Efficiently computing the likelihoods of cyclically interdependent risk scenarios

Steve Muller, Carlo Harpes, Yves Le Traon, Sylvain Gombault, Jean-Marie Bonnin

Efficiently computing the likelihoods of cyclically interdependent risk scenarios

Steve Muller[a,b,c,*], Carlo Harpes[a], Yves Le Traon[b], Sylvain Gombault[c], Jean-Marie Bonnin[c]

[a] *itrust consulting s.à r.l., Luxembourg*
[b] *University of Luxembourg, Luxembourg*
[c] *Telecom Bretagne, Luxembourg*

Quantitative risk assessment provides a holistic view of risk in an organisation, which is, however, often biased by the fact that risk shared by several assets is encoded multiple times in a risk analysis. An apparent solution to this issue is to take all dependencies between assets into consideration when building a risk model. However, existing approaches rarely support cyclic dependencies, although assets that mutually rely on each other are encountered in many organisations, notably in critical infrastructures. To the best of our knowledge, no author has provided a *provably* efficient algorithm (in terms of the execution time) for computing the risk in such an organisation, notwithstanding that some heuristics exist.

This paper introduces the dependency-aware root cause (DARC) model, which is able to compute the risk resulting from a collection of root causes using a poly-time randomised algorithm, and concludes with a discussion on real-time risk monitoring, which DARC supports by design.

1. Introduction

Risk management constitutes an important aspect of decision taking, especially if the outcome is uncertain or has a large-scale impact on an organisation, which is why it forms a basis for many information security standards, including ISO/IEC 27xxx (International Organization for Standardization, 2014). Risks can be evaluated in two ways (Laboratory, 2012): qualitatively and quantitatively. Furthermore, some authors have suggested combining (Mangan et al., 2004) or converting (Cox et al., 2005) both methods to get better results, but this topic is beyond the scope of this paper.

In a *qualitative* assessment, risk scenarios are identified and then estimated in terms of probability and impact on a discrete (and often abstract) scale, which consists of some few values, such as "low", "normal" and "high". A previously defined set of *unacceptable tuples* $\langle \text{probability}, \text{impact} \rangle$ permits to distinguish between risk scenarios for which counter-measures need to be implemented (so as to reduce risk) and scenarios that are critical for an organisation (see Fig. 1).

In contrast, *quantitative* assessments express risk in terms of the potential damage that is inflicted to an organisation, e.g. in financial terms. So instead of qualifying a risk scenario as above, its likelihood and impact are expressed numerically; for example, by stating that "scenario X is estimated to occur every

* *Corresponding author.*
E-mail addresses: steve.muller@itrust.lu (S. Muller), harpes@itrust.lu (C. Harpes), yves.letraon@uni.lu (Y. Le Traon), sylvain.gombault@telecom-bretagne.eu (S. Gombault), jm.bonnin@telecom-bretagne.eu (J.-M. Bonnin).

Freq.	Impact			
	low	normal	high	critical
very often				
often	✓			
normal	✓	✓		
rarely	✓	✓	✓	

Fig. 1 – Sample table that can be used in a qualitative risk analysis. White cells denote acceptable, black cells unacceptable risk scenarios.

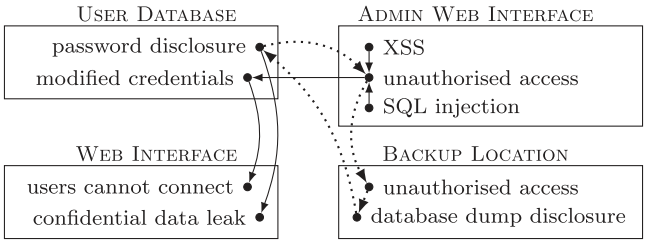

Fig. 3 – Illustration of cyclic dependencies of a poorly designed web service.

5 years and when it does, it causes a loss of 10000 Euro", leading to an expected loss of 2000 Euro per year. Note that unlike above, quantitative risk analyses provide an integral view of risk faced by an organisation since all scenarios can be inspected and compared to one another at once, thanks to the numerical value of the expected losses. By consequence, the urgency of securing a specific asset in favour of another one, can be readily deduced, which is not so obvious to achieve in qualitative analyses.

There is a major drawback of many risk assessment methods regarding their support for interdependent assets. For instance, in the case of a server hosting a critical service, the efficiency of the responsible software is not only threatened on the one hand by vulnerabilities (bugs, security flaws) of the *service* itself, but also, on the other hand, by any down-time of the *server*. This relationship can be accounted for in several ways, the flaws and strengths of which are summarised in Fig. 2:

One can analyse the common risk scenario ("service not provided") for each of the assets ("server is down", "software has vulnerabilities"); however, the risk scenario in question is going to be accounted for multiple times in the global risk analysis, the outcome of which thereby becomes distorted.

Another option is to eliminate any redundancy from the risk assessment, for instance by assigning each scenario only to the most related asset. Now the global risk analysis features every risk scenario exactly once, but the view on an individual asset is no longer complete, since it does not take care of every possible risk scenario.

Finally, several authors (Brændeland et al., 2010; Breier, 2014; Pederson et al., 2006; Tong and Ban, 2014; Utne et al., 2011; Wang et al., 2008) incorporate the assets and/or risk scenarios together with their interdependencies into a hierarchical graph, based on which they deduce the risk for an asset, a group of assets or the whole organisation by reading off all subordinate risk scenarios. However, hardly any risk assessment model supports cycles in the dependency chain (e.g. A depends on B depends on C depends on A), although such cycles exist in every (sub-) system where the compromise of one component affects the whole (sub-) system. This is especially true in the context of Industrial Control Systems (ICS), where cascading effects can be of devastating order; for instance, the power grid and a voltage control system (requiring electricity to work) constitute an example of interdependent assets. As a second illustration, consider a poorly designed web service hosting confidential and valuable data (e.g. medical information) where the administrator can change any user passwords and can retrieve any of the regularly made backups of the user account database; then unauthorised access to the administrator interface allows an attacker to fetch a back-up, read out and disclose the user passwords, which again leads to unauthorised access. This scenario is depicted in Fig. 3: note the cycle "admin interface – backup location – user database" (dotted lines).

	global risk analysis	individual assets
complete scenario list for each asset	biased outcome	✓
redundancies removed	✓	incomplete view
dependency model	✓	✓

Fig. 2 – Flaws and strengths of the several risk assessment models. Check marks (✓) indicate correct outcome.

Some authors (Homer et al., 2009; Wang et al., 2008) propose a solution to deal with cyclic dependency graphs using graph unfolding techniques, but they fail at providing a complexity analysis for their methods.

This paper introduces a novel approach for computing the risk faced by cyclically dependent assets. Indeed, the proposed algorithm is based on a randomised (non-deterministic) simulation and – in contrast to other algorithms – provably efficient in terms of their execution time (which is important when doing real-time risk monitoring).

Section 2 presents related papers dealing with (cyclic) dependencies in risk assessments. The risk model used by the algorithm is defined in Section 3, along with the algorithm itself, whereas the proof of its correctness and running time can be found in the Appendix. The conducted experiments and their results are exposed in Section 4, Section 5 deliberates a generalisation of the model to also support more complex dependencies, and the paper closes with a conclusion in Section 6.

2. Related work

The model presented in this paper combines the concepts developed by various authors.

Breier (2014) encodes information security assets and their dependencies in a directed graph, where edges denote causal relations between nodes. The model supports the use of logical AND (for assets that depend on all parents) and OR (for assets that depend on one of the parents) operations. McQueen et al. (2006) also use attack graphs to encode dependencies (in their case, between services in a SCADA environment) and express risk as the time needed to compromise a component. Utne et al. (2011) focus on cascading effects in critical infrastructures by analysing the interdependencies between several high-level services (such as electricity) and their impact to the society. They also provide a framework to quantise the several magnitudes involved in the risk assessment, allowing an explicit computation of risk. Most interestingly, the risk itself is expressed as expected number of people affected by an incident.

Similarly, the risk assessment methodology supported by the Spanish government, *MAGERIT* (Amutio et al.), also deals with asset dependencies by embedding them into a graph. However, it does not directly link related assets, but their security objectives (such as confidentiality, integrity and availability), whenever they have an impact on each other. Rahmad et al. (2012) aim at improving on *MAGERIT* by combining it with thoughts from Fenz et al. (2009). They use an exhaustive list of threat scenarios instead of security objectives, which considerably increases the size of the model. This paper further generalises this concept to arbitrary and user-definable security incidents, which shows that quantitative assessments do not necessarily have to be expressed in financial terms.

To structure the assets (and thus the risk analysis) in a sensible way, Aubigny et al. (2011) establish a risk ontology for highly interdependent (critical) infrastructures, based on the estimation of quality of service (QoS). The proposed model supports risk prediction and incorporates a data structure which allows QoS information to be shared among interconnected infrastructures. In a similar spirit, Xin and Xiaofang (2014) classify assets into three layers, namely business, information and system. On the lowest layer, risk is computed traditionally as $\mathrm{risk} = \mathrm{impact} \times \mathrm{likelihood}$. Dependencies appear in the model as weighted impact added to the risk of dependent higher-level assets.

Moreover, in order to reduce the workload on risk assessors, several authors suggest models where risk can be computed from little input data. For instance, Homer et al. (2009) adapt the concepts and algorithms known from Bayesian networks to the realm of risk assessments and apply a graph unfolding technique to generalise the model for cyclic dependency graphs. The running time of this process is exponentially large in general, though, and thus only works for small or sparse graphs.

As a workaround, Wang et al. (2008) provide a simplified (and efficiently computable) probability metric for Bayesian networks, which they generalise to cyclic graphs, as well. However, the chosen metric does not properly take dependent events into account, so that the computed probabilities do not reflect reality.

Although Baiardi and Sgandurra (2013) work on attack trees rather than on causal graphs, the randomised algorithm they provide can be generalised in such a way that the probabilities of arbitrary causal graphs can be efficiently computed – which is done in this paper.

3. Modelling the risk analysis context

3.1. Risk assessment

The *context* of a risk analysis is primarily determined by the set of *assets*, which can be virtually anything of value to a company or institution. In terms of risk, each asset is characterised by its impact on business when one of its (security-related) properties can no longer be guaranteed. Such properties are called *security aspects* in the following, and include most notably the three notions (Bishop, 2012) below. Note that on the one hand, some of these aspects might not be applicable to certain assets. On the other hand, it is sometimes sensible to add further properties, or to be more specific about existing ones – especially if the impact considerably changes when doing so. For instance, one usually wants to distinguish between *temporary* unavailability, causing business interruption, and *permanent* loss, which may be fatal to business.

- Confidentiality – the assumption that sensible information is known only to a well-defined group of people.
- Integrity – the state that an asset is guaranteed to remain in a well-defined state.
- Availability – the property that an asset can be accessed in the way that was previously defined.

The impact itself is expressed in financial terms; this approach has the notable advantage that estimates have an objective meaning and can thus be easily compared to each other. More precisely, the impact is defined to be the financial damage caused by the threatened security aspect, or the amount of money necessary to recover back to the original state. That way, one can compute the *loss expectancy (LE)* which represents the total losses to be expected in a given period, typically a year:

$$\mathrm{LE} = \sum_{s:\mathrm{riskscenario}} \mathrm{impact}(s) \cdot \mathrm{likelihood}(s), \tag{1}$$

where $\mathrm{likelihood}(s)$ denotes the number of times that scenario s is expected to occur in the given period (i.e., the expected frequency) – quantity usually to be estimated by a risk assessor, a methodology or a tool. This paper will present an algorithm for efficiently computing the likelihood function; see Section 3.5.

3.2. Dependencies

A considerable flaw in many risk management models is the lack of understanding of asset dependencies. Indeed, consider a hard disk hosting valuable data (and suppose, for the sake of the example, that no back-up is available), then a hard disk failure does not only require the physical disk to be replaced (which is cheap), but also implies the complete loss of core data (which may be business-ending). By consequence, dependency-unaware models do not give disk health monitoring the attention it deserves.

In fact, asset dependencies represent nothing else than relations of cause and consequence of security incidents on given assets. Causal graphs are thus a natural candidate for encoding

Magnitude	Unit	Description
Impact	Euro	Damage faced if threat occurs
Likelihood	1/year	Expected occurrence per year
LE	Euro/year	Expected loss per year

Fig. 4 – Table summarising the notions involved in the assessment of risk.

these relationships in a mathematical model. Recall that a causal graph on a vertex set of events is a directed graph such that two incidents are linked whenever they cause one another.

3.3. *Likelihoods and probabilities*

Whereas many approaches found in the literature (e.g. Breier, 2014; Liu et al., 2011; Loloei et al., 2012) use an abstract scale (like "low", "medium", "high") for describing the likelihood of an event, this paper relies on concrete physical magnitudes that support the direct use in a computation, see Fig. 4.

A reason why one prefers to use an abstract scale over a number is because precise values are rarely known, so only a rough estimate can be given. However, to make the assessment task easier, one can still restrict to a given set of discrete values to choose from *for orientation*, and interpolate to express slight nuances. One such possible mapping is given in Fig. 5, but can (and has to) be adapted to the setting in question.

Note the fundamental difference between "likelihood" and "probability". Probabilities are only meaningful when characterising a random event *a priori*: the event either happens or it does not with a certain chance. Likelihoods, however, express the *a posteriori* statistical occurrence (or expected frequency) of events over time. Mathematically, probabilities are unit-less and lie within [0,1], whereas likelihoods can be arbitrarily large and have as unit $\frac{1}{\text{time}}$.

Both notions of "probability" and "likelihood" can be easily confused, because they are somewhat related. It is meaningless to say that a risk scenario occurs with a certain probability, though. Probabilities do not scale linearly and are bounded by 100%, which is not in line with the common understanding of risk. Indeed, consider the statement "there is a 10% chance of fire", then how to encode events that occur 100 times more often. If, however, one estimates the likelihood of fire to be "on average once every 10 years", then the expected damage is trivially $10^5\,\text{EUR} \cdot \frac{1}{10\text{y}} = 10000\,\text{EUR/y}$.

level	likelihood	/year
very low	every 30 years	0.0333
low	every 10 years	0.1
moderate	every 3 years	0.333
high	once a year	1
very high	once a month	12

Fig. 5 – One possible mapping of an abstract scale to a concrete likelihood.

The model introduced in this paper makes use of both notions, that is, on the one hand, the *likelihood* of an incident or risk scenario, and on the other hand, the *probability* that it *entails* another (dependent) incident.

In order to distinguish between the two, write $\mathbb{P}$ for the probability measure and $\mathbb{L}$ for the likelihood.

3.4. *The dependency-aware root cause (DARC) model*

Pick a set of nodes V, each representing a security incident. A possible choice is to opt for $V \subseteq \mathcal{A} \times \mathcal{S}$, where $\mathcal{A}$ is the set of all assets and $\mathcal{S}$ the set of security properties. For instance, $S := \{C, I, A\}$ could comprise confidentiality, integrity and availability threat scenarios. In that case, for readability, write $a.s$ instead of $(a, s) \in \mathcal{A} \times \mathcal{S}$ to denote the security aspect s of an asset a.

Let $E \subseteq V \times V$ be the set of (directed) edges such that $(\alpha, \beta) \in E$ iff security incident α has an impact on incident β. For readability, write $\alpha \to \beta$ instead of (α, β) if the set of edges E is understood from the context. Illustrating the notation, the example above can be rewritten in a very short and intuitive form:

$$HDD \cdot A \to DATA \cdot A,$$

which reads "if the availability of the hard-drive is compromised, then so is the availability of any data stored on it".

The tuple (V, E) constitutes a directed graph which is henceforth referred to as the *(asset) dependency graph*. It is not required to be acyclic.

Whereas manual estimation of the full probability distribution of the several incidents is theoretically possible, statistical experiments in real-world systems are usually infeasible, for it would require the simulation of threats on business processes. Instead, the proposed model will use a slightly simplified approach by basing itself on estimating the probability that a particular incident *causes* another, independently of possible other causes. In particular, the parent causes of an event are related by a boolean OR operation, but the model can be extended to support arbitrary boolean formulas as well; see Section 5. Formally, this describes a mapping $p : E \to [0, 1]$ where $p(\alpha \to \beta)$ denotes the probability that β is entailed by α. Note that this is *not* the same as $\mathbb{P}[\beta|\alpha]$, since β could also be entailed by other events. $p(\alpha \to \beta)$ is often denoted $\mathbb{P}[\beta|\text{do}(\alpha)]$ or $\mathbb{P}[\beta|\text{set}(\alpha)]$ in the literature (Pearl, 2000).

Define a *root cause* to be a vertex without parent nodes in the dependency graph. These are the causes for which an explicit likelihood needs to be specified later on, whereas for non-root nodes it is deduced from the model.

An example of a dependency graph is depicted in Fig. 6: the edge weights represent the values of p, and A and I stand for the availability and integrity properties of the assets, respectively. Note how the model does not make a difference between external threats (circled) and security properties of assets (boxed).

3.5. *Computing the probability distribution*

The complexity of the DARC model lies in the fact that one is interested in the *probability* that a certain sequence of events

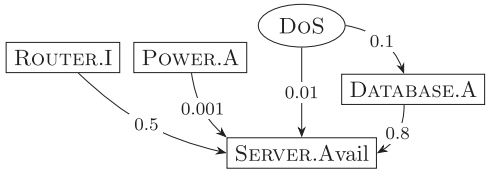

Fig. 6 – An example illustrating the representation of the DARC model as a graph.

cause each other, which is different from the problem of finding such a sequence. Indeed, for the former, one needs to determine *all* such sequences and compute the probability that *one* of them occurs. For this, simply listing those sequences and adding up their probabilities of occurrence yields wrong results, since some edges are accounted for twice.

3.5.1. *The acyclic case*

If the dependency graph is acyclic, a well-established theory can be used to describe the full probability distribution. Indeed, by its definition, the dependency graph together with the associated probability distribution constitutes a Bayesian network. This case has been extensively studied (Idika and Bhargava, 2012), notably in Poolsappasit et al. (2012) and Homer et al. (2009). This paper will thus focus on cyclic dependencies.

It is important to note that already in this simpler case, it is computationally infeasible to determine the full probability distribution of *general* Bayesian networks (Cooper, 1990). If one assumes further properties of the graph, efficient algorithms do exist, though (Zhang and Poole, 1994). A new approach will thus be required to support cyclic graphs as well.

3.5.2. *The general case*

For cyclically related security incidents, computing their likelihoods constitutes a more delicate problem than it seems.

Based on how the model is defined, an event (i.e., a node) is not triggered multiple times throughout the course of the experiment, but only once. For a concrete instance of the random experiment, the issue consists in finding all events that are activated by any of the root causes. See for example Fig. 7: event E can be caused by C or F, but inspecting the situation in more detail, E is only caused by either of the two event chains $A \to B \to C \to E$ or $F \to E$. In particular, E can only by triggered by C if C is not already triggered by the chain $F \to E \to D \to B \to C$.

By consequence, the probability that a risk scenario occurs cannot only be expressed by the probability of its direct parents, but has to involve all cycle-free paths from a root node. The computation effort for enumerating all such paths can be huge (in the worst case, namely in complete graphs, the running time

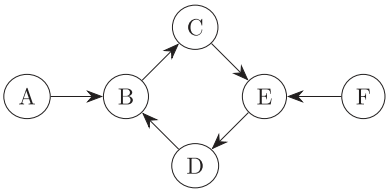

Fig. 7 – A simple cyclic dependency graph.

is exponential in the number of vertices), which is also why any efforts in finding an *efficient* deterministic algorithm failed. Instead, the randomised algorithm given in Algorithm 1 aims to give a good approximation; note that risk assessments as introduced in this paper do not require input data to be precise and are stable with respect to fluctuations.

The running time of Algorithm 1 is polynomial in its input data. More precisely, it is bounded by

$$\mathcal{O}\left(n \cdot m \cdot \ln\left(\frac{2n}{\delta}\right) \cdot \varepsilon^{-3}\right),$$

where n is the number of vertices, m is the number of edges, δ is the probability that the algorithm output is wrong and ε is an upper bound for the absolute error of the computed values. Observe the logarithmic dependency on δ, which permits amplifying the algorithm accuracy without significantly increasing its running time.

Algorithm 1 Compute probability matrix

Input: Graph $G = (V, E)$ with root nodes $V_R \subset V$

Input: Probability map $p : E \to [0, 1]$

Input: $\varepsilon > 0$, $\delta > 0$

Output: Probabilities $C : V_R \times V \to [0, 1]$ that a root node causes a node, each value with absolute error at most ε. The algorithm will fail with probability at most δ.

$\gamma := \frac{\varepsilon}{1+\sqrt{\varepsilon}}$
$N := \frac{6}{\varepsilon^2 \gamma} \ln\left(\frac{2n}{\delta}\right)$ where $n := |V|$
// N is chosen large enough so that there are guaranteed bounds on the
// error probabilities – those are formally proven in Proposition 1.

for $(v_r, v) \in V_R \times V$ **do**
    $C(v_r, v) \leftarrow 0$.
loop N times
    Sample a random graph G' from G according to p
    for $v_r \in V_R$ **do**
        for $v \in V(G')$ **do**
            if $\exists$ path in G' from v_r to v **then**
                $C(v_r, v) \leftarrow C(v_r, v) + 1/N$
return C

The proof of correctness, running time and error probability is given in the Appendix.

3.6. *Real-time risk monitoring*

The DARC model paves the way for real-time risk monitoring in the sense that the likelihoods of the root nodes, which usually have to be estimated manually, can be automatically determined by external sources such as intrusion detection systems (IDS) or security information and event management (SIEM) appliances. Note that "real-time" denotes a process where an explicit bound on the running time is known.

Observe that the function $C : V_R \times V \to \mathbb{R}$ computed in Algorithm 1 only depends on the probability weights p encoded into the graph, but not on the likelihoods of the root causes. By consequence, since the model (including p) is not supposed to change during the risk monitoring phase, the above

algorithm is invoked once, namely after the model design phase. As such, C can be considered static.

Since C expects two arguments, it can be viewed as a two-dimensional matrix $C \in \mathbb{R}^{V_R \times V}$ where each row represents a root node $\in V_R$ and each column an arbitrary node $\in V$. In fact, an entry of C denotes the probability that a given root node (=row) entails any given node (=column).

If the vector $L_R \in \mathbb{R}^{V_R}$ denotes the (estimated) likelihoods of root causes, then the likelihoods L of all the nodes can be computed as

$$L = L_R^T \cdot C,$$

where $\cdot$ denotes matrix multiplication and L_R^T the transpose of the vector L_R. Moreover, let $I \in \mathbb{R}^V$ be the vector that holds the direct impact caused by each node. The global risk can finally be written as

$$\text{risk} = L_R^T \cdot C \cdot I.$$

In a more explicit fashion,

$$\text{risk} = \sum_{v_r \in V_R} \sum_{v \in V} L_R(v_r) \cdot C(v_r, v) \cdot I(v).$$

C is computed from the model by Algorithm 1. The impacts I are manually estimated by a risk assessor for every security incident (including root nodes) in the graph – the value 0 describes incidents without impact. The likelihoods L_R of the root causes are either manually estimated (if they are static) or dynamically monitored by external sources (IDS, SIEM, etc.).

4. Experiments

The algorithm is constructed in such a way that it can be interrupted at any point, yielding a less precise, but complete solution. More precisely, if it is aborted after αN steps, for $0 < \alpha < 1$, then the relative error ε will increase at most by a factor $\alpha^{-\frac{1}{3}}$: this estimate follows directly from the definition of N (the number of simulation iterations) and is formally proven in Lemma 1 in the Appendix.

Moreover, since all simulations are run in an independent manner, they can be perfectly run in parallel (profiting from multi-threading capabilities of a CPU) or in a distributed way.

To test the performance of the algorithm on "average" graphs, dependency graphs have been generated uniformly at random. A typical[1] risk analysis may cover up to 50 different assets, each of which generally encounters 3–5 threats, so a related graph is composed of a few hundred nodes. It is sensible to assume that nodes are not connected (on average) to more than a few edges, so a typical graph will consist of a few thousand edges at most.

The simulation was performed on a dual-core 2.5 GHz processor (i7-3537U). The results are depicted in Figs 8–10 – as expected, they reflect the running time computed in Proposition 1 in the Appendix. The precise numbers can be found in Table A1 in the Appendix.

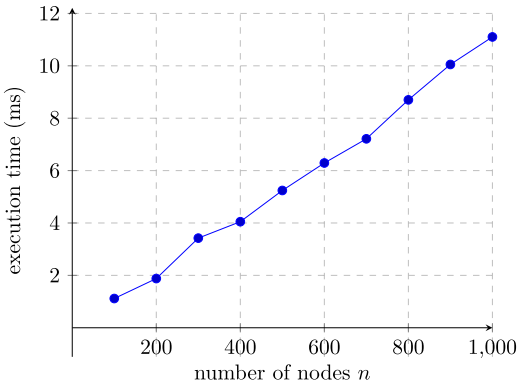

Fig. 8 – Execution time of Algorithm 1 in seconds, depending on the graph size n, with $\varepsilon = 0.1$ and $\delta = 0.01$ and an average of 5 neighbours per node.

In order to compare the performance of Algorithm 1 to other approaches, similar experiments have been conducted with

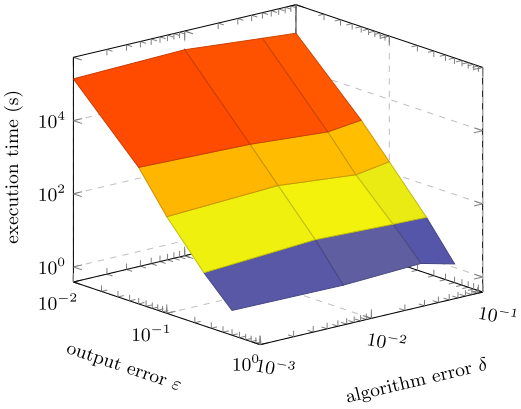

Fig. 9 – Execution time of Algorithm 1 in seconds, depending on the algorithm precision ε and δ, with $n = 1000$ and $m \approx 5000$. Note the logarithmic scales.

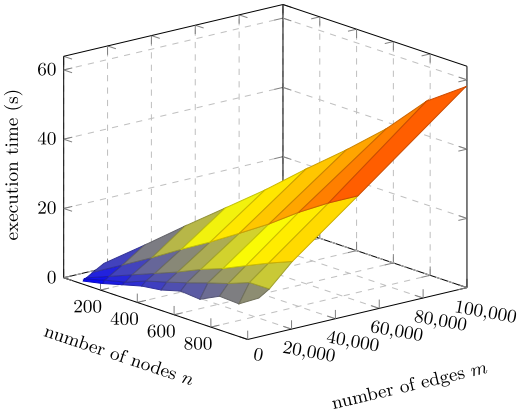

Fig. 10 – Execution time of Algorithm 1 in seconds, depending on the graph size n and m, with $\varepsilon = 0.1$ and $\delta = 0.01$.

[1] Based on the experience from past risk analyses performed by the authors.

Table A1 – Results of the simulation experiments

$\|V\|$	$\|E\|$	ε	δ	Iterations	Time (s)
Varying size of input graph					
100	515	0.1	0.01	57290	1.1163832
200	995	0.1	0.01	62764	1.8782055
300	1558	0.1	0.01	65966	3.4224201
400	2010	0.1	0.01	68238	4.0489518
500	2571	0.1	0.01	70000	5.2410411
600	2998	0.1	0.01	71440	6.290252
700	3463	0.1	0.01	72657	7.2129071
800	4079	0.1	0.01	73712	8.7000471
900	4499	0.1	0.01	74642	10.0516358
1000	5058	0.1	0.01	75474	11.1025155
Varying precision of algorithm output					
500	2523	0.5	0.01	726	0.0577856
500	2392	0.2	0.01	9620	0.72131
500	2541	0.1	0.01	70000	5.117056
500	2464	0.05	0.01	520596	31.2303292
500	2557	0.02	0.01	7587969	274.7254227
Varying algorithm error probability					
500	2574	0.1	0.0001	88184	6.5013409
500	2521	0.1	0.001	79092	5.8744299
500	2479	0.1	0.01	70000	5.3008785
500	2540	0.1	0.1	60908	4.4340709
Varying average number of node neighbours					
100	508	0.1	0.01	57290	0.74860860
200	980	0.1	0.01	62764	1.74781950
200	2004	0.1	0.01	62764	2.57128770
200	3957	0.1	0.01	62764	3.36202100
200	10067	0.1	0.01	62764	5.74008430
200	19992	0.1	0.01	62764	9.95537590
300	1491	0.1	0.01	65966	3.10028020
300	2956	0.1	0.01	65966	3.62702600
300	6032	0.1	0.01	65966	5.28465980
300	15068	0.1	0.01	65966	8.88317620
300	30075	0.1	0.01	65966	15.73896320
400	2002	0.1	0.01	68238	4.42690590
400	4007	0.1	0.01	68238	5.58108290
400	7993	0.1	0.01	68238	7.18062480
400	20094	0.1	0.01	68238	12.31568510
400	40110	0.1	0.01	68238	21.25638300
500	2497	0.1	0.01	70000	4.99750850
500	5091	0.1	0.01	70000	6.88772690
500	10048	0.1	0.01	70000	9.05478420
500	24883	0.1	0.01	70000	15.50143540
500	49828	0.1	0.01	70000	26.78152520
600	3012	0.1	0.01	71440	6.38884460
600	5897	0.1	0.01	71440	8.35841490
600	12163	0.1	0.01	71440	11.27393760
600	29795	0.1	0.01	71440	19.07167170
600	60180	0.1	0.01	71440	33.08584440
700	3393	0.1	0.01	72657	5.75831260
700	7078	0.1	0.01	72657	9.93901220
700	13910	0.1	0.01	72657	13.05916540
700	34940	0.1	0.01	72657	22.79221370
700	70164	0.1	0.01	72657	38.90024520
800	3936	0.1	0.01	73712	8.32702800
800	7963	0.1	0.01	73712	11.40233290
800	16012	0.1	0.01	73712	15.34878480
800	40093	0.1	0.01	73712	27.16270650
800	79989	0.1	0.01	73712	45.30316310
900	4426	0.1	0.01	74642	7.83409540
900	9219	0.1	0.01	74642	13.27833110
900	17995	0.1	0.01	74642	17.58192730
900	45347	0.1	0.01	74642	30.87114070
900	89958	0.1	0.01	74642	53.56402540
1000	5073	0.1	0.01	75474	11.19835840
1000	9867	0.1	0.01	75474	13.10751250
1000	20336	0.1	0.01	75474	19.48352650
1000	50238	0.1	0.01	75474	33.94672970
1000	99469	0.1	0.01	75474	58.11831390

recursive deterministic algorithms, which condition on the existence of each edge in the graph (Algorithm 2).

Algorithm 2 Compute probability matrix in a recursive way

Input: Graph $G = (V, E)$ with root nodes $V_R \subset V$

Input: Probability map $p : E \to [0, 1]$

Output: Probab. $C : V_R \times V \to [0, 1]$ that a root node causes another node.

if $\exists e \in E$ with $0 < p(e) < 1$ **then**
  Set $p_0 := p$ but $p_0(e) := 0$. Recursively run on (G, p_0) to get output C_0.
  Set $p_1 := p$ but $p_0(e) := 1$. Recursively run on (G, p_1) to get output C_1.
  For all $x \in V_R, y \in V$: $C(x, y) := C_0(x, y) \cdot (1 - p(e)) + C_1(x, y) \cdot p(e)$
else
  For all $x \in V_R, y \in V$: $C(x, y) := 1$ if there is a path $x \to y$, otherwise 0.

Unfortunately, such algorithms have exponential running time and take more than a few minutes already for small graphs ($|V| \geq 20$, $|E| \geq 100$). All other attempts to solving the problem in a deterministic way resulted in similarly bad execution times. More advanced algorithms (see for instance Homer et al., 2009) suffer from the same problem.

The simulations revealed that $\varepsilon = 0.1$ and $\delta = 0.01$ are sensible choices for typical dependency graphs ($n \leq 1000$, $m \leq 5000$). Indeed, the running time is still relatively low (1–2 minutes), but the computed values lie very close (±8%) to the ones produced by $\varepsilon = 0.01$ and $\delta = 0.001$, although the computation of the latter took much longer (38 hours). The values computed by setting $\varepsilon = 0.05$ lie within ±5% of the high-precision results, but the running time is increased to 15 minutes; depending on the context, this may or may not be acceptable. In general, reducing ε is more costly (in terms of running time) than reducing δ, but has a much greater impact on the precision of the output values.

Computing the risk faced by an organisation requires three steps: manually collecting impacts I and likelihoods L_R, computing the probability matrix C using Algorithm 1, and finally evaluating the matrix product $L_R^T \cdot C \cdot I$. It turns out that the multiplication of those matrices is very fast and insignificant with respect to the running time of Algorithm 1 for all choices of $\varepsilon < 0.5$ and $\delta < 0.1$.

5. Extension to boolean formulas

The dependency graph is based on the concept of causality; that is, the parents of a node represent alternative causes, each of which can engender the consequential scenario. Formally, the dependency relationship of a vertex v_0 and its parent nodes $P_{v_0} \subset V(G)$ can be expressed as

$$\rho(v_0) := \bigvee_{x \in P_{v_0}} \mathbb{I}_x,$$

where $\mathbb{I}_x$ denotes the boolean variable encoding whether the event x occurs or not.

The beauty of Algorithm 1 lies in the fact that it does not depend at all on the topology of the graph or on the form of the dependencies. In fact, generalising the "OR" relations to arbitrary boolean expressions $\rho(\cdot)$ is straightforward and does not change the main lines or the proof of the algorithm; yet

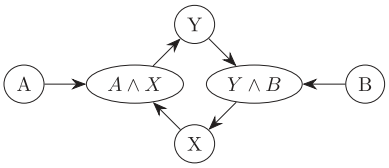

Fig. 11 – Endless loop in dependencies for general boolean formulas.

considerable adaptations have to be made in order to find whether a node is triggered or not (line 10 in the algorithm). For general boolean formulas the effort for computing the likelihoods is considerably higher, since a recursive search might no longer be possible.

In order to evaluate $A \wedge X$, one needs to evaluate *both* parents, including X and thus $Y \wedge B$ and Y (see Fig. 11). But Y can only be evaluated if $A \wedge X$ is known already. It is not so clear how to proceed in such a case: here, one solution is to set the likelihood to zero for all non-reachable nodes, because the cycle can never be entered; however, this might not be sensible in all situations. A different theory, such as boolean satisfiability (Clote and Kranakis, 2013), is required in these matters.

The comparatively good running time of Algorithm 1 was due to the fact that evaluating the likelihood (sc. finding all cycle-free paths) can be implemented in an efficient way. For more complex boolean formulas this may not necessarily be the case (indeed, the boolean satisfiability problem[2] SAT is $\mathcal{NP}$-complete (Clote and Kranakis, 2013)) so that deterministic (and possibly even error-free) algorithms could outperform the simulation variant.

6. Conclusion and outlook

This paper provides a simple and lightweight approach for encoding asset dependencies into a graph structure. Since that graph is not assumed to be acyclic (which allows to model situations where asset A depends on B, which depends on C, which depends on A again), the model can also be used in environments with interdependencies, such as in Industrial Control Systems (ICS) or Critical Infrastructures (CI). The major contribution of this piece of work (apart from the DARC model) is Algorithm 1, which computes the resulting risk in such a graph, but in a provably efficient way. Indeed, as it turned out, any deterministic approach that we could think of is computationally too complex to serve as a basis for any usable algorithm. Indeed, the running time is exponential in the number of nodes and edges, which rapidly becomes a problem already for small graphs ($|V| \geq 30$).

The DARC model was developed with the intention of creating a tool that continuously computes and monitors the *current* risk faced by an organisation, taking all dependencies into account. For now, it merely encodes the causal relationships between incidents (i.e. *A* causes *B*), so that a quantitative risk assessment can only be performed in a very basic way (risk = likelihood × impact). Therefore, the next steps consist in embedding the DARC model into a whole risk methodology, by including more fine-grained notions into the model (such as threat exposure, vulnerabilities or preventive measures). Doing so will also enable more sources of risk information to be integrated into the monitoring tool, for instance software agents that rate and report the performance of preventive security measures.

A open problem related to quantitative risk assessments is the lack of statistical data required for estimating the likelihoods of risk scenarios. A quick workaround consists in specifying confidence intervals for the input values and evaluating the risk model for the lower and upper bounds, respectively.

The concepts developed in this paper are currently being implemented for a real organisation in the electricity domain; an upcoming publication is planned for publishing the results and lessons learned.

Acknowledgements

This work was supported by the Fonds National de la Recherche Luxembourg (project reference 10239425).

Appendix

Proposition 1. *Algorithm 1 is correct with probability δ and terminates within time*

$$\mathcal{O}\left(n \cdot m \cdot \ln\left(\frac{2n}{\delta}\right) \cdot \varepsilon^{-3}\right).$$

Moreover, each computed value lies within an interval of $\pm\varepsilon$ around the true value.

Proof. Fix a root node $v_r \in V_R$. For $v \in V$ and $1 \leq i \leq N$, let $X_i(v)$ be the indicator variable that there is a path from v_r to v in the i-th random experiment. Observe that

$$\frac{1}{N}\sum_{i=1}^{N} \mathbb{E}[X_i(v)] = \mathbb{E}[X_0(v)] = \mathbb{P}[X_0(v) = 1],$$

that is, the quantity approximated by the algorithm (left hand-side) equals the probability that node v is reachable by v_r. So if the random experiments do not deviate too much from their expectations, the algorithm output is correct up to a certain relative error, which is determined in the following.

Define γ and N as in the algorithm. Note that $\gamma < \varepsilon < 1$.

Fix $v \in V$. Suppose for now that $\mu(v) \geq \gamma$. Using a two-sided Chernoff bound (Motwani and Raghavan, 2010),

$$\mathbb{P}\left[\left|\sum_{i=1}^{N} X_i(v) - N\mu(v)\right| > N\mu(v)\varepsilon\right] \leq 2\exp\left(-\frac{\varepsilon^2}{3}N\gamma\right)$$
$$= \frac{\delta}{n^2}. \quad \text{(by the choice of N)}$$

[2] Given an arbitrary boolean formula ρ on variables $x_1, \ldots, x_n$, the SAT problem consists in determining whether there is a value $\alpha \in \{0,1\}^n$ for all of these variables x such that $\rho(x_1 := \alpha_1, \ldots, x_n := \alpha_n) = 1$.

If however $\mu(v) \le \gamma < \varepsilon$, using a one-sided Chernoff bound,

$$\begin{aligned}\mathbb{P}\left[\sum_{i=1}^{N} X_i(v)/N > \varepsilon\right] &= \mathbb{P}\left[\sum_{i=1}^{N} X_i(v) > \left(1+\frac{\varepsilon-\mu(v)}{\mu(v)}\right)N\mu(v)\right] \\ &\le 2\exp\left(-\left(\frac{\varepsilon-\mu(v)}{\mu(v)}\right)^2 \frac{N\mu(v)}{3}\right) \\ &\le 2\exp\left(-(\varepsilon-\gamma)^2\frac{N}{3\gamma}\right) \\ &= 2\exp\left(-\left(\frac{\varepsilon-\gamma}{\varepsilon\gamma}\right)^2 \ln\left(2n^2/\delta\right)\right). (*)\end{aligned}$$

By the definition of γ it holds that $\varepsilon - \gamma > \varepsilon\gamma$ and thus $(*) \le \frac{\delta}{n^2}$. To summarise, with probability at least δn^{-2} the following two statements hold:

- If $\mu(v) \ge \gamma$ then the relative error of the random experiment is at most ε; however, since $\mu(v) < 1$, this also implies that the absolute error $e(v) := \left|\sum_{i=1}^{N} X_i(v) - N\mu(v)\right|$ is at most ε.
- If $\mu(v) \le \gamma$ then the absolute error $e(v)$ error is at most ε.

Note that the statements above hold for any fixed vertex v_0 and any fixed root node $v_{r,0}$. Using a union bound,

$$\mathbb{P}[\exists v_r \exists v : e(v) > \varepsilon] \le n^2 \cdot \mathbb{P}[e(v_0) > \varepsilon] = \delta$$

yielding the desired error probability for the algorithm.

Regarding the running time, note that the inner FOR-loop can be implemented (e.g. using a breadth-first search) in linear time $\mathcal{O}(m)$ for each of the at most n root nodes, whereas the sampling requires time $\mathcal{O}(m)$, resulting in a total execution time of $\mathcal{O}(n \cdot m \cdot N)$. □

Lemma 1. *For fixed δ, if Algorithm 1 is aborted after αN iterations, for $0 < \alpha < 1$, then the relative error of the algorithm output increases at most by a factor $\alpha^{-\frac{1}{3}}$.*

Proof. Suppose Algorithm 1 requires N_0 iterations to achieve a relative error ε_0. If one picked $\varepsilon = \beta\varepsilon_0$, it would require

$$\begin{aligned}\frac{\left(1+\sqrt{\varepsilon}\right)}{\varepsilon^3}\cdot 6\log\frac{2n}{\delta} &= \frac{\left(1+\sqrt{\beta}\sqrt{\varepsilon_0}\right)}{\beta^3\varepsilon_0^3}\cdot 6\log\frac{2n}{\delta} \\ &= \beta^{-3}\frac{\left(1+\sqrt{\beta}\sqrt{\varepsilon_0}\right)}{\left(1+\sqrt{\varepsilon_0}\right)}\cdot N_0 \le \beta^{-3}\cdot N_0\end{aligned}$$

iterations instead, for any $0 < \beta < 1$. Since the outputs get more precise the longer the algorithm runs, running it precisely $\beta^{-3}N_0$ times will yield a relative error $\beta\varepsilon_0$ (or even better). Setting $\beta = \alpha^{-\frac{1}{3}}$ concludes the proof. □

REFERENCES

Amutio M.A., Candau J., Mañas J. Magerit–version 3, methodology for information systems risk analysis and management, book I – the method, Ministerio de administraciones públicas.

Aubigny M, Harpes C, Castrucci M. Risk ontology and service quality descriptor shared among interdependent critical infrastructures. In: Critical information infrastructures security. Springer; 2011. p. 157–60.

Baiardi F, Sgandurra D. Assessing ICT risk through a Monte Carlo method. Environ Syst Decis 2013;33(4):486–99.

Bishop M. Computer security: art and science, vol. 200. Addison-Wesley; 2012.

Brændeland G, Refsdal A, Stølen K. Modular analysis and modelling of risk scenarios with dependencies. J Syst Softw 2010;83(10):1995–2013.

Breier J. Asset valuation method for dependent entities. J Int Ser Inf Secur 2014;4(3):72–81.

Clote P, Kranakis E. Boolean functions and computation models. Springer Science & Business Media; 2013.

Cooper GF. The computational complexity of probabilistic inference using Bayesian belief networks. Artif Intell 1990;42(2):393–405.

Cox LAT, Babayev D, Huber W. Some limitations of qualitative risk rating systems. Risk Anal 2005;25(3):651–62.

Fenz S, Tjoa AM, Hudec M. Ontology-based generation of Bayesian networks. In: International Conference on Complex, Intelligent and Software Intensive Systems, 2009 (CISIS'09), IEEE; 2009. p. 712–17.

Homer J, Ou X, Schmidt D. A sound and practical approach to quantifying security risk in enterprise networks, Kansas State University Technical Report, 1–15; 2009.

Idika N, Bhargava B. Extending attack graph-based security metrics and aggregating their application. IEEE Trans Dependable Secure Comput 2012;9(1):75–85.

Laboratory SL. Qualitative vs. quantitative risk assessment, <http://www.sans.edu/research/leadership-laboratory/article/risk-assessment>; 2012.

Liu N, Zhang J, Wu X. Asset analysis of risk assessment for IEC 61850-based power control systems – part I: methodology. IEEE Trans Power Deliv 2011;26(2):869–75.

Loloei I, Shahriari HR, Sadeghi A. A model for asset valuation in security risk analysis regarding assets' dependencies. In: 20th Iranian Conference on Electrical Engineering (ICEE 2012), IEEE; 2012. p. 763–8.

Mangan J, Lalwani C, Gardner B. Combining quantitative and qualitative methodologies in logistics research. Int J Phys Distrib Logist Manag 2004;34(7):565–78.

McQueen M, Boyer WF, Flynn M, Beitel G. Quantitative cyber risk reduction estimation methodology for a small SCADA control system. In: Proceedings of the 39th Annual Hawaii International Conference on System Sciences, 2006 (HICSS'06), vol. 9. IEEE; 2006. p. 226.

Motwani R, Raghavan P. Randomized algorithms. Chapman & Hall/CRC; 2010.

Pearl J. Causality: models, reasoning, and inference. New York: Cambridge University Press; 2000.

Pederson P, Dudenhoeffer D, Hartley S, Permann M. Critical infrastructure interdependency modeling: a survey of US and international research. Idaho National Laboratory; 2006. p. 1–20.

Poolsappasit N, Dewri R, Ray I. Dynamic security risk management using Bayesian attack graphs. IEEE Trans Dependable Secure Comput 2012;9(1):61–74.

Rahmad B, Supangkat SH, Sembiring J, Surendro K. Modeling asset dependency for security risk analysis using threat-scenario dependency. Int J Comput Sci Inf Secur 2012;10(4):103.

Tong X, Ban X. A hierarchical information system risk evaluation method based on asset dependence chain. Int J Secur Its Appl 2014;8(6):81–8.

Utne IB, Hokstad P, Vatn J. A method for risk modeling of interdependencies in critical infrastructures. Reliab Eng Syst Saf 2011;96(6):671–8.

Wang L, Islam T, Long T, Singhal A, Jajodia S. An attack graph-based probabilistic security metric. In: Data and applications security XXII. Springer; 2008. p. 283–96.
Xin T, Xiaofang B. A hierarchical information system risk evaluation method based on asset dependence chain. Int J Inf Netw Secur 2014;3(3).
Zhang NL, Poole D. A simple approach to Bayesian network computations. In: Proc. of the Tenth Canadian Conference on Artificial Intelligence. 1994.
International Organization for Standardization, ISO/IEC 27000; 2014.