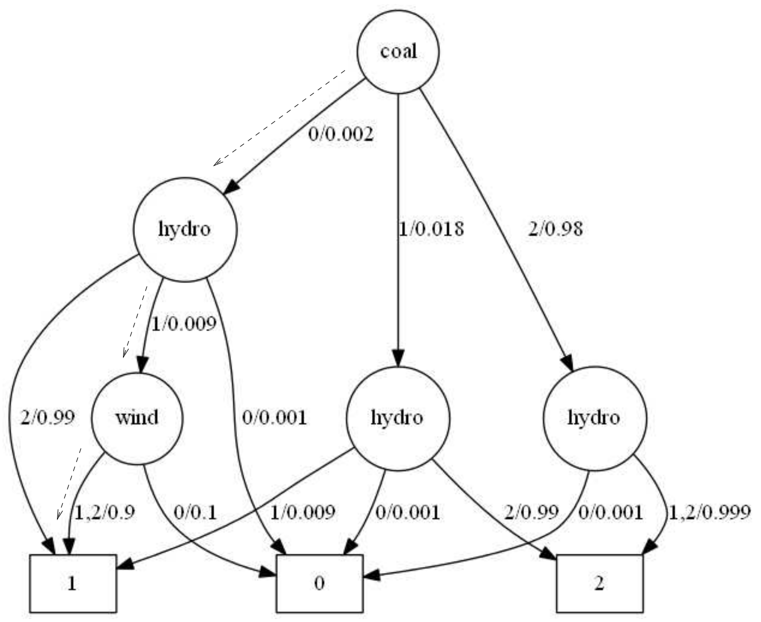

FIGURE 7
MDD Radix-3 for the Power Grid Subsystem

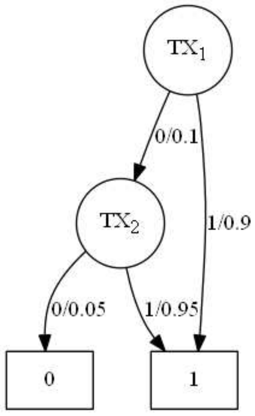

FIGURE 8
MDD Radix-2 for the Power Transmission Line Subsystem

$P_{out}\{0\} = (0.1)(0.05) = 0.005$
$P_{out}\{1\} = (0.1)(0.95) + (1)(0.9) = 0.995$

4.3 Entire power grid system

Now that we have the output probabilities of the subsystems of the power grid, these will be used as input probabilities to determine the output probabilities of the entire power grid system P. Recall from Figure 6 that the city's power condition $P = min\{G, T\}$. Sub-system T is a radix-2 system, while sub-system G is a radix-3 system. Therefore, P must be a radix-3 system with the *m*in function operation as described in Section 2. Figure 9 shows the mixed-radix MDD for system P. Using the algorithm of Figure 5, the resulting output probabilities are:

$P_{out}\{0\} = (0.005)(1) + (0.995)(0.001) = 0.005995$
$P_{out}\{1\} = (0.995)(0.0022) = 0.002189$
$P_{out}\{2\} = (0.995)(0.9918) = 0.991816$

5 CONCLUSION

The determination of system threat probabilities is an important component of system classification and threat cataloging. We have shown how to model system threat probabilities using edge weights on MDD's, which is a more

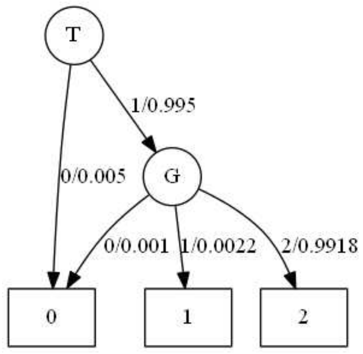

FIGURE 9
Mixed-radix MDD for Power Grid System

efficient approach than the current method of developing separate probability equations for each possible output threat scenario. It allows easier determination of overall system state probabilities and it can accommodate complex systems with the efficient scalability of modern MDD packages. The framework discussed in this paper can be further applied to risk analysis which is useful in the determination of the initial system element probability values.

In terms of augmenting large systems to make them more disaster tolerant, we intend to develop further analysis methods based on threat trees. Because threat trees inherently combine common subtree structures, we note that a tree representing a collection of threats automatically yields common characteristics of subsets of threat by combining common subpaths. Such common subpaths correspond to similar characteristics among subsets of threats and they may be used to determine which parts of the system to augment in order to address a maximum number of threats. Finally, another area for further research is how to model threat probabilities that are not mutually exclusive, such as conditional probabilities, and determine how to incorporate these into the MDD structure.

6 ACKNOWLEDGMENT

This research was partially supported by the U.S. Office of Naval Research (ONR) project N000140910784.

7 NOMENCLATURE

p_j	radix of node j in MDD
$\|p\|$	maximum radix value in mixed-radix MDD
$P_{out}\{x\}$	probability that output value of function will be x

REFERENCES

[1] G. Andersson, P. Donalek, R. Farmer, N. Hatziargyriou, I. Kamwa, P. Kundur, N. Martins, J. Paserba, P. Pourbeik, J. Sanchez-Gasca, R. Schulz, A. Stankovic, C. Taylor, and V. Vittal. (2005). Causes of the 2003 major grid blackouts in north america and europe, and recommended means to improve system dynamic performance. *IEEE Transactions on Power Systems*, 20(4):1922–1928.

[2] L. M Bartlett and J. D Andrews. (September 2002). Choosing a heuristic for the ”fault tree to binary decision diagram” conversion, using neural networks. *IEEE Transactions on Reliability*, 51(3):344 – 349.

[3] Stefano Bistarelli, Pamela Peretti, and Irina Trubitsyna. (2008). Analyzing security scenarios using defence trees and answer set programming. *Electronic Notes in Theoretical Computer Science*, 197(2):121 – 129.

[4] R. E Bryant. (August 1986). Graph-Based algorithms for boolean function manipulation. *IEEE Transactions onComputers*, C-35(8):677 –691.

[5] S. Contini, G. G.M Cojazzi, and G. Renda. (2008). On the use of non-coherent fault trees in safety and security studies. *Reliability Engineering and System Safety*, 93(12):1886 – 1895.

[6] Thomas H Cormen, Charles E Leiserson, and Ronald L Rivest. (1990). *Introduction to algorithms*. MIT Press.

[7] Lars Grunske and David Joyce. (2008). Quantitative risk-based security prediction for component-based systems with explicitly modeled attack profiles. *Journal of Systems and Software*, 81(8):1327 – 1345.

[8] M. A Harper, C. M Lawler, and M. A Thornton. (2005). IT application downtime, executive visibility and disaster tolerant computing. In *Proc. Int. Conf. on Cybernetics and Information Technologies, Systems and Applications (CITSA 2005), and Int. Conf. on Information Systems Analysis and Synthesis (ISAS)*, pages 165–170.

[9] T. W Manikas, M. A Thornton, and D. Y Feinstein. (May 2011). Using Multiple-Valued logic decision diagrams to model system threat probabilities. In *2011 41st IEEE International Symposium on Multiple-Valued Logic (ISMVL)*, pages 263 –267.

[10] D. M Miller and R. Drechsler. (May 1998). Implementing a multiple-valued decision diagram package. In *Proceedings. 1998 28th IEEE International Symposium on Multiple-Valued Logic*, pages 52 –57.

[11] D. M Miller and M. A Thornton. (2007). *Multiple Valued Logic: Concepts and Representations*. Morgan & Claypool Publishers.

[12] Igor Nai Fovino, Marcelo Masera, and Alessio De Cian. (2009). Integrating cyber attacks within fault trees. *Reliability Engineering and System Safety*, 94(9):1394 – 1402.

[13] P. Ongsakorn, K. Turney, M. Thornton, S. Nair, S. Szygenda, and T. Manikas. (April 2010). Cyber threat trees for large system threat cataloging and analysis. In *2010 4th Annual IEEE Systems Conference*, pages 610 –615.

[14] Andreas L Opdahl and Guttorm Sindre. (2009). Experimental comparison of attack trees and misuse cases for security threat identification. *Information and Software Technology*, 51(5):916 – 932.

[15] A. B. Rauzy, J Gauthier, and X Leduc. (2007). Assessment of large automatically generated fault trees by means of binary decision diagrams. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, 221(2):95–105.

[16] Karen A. Reay and John D. Andrews. (2002). A fault tree analysis strategy using binary decision diagrams. *Reliability Engineering and System Safety*, 78(1):45 – 56.

[17] R. Remenyte and J. D Andrews. (2006). A simple component connection approach for fault tree conversion to binary decision diagram. In *Proceedings - First International Conference on Availability, Reliability and Security, ARES 2006*, volume 2006, pages 449 – 456, Vienna, Austria.

[18] Bruce Schneier. (1999). Attack trees. *Dr. Dobb's Journal*, 24(12):21 – 21.

[19] S.A. Szygenda and M. A Thornton. (2005). Disaster tolerant computing and communications. In *Proc. Int. Conf. on Cybernetics and Information Technologies, Systems and Applications (CITSA 2005), and Int. Conf. on Information Systems Analysis and Synthesis (ISAS)*, pages 171–173.

[20] M. A. Thornton. (2004). Mixed-radix MVL function spectral and decision diagram representation. *Autom. Remote Control*, 65(6):1007–1017.

[21] W. E Vesely, F. F Goldberg, N. H Roberts, and D. F Haasi. (January 1981). Fault tree handbook. Technical Report NUREG-0492, U.S. Nuclear Regulatory Commission.

[22] Liudong Xing and Yuanshun Dai. (September 2009). A new Decision-Diagram- based method for efficient analysis on multistate systems. *IEEE Transactions on Dependable and Secure Computing*, 6(3):161 –174.

[23] Olexandr Yevkin. (2009). Truncation approach with the decomposition method for system reliability analysis. In *2009 - Annual Reliability and Maintainability Symposium, RAMS 2009, January 26, 2009 - January 29, 2009*, Proceedings - Annual Reliability and Maintainability Symposium, pages 430–435, Fort Worth, TX, United states. Institute of Electrical and Electronics Engineers Inc.