

A Prototype Tool for Distinguishing Attacks and Technical Failures in Industrial Control Systems

Khurshid Abbas¹, Sabarathinam Chockalingam², Thi Thuy Nga Dinh¹,
Vikash Katta²

¹Østfold University College, PO Box 700, 1757 Halden, Norway

²Institute for Energy Technology, Os Alle 5, 1777 Halden, Norway

khurshid.abbas@hiof.no, sabarathinam.chockalingam@ife.no,
thi.t.dinh@hiof.no, vikash.katta@ife.no

Abstract

Critical Infrastructures (CIs) are governed by Industrial Control Systems (ICSs). Modern ICSs do not operate in isolation anymore, but they are connected to the Internet. This transformation introduced numerous advantages, however, there are a few drawbacks as well. Integration with the Internet has left ICS exposed to potential cyber-attacks. Additionally, ICSs could also encounter technical failures during operation. Consequently, it is crucial to distinguish between attacks and technical failures to initiate an appropriate response. There is a deficiency of robust technology to assist operators in distinguishing attacks and technical failures in an ICS environment. However, a framework is proposed to construct Bayesian Network (BN) models that would help to distinguish between attacks and technical failures for different observable problems in our previous work. There are tools available to implement such BN models, but these tools are not appropriate to use in an ICS environment. In order to address this limitation, this paper develops and demonstrates a prototype tool for swift identification of the major cause (Intentional Attack/Accidental Technical Failure) in case of an abnormal behaviour in a component of ICS.

The proposed tool enables BN models to automatically update prior probabilities based on the historical data and/or expert knowledge corresponding to the application. The developed tool can be further evaluated and used to distinguish between attacks and technical failures during operation in CIs where ICSs are employed.

Keywords: Attack, Bayesian Network, Cyber Security, Incident Response, Safety, Technical Failure.

1 Introduction

Critical Infrastructures (CIs) such as energy, water management infrastructures are increasingly relying on Industrial Control Systems (ICSs) [1, 2]. ICSs ensure smooth operations of physical processes in such infrastructures and support end-users. Modern ICS do not operate in isolation, but they are connected to the Internet. This transition has induced numerous advantages such as efficient performance of manufacturing systems.

However, this also makes it vulnerable to cyber-attacks in addition to technical failures [3]. For instance, the German steel mill incident is a typical example of a cyber-attack. In this attack, the adversaries exploited the corporate networks and then penetrated the ICS network to cause physical damage to the blast furnace [4]. On the other hand, sensors sending wrong measurements due to ageing in an ICS is a typical example of a technical failure. Usually, when the operator notices an abnormal behaviour in an ICS, they diagnose it as a technical failure and initiate corresponding response strategies. This is because technical failures are more frequent than attacks [5]. Choosing response strategies corresponding to technical failures for an attack might result in substantial damage to the ICS as it might not be appropriate. Importantly, adequate decision-making technology is not available to help operators distinguish between attacks and technical failures [6].

There has been a little work done on distinguishing attacks and technical failures in ICS which includes [6-8]. In our previous work, we proposed a framework to address the above-mentioned problem of distinguishing attacks and technical failures with the help of the Bayesian Network (BN) models [6, 9]. BNs belong to the family of probabilistic graphical models. Typically, BNs consist of several variables and probabilistic dependencies among these variables [6].

There are numerous tools available such as BayesFusion, GeNIe Modeler and SMILE Engine to implement such BN models [10, 11]. However, there is a lack of tool support that can help such BN models to be used in an ICS environment for distinguishing attacks and technical failures, which is the aim of this work. Typically, the above-mentioned tools offer rich graphical user interfaces and useful functionalities such as creating networks, probability propagation. For instance, it is possible to develop a BN model for our problem using GeNIe modeler. In the developed BN model, the objective can be achieved by providing evidence to some variables, however, it is time-consuming to manually update the evidence of individual nodes and then compute posterior probabilities corresponding to the target variable. Furthermore, it is also time-consuming to update prior probabilities corresponding to each node based on new data on attacks and technical failures. These tools might be suitable for research purposes to develop BN-models, but not for an ICS environment as it is time constrained. Swift and reliable decision support is needed to initiate a timely and effective response to a problem caused by an attack/technical failure. This would help to recover the system from adversaries promptly and minimise negative consequences. In addition, SMILE engine is also not directly usable by an operator as it is a programming library and require personnel having sound programming knowledge as a prerequisite in deploying the tool. Therefore, they cannot be customised to be implemented in an ICS environment. Consequently, a tool is required that is specifically tailored to the necessities of the ICS environment based on the developed BN-based framework by addressing the following research question (RQ) and corresponding research objectives (ROs):

- **RQ.** How to develop a prototype tool that could help to use Bayesian Network (BN) models in an Industrial Control Systems (ICSs) environment to distinguish between attacks and technical failures?
- **RO1.** To develop a prototype tool that can be used in an ICS environment for distinguishing attacks and technical failures.
- **RO2.** To demonstrate and evaluate the developed prototype tool for distinguishing attacks and technical failures in an ICS environment.

The main contributions of this paper are the following: (i) a prototype tool that can help to use BN models for distinguishing attacks and technical failures in an ICS environment effectively taking into account the end-user requirements, (ii) demonstration and initial evaluation of the developed prototype tool to refine it further and evaluate it in the real environment in the future.

This paper is structured as follows: Section 2 describes typical layers and components of an ICS, the structure of a BN with an example and the proposed BN-based framework to distinguish between attacks and technical failures. Section 3 explains the method which we used to tackle the above-mentioned research question. Section 4 is dedicated to the requirements elicitation. Sections 5, 6 and 7 develop, demonstrate and evaluate the developed prototype tool respectively. Finally, Section 8 provides conclusions and future work directions.

2 Background

2.1 Industrial Control Systems

Generally, an ICS consists of three layers: (i) field instrumentation layer, (ii) process control layer and (iii) supervisory control layer as shown in Figure 1 [6]. The field instrumentation layer consists of sensors and actuators. The process control layer consists of Programmable Logic Controllers (PLCs) to process the data collected from the sensors and control physical processes through actuators. The supervisory control layer consists of historian databases and software applications that are executed on the workstations to ensure efficient operation of the ICS [6].

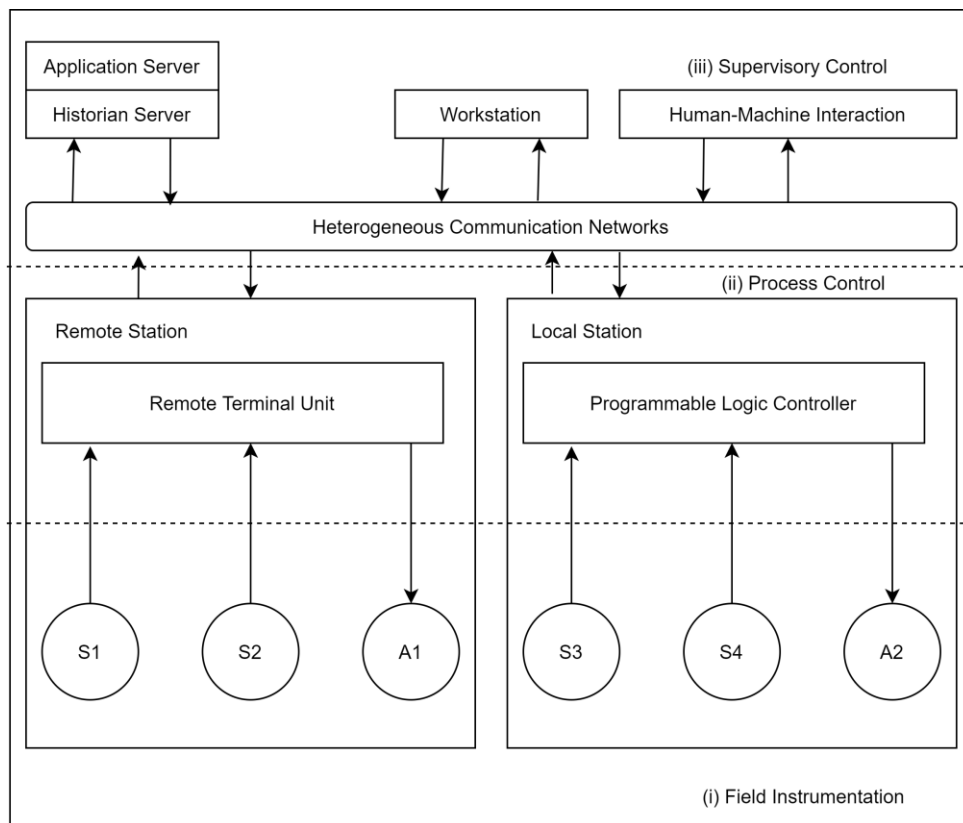


Figure 1: Typical Industrial Control System (ICS) Architecture

2.2 Bayesian Networks

BNs are known to be a type of probabilistic graphical models. A BN consists of a number of variables (nodes) and edges between these variables [10]. The edges connecting the nodes (variables) constitute the directed acyclic graphs. Each variable has a number of states and a conditional probability table (CPT) associated with it. The CPT entries are typically based on historical data and/or expert knowledge [12]. Subsequently, when evidence is available for different variables in the BN, the probabilities of other non-evidenced variables are updated accordingly. This process of updating probabilities is described as probability propagation or belief updating [10].

BN support four different types of reasoning: (i) predictive reasoning, (ii) diagnostic reasoning, (iii) intercausal reasoning and (iv) combined reasoning. The framework proposed by Chockalingam et al. [6] helps to develop BN models that utilise combined reasoning, which is the combination of predictive and diagnostic reasoning. Diagnostic or bottom-up reasoning is the process of inferring a cause using symptom(s). In Figure 2, a cause (Y) can be inferred by providing evidence to symptom(s) (Z1 and/or Z2). On the other hand, predictive or top-down reasoning is the process of reasoning from cause(s) (X1 and/or X2) to an effect. In Figure 2, an effect (Y) can be inferred by providing evidence to cause(s) (X1 and/or X2).

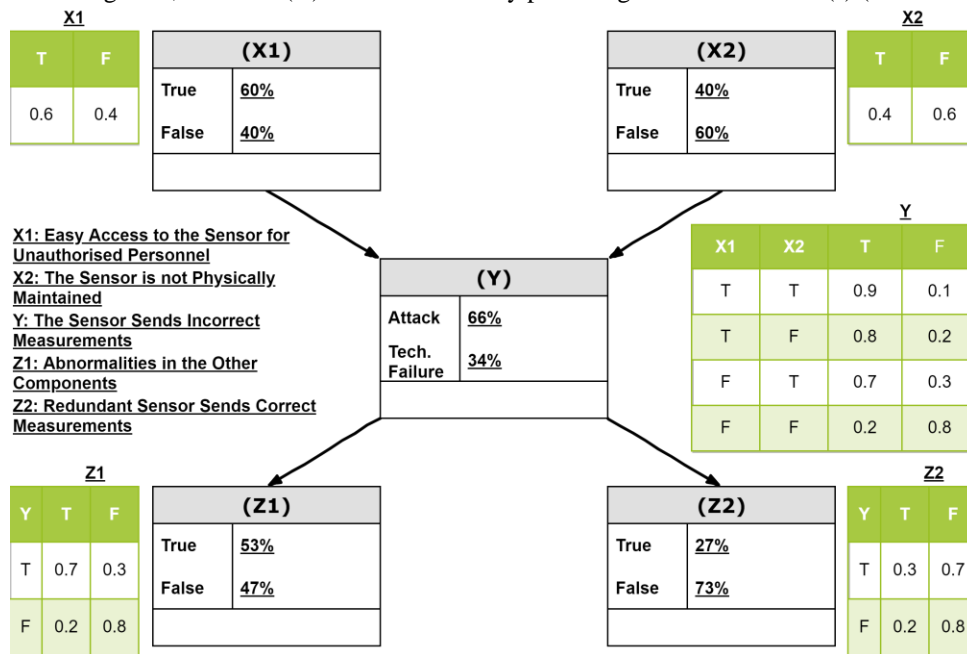


Figure 2: Bayesian Network Example

2.3 Framework for Distinguishing Attacks and Technical Failures

Chockalingam et al. [6] proposed a framework to construct BN models that would help to distinguish between attacks and technical failures. Such BN models would help to choose an effective response strategy to abnormal behaviour in an ICS. The proposed framework consists of three different types of variables i.e., contributory factors, observable problem and observations (or test results) as shown in Figure 3 [6]. The contributory factors are factors that could contribute to the considered problem due to an attack or technical failure, whereas the observations (or test results) provide real-time system information based on the outcome of tests conducted as soon as an operator notices the problem [6].

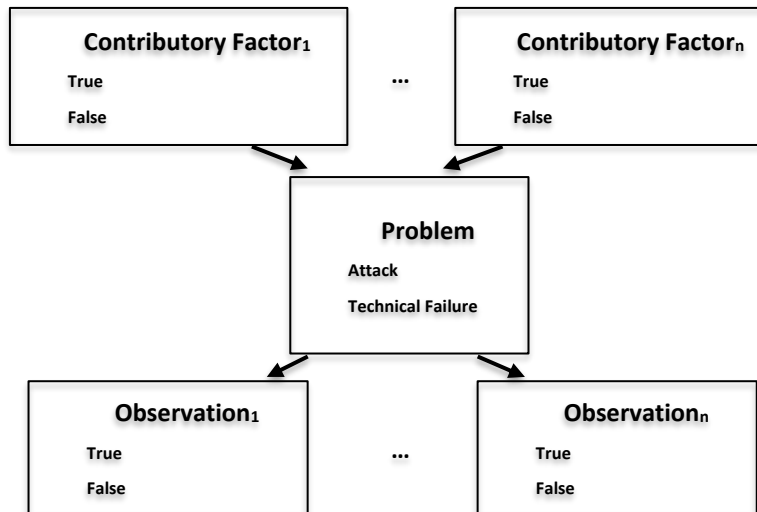


Figure 3: Framework to Develop BN Models for Distinguishing Attacks and Technical Failures

3 Research Methodology

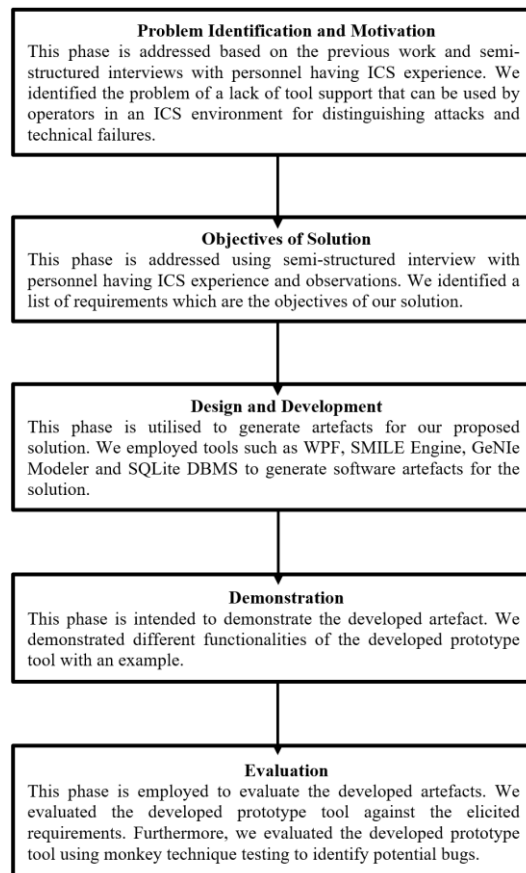


Figure 4: Research Methodology Corresponding to DSRM Phases

In this paper, we aim to solve the practical problem of a lack of tool support that can be used by operators in an ICS environment for distinguishing attacks and technical failures by addressing the above-mentioned research question. We tackled our research question using the Design Science Research (DSR) methodology, which is widely used to create artefacts [13]. The practical problems can be solved by using artefacts in numerous cases. The DSR process consists of five main phases as shown in Figure 4: (i) problem identification and motivation, (ii) objectives of solution, (iii) design and development, (iv) demonstration, and (v) evaluation [14]. Figure 4 also shows how we addressed each phase in this study.

4 Requirements Elicitation

A tool that can be used in an ICS environment to distinguish between attacks and technical failures is lacking. In order to develop this tool, we gathered requirements from personnel with ICS experience and made observations from an ICS environment, which are listed in this section.

The major objectives of this tool are: (i) To support in developing BN models based on the framework proposed by Chockalingam et al. [6], (ii) To implement a repository which consists of attack/technical failure-related information that provides input to the developed BN model in computing prior probabilities and (iii) To integrate the developed BN model and corresponding repository which in turn would help to compute prior probabilities dynamically.

In order to gather requirements to develop a tool that can be used in an ICS environment, we conducted semi-structured interviews, made observations from an ICS environment and reviewed scientific literature corresponding to ICS. The semi-structured interviews were focused on the following aspects of the tool as they are the core elements of any such tools: (i) core functionality of the tool, (ii) development and production environment, (iii) scientific and statistical calculation methods, (iv) user interface and workflow of the tool.

The high-level requirements that were based on the interview, observations and literature study are: (i) A tool is required to use BN models developed using the proposed framework for distinguishing attacks and technical failures in an ICS environment, (ii) The tool should ensure a swift response in case an operator notices an abnormal behaviour in a component of the ICS, (iii) The tool should demand only minimal effort from the operator while using it, (iv) The tool should be a desktop application as most of the operators in such infrastructures uses such workstations, (v) The tool should be able to benefit from the historical data related to attacks/technical failures stored in a repository in addition to the expert knowledge, (vi) The tool should be capable of processing a BN designed in GeNIe Modeler, (vii) The tool should be able to handle the amendments into the BN i.e., operators should be able to provide evidence based on his/her observations and the BN should be able to update probabilities of the target variable based on the provided input, (viii) The tool should be able to acquire historical data related to attacks/technical failures and dynamically compute CPT values of the developed BN models.

5 Design and Development

Design and development phase address the structure of the solution. The design of the prototype tool depends on the requirements elicited in the previous phase.

Subsequently, during the development phase, the system requirements and design specifications are transformed into an executable system that could be implemented for use after evaluations.

Table 1 presents an overall relationship between different tools employed to implement the overall solution and how it addresses the gathered requirements.

Tool	Addressed Requirement	Primary Role
Windows Presentation Foundation (WPF)	The desktop nature of the solution.	WPF provides a solid basis for the desktop app development.
MahApps.Metro	Smooth usability and better User Experience (UX).	MahApps.Metro offers modern UI and user controls for enhanced UX.
GeNIe Modeler	Modelling Bayesian Networks.	GeNIe Tool shall be used to model BNs in a user-friendly manner.
SMILE Engine	Processing and amending a BN designed in GeNIe Modeler.	SMILE Engine provides the necessary computation for BN.
.NET Wrapper for SMILE	The desktop nature of the solution.	It enables the integration of SMILE Engine with WPF.
SQLite DBMS	The solution should be able to benefit from the historical information.	SQLite DBMS forms the core of the historical data repository that feeds the data to the model.
Entity Framework (EF) Core	Dynamic flow of information between the model and Historical Data Repository (HDR).	EF Core establishes a robust connection between the model and HDR, hence, CPT values can be updated dynamically.

Table 1: Relationship between the tools utilised and the addressed requirements.

5.1 Desktop Paradigm and Windows Presentation Foundation

The developed tool should be a desktop application based on the requirements specified in the previous section. Moreover, desktop applications are appropriate for deployment in control rooms. There are several frameworks to build desktop applications, Microsoft's Windows Presentation Foundation (WPF) is one of them. WPF is an open-source platform offered by Microsoft to build desktop applications with graphical user interfaces [15]. WPF runs on .NET Core. Therefore, it is cross-platform. WPF comprises of all essential features that a modern desktop application is supposed to possess. In addition, it is stable and trustworthy [15]. Consequently, our developed tool is a desktop application and built on Microsoft's Windows Presentation Foundation (WPF).

5.2 MahApps.Metro Library

The developed tool should only demand minimal efforts and guarantee a swift response from operators. In order to achieve that, the interface of the tool must be user friendly and highly usable. Because of this reason, MahApps.Metro is used in combination with WPF to enhance the user interface of the tool. MahApps.Metro is an open-source project developed by Paul Jenkins and is currently used by thousands of developers around the world to integrate modern user interfaces into their WPF applications [16]. MahApps.Metro combines essential graphical user interface elements to the user interface that enhances the user experience to a significant level [16].

5.3 GeNIe Modeler and SMILE Engine

The developed tool has to develop BN models for distinguishing attacks and technical failures based on the BN-based framework developed in [6]. There are several tools available to develop BN models for our application. Among them, GeNIe modeler is a well-known tool that is used to develop BN models. GeNIe modeler operates with the support of SMILE engine. SMILE engine is developed by BayesFusion LLC to work with Bayesian Networks [17]. It provides the essential computational capabilities to GeNIe modeler. GeNIe modeler is an excellent design tool for BN; however, it is not suitable to be used in an ICS environment as it is time-consuming to manually update evidence to compute posterior probabilities. Moreover, GeNIe Modeler cannot be used to communicate with a historical repository dynamically. Therefore, we only design BNs with help of GeNIe modeler in our developed tool to be used later with SMILE Engine.

Nevertheless, the above-mentioned deficiencies can be fulfilled by employing SMILE Engine. SMILE Engine is an advanced reasoning and discovery tool that can be utilised to perform the necessary calculations related to the BN model [18]. Besides, it offers the much-needed flexibility to be customised to the requirements of this tool. It is originally developed in C++, however, BayesFusion LLC offers a few more wrappers such as Java, Python, .NET and R, therefore, it can be used with other platforms as well [18].

5.4 .NET Wrapper for SMILE Engine

Since our tool must be a desktop application developed in WPF, therefore, a wrapper for SMILE engine is needed to be deployed into the .NET framework. SMILE engine offers several wrappers to be adopted into different platforms, one of them is for .NET that renders WPF and SMILE engine compatible. Therefore, .NET wrapper for SMILE engine is utilised to ensure compatibility between SMILE engine and WPF.

5.5 SQLite Relational Database Management System

One of the core requirements of the tool demands a feature that allows BN models to benefit from historical data related to attack/technical failure. A historical database needs to be built to store the historical data on attacks/technical failures. Once the data related to attacks and technical failures have been stored/updated in the historical repository, this can be supplied to corresponding variables in the BN model for computing corresponding prior probabilities dynamically.

There are numerous tools available to develop historical databases, however, SQLite is employed as a part of our developed tool because it is a robust relational database management system that is highly efficient and lightweight [19]. It is extensively used in prototyping and the production level software [19]. SQLite offers a remarkably good match for the prototype tool since it is fairly straightforward to be deployed in different environments and has a portable nature. Moreover, SQLite is open-source, free to be used and fully compatible with the ecosystem of our set of tools [19].

6 Demonstration

This section demonstrates the developed tool using an example. The example has been thoroughly illustrated in the following subsections.

6.1 Flowchart of the Prototype Tool

The execution of the tool prototype can be illustrated with the help of a flowchart. The execution commences by loading of the BN file from the user. Moreover, during the first stage, the system loads the CPT values from the Historical Data Repository (HDR). The second stage is for the calculation, the system calculates the posterior probabilities of the outcome node. Following the second stage, there is an option to provide additional evidence. If the user decides to not provide any further evidence, then the system draws the results and the execution comes to an end. However, if the user provides supplementary evidence to the system, then the system returns to the second stage. The calculations are updated, and the execution continues as illustrated in Figure 5.

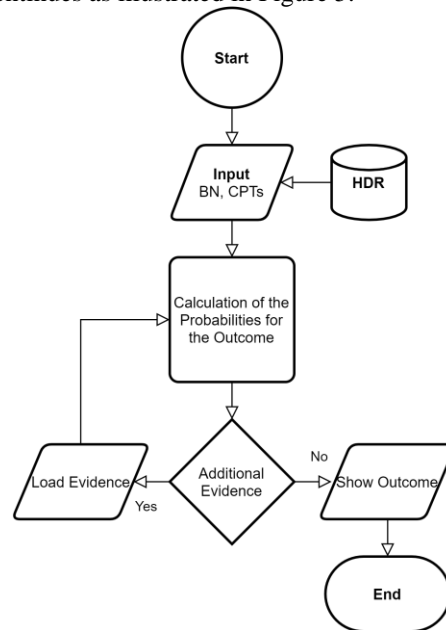


Figure 5: Flowchart of the Prototype Tool

6.2 Fundamental Structure of the Application

The basic structure of the application is built on “Window” control offered by the WPF user control library. The main screen extends MahApps.Metro library, therefore, the user interface is sleek and modern.

As illustrated in Figure 6, there is a button on the top right corner of the window that can be used to load a BN file. Once loaded, the address of the network file is stored in a global variable that is available throughout the application. There is a label towards the top left edge of the application that declares that the Outcome node in a network should have the “Results” as ID consequently the application can accurately unfold its dynamic user interface. Furthermore, there is a couple of buttons marked “up-date” and “reset” that will be discussed in the following sections of the paper.

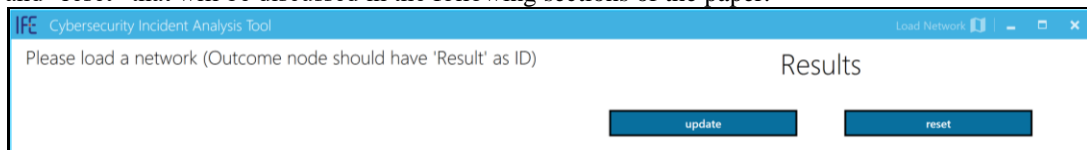


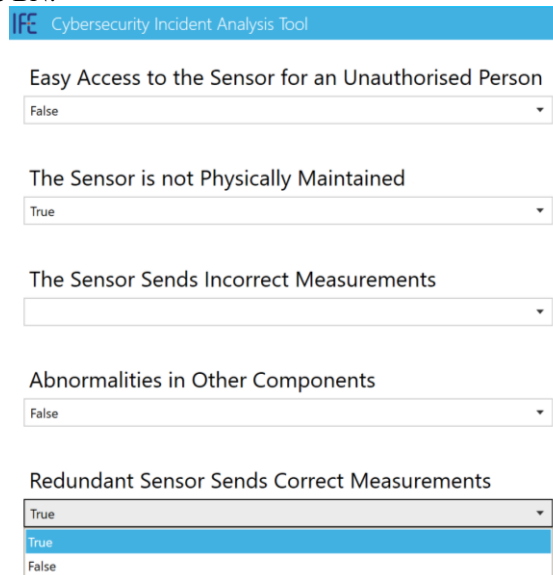
Figure 6: Basic Structure of the Application

The process of prediction is initiated with the loading of the BN into the tool. Once the BN has been loaded, the further calculations and the operations are performed.

6.3 Network File Processing and UI for Nodes

The structure of BN is defined using GeNIe modeler that can be stored in a standalone file. The file is loaded into the tool with the help of “File Picker Dialog” that is a standard WPF control. Once the network file has been loaded using “Load Network” button, A “for loop” goes through all the nodes in the network file to generate respective UI controls/nodes. Each node is rendered in terms of Node Name, Node ID and Outcome IDs. The outcomes can be updated with the help of a ComboBox control. It is a dropdown menu where the user is permitted to set evidence on a particular node. The ComboBox has been equipped with “OnSelectionChanged” event handler that makes the evidence setting process significantly easier. Probability propagation is initiated whenever the evidence is updated without any additional commands. Essentially, the posterior values at the outcome node are similarly updated as a result of the probability propagation, consequently, the graph illustrating the outcome is also updated whenever the evidence is updated.

Figure 7 is used as an example to explain the above-mentioned functionality of the tool to set evidence for different nodes in the BN.



The screenshot shows the 'Cybersecurity Incident Analysis Tool' interface. It features five dropdown menus for setting evidence on different nodes:

- Easy Access to the Sensor for an Unauthorised Person:** Set to 'False'.
- The Sensor is not Physically Maintained:** Set to 'True'.
- The Sensor Sends Incorrect Measurements:** Empty dropdown menu.
- Abnormalities in Other Components:** Set to 'False'.
- Redundant Sensor Sends Correct Measurements:** Set to 'True', with the dropdown menu open showing 'True' and 'False' options.

Figure 7: User Controls for the BN Nodes

6.4 Probability Propagation and Data Visualisation via LiveCharts

Probability propagation is a rather essential feature of the tool. Therefore, there is a particular focus on enhancing the functionality to effortlessly propagate the probability whenever the evidence is updated. There is a ComboBox interface for each node, the ComboBox provides a modest feature that allows the user to choose between the outcomes of the node. Consequently, the evidence is updated.

Two automatic operations are initiated when a user changes the outcome of a particular node. Firstly, the posterior values of the associated node are updated with the help of conditional probability tables and the statistical methods described in Section 2. Secondly, the outcome probabilities of the target node are represented graphically and numerically.

This tool utilises the LiveCharts in order to demonstrate the visual results. LiveCharts is an open-source library that can be applied to draw highly interactive and flexible data visualisations [20]. A pie chart

is utilised to demonstrate the posterior probabilities of the target variable, the numerical values and the visual chart are refreshed automatically. Moreover, the chart is implemented to respond when a cursor hovers around, therefore, making it interactive and robust. The posterior probabilities of the target variable would help operators to perform immediate actions corresponding to the major cause of the problem to minimise the negative consequences. In Figure 8, it is determined that the problem is most likely due to an attack (87%) based on the evidence provided. This could help operators to perform immediate actions in a more informed way and choose effective response strategies by identifying the specific attack-vector used by the adversary to cause this attack.

As shown in Figure 8, there are two more controls accessible to the user. Firstly, the update control is applied to refresh the beliefs for the entire network in case there are some underlying changes to be implemented. The other one is designated to reset the entire network. Once the user chooses to reset the network, entire evidence is lost, posterior values are reset to default, the data visualisation is brought to default as well, and the network is reloaded for further operations.

Results

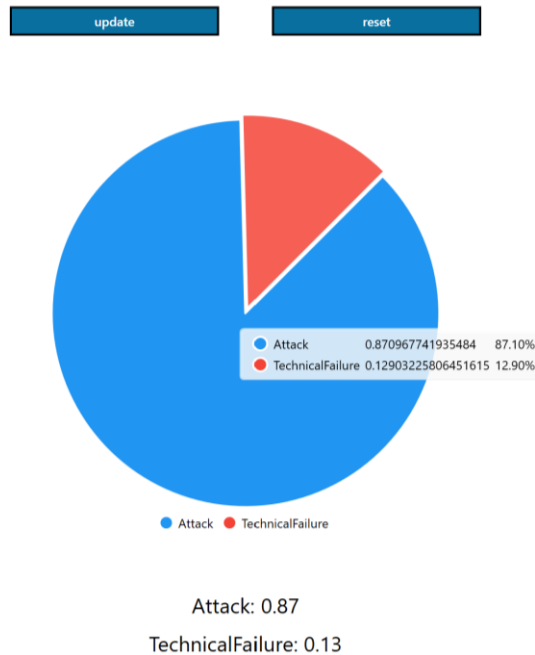


Figure 8: Data visualisation via LiveCharts

6.5 Historical Data Repository

The CPT values in BN models are typically based on historical data and/or expert knowledge. A repository which contains historical information corresponding to attacks and technical failure can be built and integrated with the BN model to meet a core requirement. The CPTs associated with each node in the BN can be dynamically updated based on the data in the repository. This automation makes it easier for the BN modeler by saving time to build such models each time manually.

The Historical Data Repository (HDR) is one of the core components of the tool, this component stores the historical data that can be supplied to the model to enhance the predictive results. Moreover, this component can be utilised to store new data into the repository and implement further Create, Read, Update and Delete (CRUD) operations. The component is based on SQLite Relational Database Management System (RDBMS). As illustrated in the earlier sections, SQLite RDBMS is very

serviceable and harmonious with our collection of tools. Similarly, Entity Framework (EF) Core has been employed to facilitate communication between HDR and SQLite RDBMS. EF Core is an Object-Relational Mapper from Microsoft that simplifies the operations on a database management system [21]. Considering our WPF (.NET Core) structure, EF Core is highly compatible with the environment.

Incident	
PK	Incident Id
	Easy Access
	Physical Maintenance
	Incident Outcome
	Abnormalities
	Redundant Sensor Behavior

Figure 9: Structure of the Incident Entity

As illustrated in Figure 9, the database structure of the example adopted in this paper consists of an Incident entity. The example model consists of numerous attributes that include, Easy Access, Physical Maintenance, Incident Outcome, Abnormalities and Redundant Sensor Behaviour. The Incident ID is utilised to trace the identification of an individual incident in the repository. The Easy Access attribute describes if unauthorised personnel can access the sensor or not. Likewise, Physical Maintenance attribute is employed to identify whether the sensors are physically maintained or not. The Incident Outcome attribute is utilised to describe if the incident caused by an attack or a technical failure. The Abnormalities attribute is used to determine whether there are abnormalities in any other components of the system or not. Finally, Redundant Sensor Behaviour attribute specifies whether the redundant sensor is sending the correct measurements.

```

Incident ID: 102
Easy Access to the Sensor for an Unauthorised Person: 0
The Sensor is not Physically Maintained: 1
Abnormalities in Other Components: 1
Redundant Sensor Sends Correct Measurements: 1
The Sensor Sends Incorrect Measurements (Attack / Technical Failure): 1

<<----->>

Outcome: Attack TT: 0.36                               Outcome: Attack TF: 0.5555555555555556
Outcome: Attack FT: 0.3870967741935484                Outcome: Attack FF: 0.39285714285714285

Easy Access to the Sensor for an Unauthorised Person: 0.4215686274509804

The Sensor is not Physically Maintained: 0.5490196078431373

Redundant Sensor Sends Correct Measurements (Outcome: Attack): 0.5714285714285714

Redundant Sensor Sends Correct Measurements (Outcome: Technical Failure): 0.5

Abnormalities in Other Components (Outcome: Attack): 0.5952380952380952

Abnormalities in Other Components (Outcome: Technical Failure): 0.5333333333333333

```

Figure 10: Data Instances Stored in the HDR (Top: An Individual Incident, Bottom: Probabilities Calculated for CPTs)

The tool provides a console application for user interaction. As shown in Figure 10, the console application displays the saved records (incidents) from the historical repository. Records are furnished

once the console application is launched. Each group of rows comprises the circumstances of a particular incident. Similarly, the console application exhibits the probabilities associated with the different attributes of the incidents. The probabilities are updated whenever a new incident is incorporated in the records.

```

Save a New Incident?
1 ==> Yes           &           0 ==> No
1

Easy Access to the Sensor for an Unauthorised Person?
1 => Yes           &           0 => No
0

The Sensor is not Physically Maintained?
1 => Yes           &           0 => No
0

```

Figure 11: Data Updating in the HDR

Moreover, the console application prompts the user to enter records for a new incident. The user can choose if a new instance of the incident needs to be created. During the creation of an instance, the attributes of the incident can be administered in the binary composition. As shown in Figure 11, it should be assumed that “1” implies “Yes” and “0” indicates a “No”. Concerning the Incident Outcome attribute, “1” and “0” indicate “Attack” and “Technical Failure” respectively. Once the new instance has been stored into the repository, the associated probabilities can be updated.

6.6 Updating Conditional Probability Tables Dynamically Using HDR

One of the core objectives of the tool is to enable the dynamic exchange of information from the historical data repository to the model. This method empowers the model to update its CPT values by fetching data from HDR.

The BN model communicates with HDR by utilising EF Core. The respective probability values are retrieved from HDR by using EF query filters. These filters return the values depending on the provided instructions. Later, these values are mapped on the CPTs. This process is explained in Figure 12.

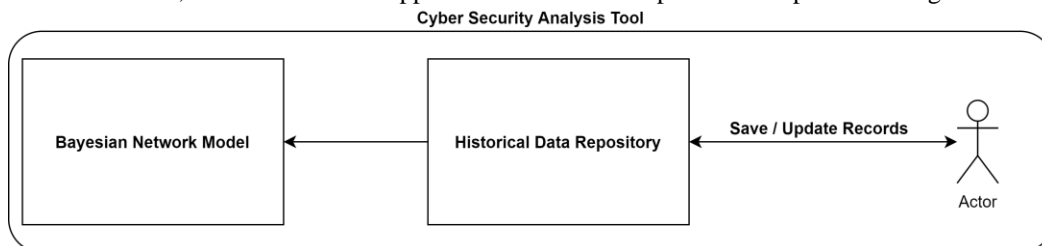


Figure 12: Dynamic Data Transfer from HDR to the Model

The “update” button can be used to retrieve updated data from the HDR. Once the data has been transferred to the BN model, the process of updating CPTs is initiated. Subsequently, the whole model is updated to reflect the changes.

7 Evaluation

The developed tool is tested following the standard Software Development Life Cycle (SDLC). In SDLC, the developed tool is evaluated against the requirements and the potential bugs are resolved [22]. The testing phase is accomplished in two separate phases that consisted of objective and subjective functionality evaluation. Objective (functional) testing is utilised to evaluate the tool against the functional requirements such as computational capabilities, user interface components of the tool. Likewise, subjective testing has been employed to ensure that the subjective requirements have been fulfilled to support the solution. For example, the tool should be easy to use for operators.

During the first phase, the objective functionality is evaluated to ensure that the solution meets the high-level requirements. Objective functionality is associated with the tangible features of the software and the corresponding requirements must be satisfied. The evaluation strategy given below is based on testing the individual requirements. This approach ensures that each requirement has been fulfilled. The requirements have been referred from the requirements elicitation that is Section 3.

(i) The developed tool prototype was evaluated by loading a Bayesian Network file generated by GeNIe Modeler. The developed tool prototype calculated the probabilities to conclude the prediction. These probabilities were compared with the ones generated by GeNIe Modeler to ensure that the calculations are accurate. Moreover, the tool was provided with evidence to analyse the updated predictions. (ii) The HDR was also evaluated by seeding the repository with data corresponding to 100 incidents. The HDR successfully stored the records and generated the probability values which shall be used to update the conditional probability tables (CPTs) in the developed tool prototype. (iii) The integration between the developed BN model using the tool and the HDR was evaluated by updating the CPTs of the model with the dynamically generated probabilities from the HDR. The predictions were updated accordingly that indicated the dynamic flow of data between the model and the HDR. Moreover, to further ensure the accuracy of the solution, the dynamic values produced by the HDR were provided to the GeNIe Modeler to verify the predictions generated by the model. Phase I establishes that the BN model has been implemented and tested for the expected functionality. Similarly, phase II and III imply that the model has been successfully integrated with other components of the developed prototype tool. The overall results of the developed prototype tool were confirmed by third-party tools (GeNIe Modeler).

Similarly, the developed prototype tool is evaluated against the subjective requirements during the second phase of the evaluation. During this phase, we utilised the “monkey testing” technique [23]. Monkey testing is an approach in software testing where all user controls are tested by several combinations of the user inputs [23]. Monkey testing is recognised as a robust, reliable and comprehensive testing technique [24]. A study regarding software testing has concluded that monkey testing can generate more system events than human testing [24]. Since the testing coverage is extensive with monkey testing, this technique guarantees that each user control in the software has been tried and tested using different combinations of user inputs [23]. Therefore, it is very effective at exposing underlying bugs and glitches. Additionally, the developed tool was executed on different screen resolutions on different computers. This is to ensure that it will not fail into different types of computers. Moreover, it was provided with different network files to evaluate the stability further. It verified the capability of the solution to handle diverse BN files. Also, the resultant predictions were compared against different BN tools available in the market i.e. GeNIe modeler. It confirmed that the underlying calculations are executed accurately. During the testing process, each potential bug was resolved to ensure smooth and comfortable user experience. A sizeable portion of the bugs and anomalies was

related to the computations regarding the CPT values. Therefore, the CPT values were calculated independently using third-party tools during the testing phase. Subsequently, the computations taking place in the solution were calibrated with the tried and tested third-party tools to ensure the maximum accuracy and robustness.

8 Conclusions and Future Work

CIs are increasingly dependent on ICSs. ICSs are becoming vulnerable to cyber-attacks and numerous infrastructures underwent drastic economic and technical losses due to cyber-attacks. In addition to the cyber-attacks, problems in ICS can also be caused by technical failures. It is particularly crucial to distinguish between these two types of causes to prompt a reasonable and adequate response to problems in ICS.

There is a framework developed by Chockalingam et al. that is based on the BN to accomplish the task of differentiating the two types of major causes. The solution implements the framework mentioned above to accomplish the goal of identifying the major cause of the problem. However, there is a lack of tool support that can help such BN models to be used in an ICS environment for distinguishing attacks and technical failures, which is the aim of this work.

The developed BN model and HDR comprise the core of the developed tool. The model enables the operators to respond swiftly by simplifying the process of providing additional evidence to the network, thus enhancing the predictability. Furthermore, the developed tool implements HDR that stores historical information concerning attacks and failures. The model can avail from this historical information to improve the analysis of the events. This also support operators in choosing an effective response strategy to abnormal behaviour in a component of the ICS.

The developed tool prototype possesses limitations which include: The binding between the HDR and the model is static. Therefore, the initial binding between the BN model and the correlated historical data is a technically complicated process. Moreover, the developed tool can only process BN files that possess a predefined, unilateral outcome node. Finally, the developed tool prototype needs further evaluation, which is the case with iterative design process after updating the tool based on the results of our evaluation.

The developed tool can be extended by introducing additional features. It is possible to develop supplementary functionality that enables dynamic binding between the HDR and the model. Consequently, allowing non-technical users to combine the new models and HDRs effortlessly. Likewise, the developed tool can be equipped with a functionality that allows dynamic assignment of the outcome node in a BN.

Acknowledgment

This work is partly funded by the Norwegian Research Council project CybWin (287808).

References

- [1] X. Feng, Q. Li, H. Wang, and L. Sun, "Characterizing Industrial Control System Devices on the Internet," in 2016 IEEE 24th International Conference on Network Protocols (ICNP), 2016: IEEE, pp. 1-10.
- [2] W. Knowles, D. Prince, D. Hutchison, J. F. P. Disso, and K. J. I. j. o. c. i. p. Jones, "A survey of cyber security management in industrial control systems," vol. 9, pp. 52-80, 2015.

- [3] J. Slay and M. Miller, "Lessons Learned from the Maroochy Water Breach," in *International Conference on Critical Infrastructure Protection*, 2007: Springer, pp. 73-82.
- [4] R. M. Lee, M. J. Assante, and T. Conway, "German Steel Mill Cyber Attack," *Industrial Control Systems*, vol. 30, p. 62, 2014.
- [5] T. Macaulay and B. L. Singer, *Cybersecurity for Industrial Control Systems: SCADA, DCS, PLC, HMI, and SIS*. Auerbach Publications, 2016.
- [6] S. Chockalingam, W. Pieters, A. Teixeira, N. Khakzad, and P. Van Gelder, "Combining Bayesian Networks and Fishbone Diagrams to Distinguish between Intentional Attacks and Accidental Technical Failures," in *International Workshop on Graphical Models for Security*, 2018: Springer, pp. 31-50.
- [7] C. Basile, M. Gupta, Z. Kalbarczyk, and R. K. Iyer, "An Approach for Detecting and Distinguishing Errors versus Attacks in Sensor Networks," in *International Conference on Dependable Systems and Networks (DSN'06)*, 2006: IEEE, pp. 473-484.
- [8] A. Anwar, A. N. Mahmood, and Z. Shah, "A Data-driven Approach to Distinguish Cyber-attacks from Physical Faults in a Smart Grid," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, 2015, pp. 1811-1814.
- [9] S. Chockalingam and V. Katta, "Developing a Bayesian Network Framework for Root Cause Analysis of Observable Problems in Cyber-Physical Systems," in *2019 IEEE Conference on Information and Communication Technology*, 2019: IEEE, pp. 1-6.
- [10] I. Ben-Gal, "Bayesian Networks," *Encyclopedia of Statistics in Quality and Reliability*, vol. 1, 2008.
- [11] J. Ding, "Probabilistic Inferences in Bayesian networks," In: A. Rebai (ed.), *Bayesian Network*, pp. 39-53, 2010.
- [12] S. Chockalingam, W. Pieters, A. Teixeira, and P. van Gelder, "Bayesian Network Models in Cyber Security: A Systematic Review," in *Nordic Conference on Secure IT Systems*, 2017: Springer, pp. 105-122.
- [13] A. Hevner and S. Chatterjee, "Design Science Research in Information Systems," in *Design Research in Information Systems*: Springer, 2010, pp. 9-22.
- [14] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A Design Science Research Methodology for Information Systems Research," *Journal of Management Information Systems*, vol. 24, no. 3, pp. 45-77, 2007.
- [15] "Getting Started - WPF." <https://docs.microsoft.com/en-us/dotnet/framework/wpf/getting-started/> (accessed 27/07/2020.)
- [16] P. Jenkins. "MahApps.Metro. A Library to Develop User Friendly WPF Interfaces." <https://github.com/MahApps/MahApps.Metro> (accessed 27/07/2020.)
- [17] "Bayesian Network Modelling with GeNIe Modeller." <https://www.bayesfusion.com/genie/> (accessed 27/07/2020.)
- [18] "SMILE Engine for Bayesian Network Modelling." <https://www.bayesfusion.com/smile/> (accessed 27/07/2020.)
- [19] "SQLite RDBMS. A Modern & Lightweight Relational Database Management System (RDBMS)." <https://www.sqlite.org/about.html> (accessed 27/07/2020.)
- [20] "Live Charts.Implement Lively Graphical Illustrations." <https://lvcharts.net/> (accessed 27/07/2020.)
- [21] "EF Core - Overview of Entity Framework Core." <https://docs.microsoft.com/en-us/ef/core/> (accessed 27/07/2020.)
- [22] G. D. Everett and R. McLeod Jr, *Software Testing: Testing Across the Entire Software Development Life Cycle*. John Wiley & Sons, 2007.
- [23] T. Wetzlmaier, R. Ramler, and W. Putschögl, "A Framework for Monkey GUI Testing," in *2016 IEEE International Conference on Software Testing, Verification and Validation (ICST)*, 2016: IEEE, pp. 416-423.
- [24] M. Mohammed, H. Cai, and N. Meng, "An Empirical Comparison between Monkey Testing and Human Testing," in *Proceedings of the 20th ACM SIGPLAN/SIGBED International Conference on Languages, Compilers, and Tools for Embedded Systems*, 2019, pp. 188-192.