applied sciences

MDPI

Article

Towards Optimized Security Attributes for IoT Devices in Smart Agriculture Based on the IEC 62443 Security Standard

Abdelkader Magdy Shaaban [1,*], Sebastian Chlup [1], Nahla El-Araby [2,3] and Christoph Schmittner [1]

[1] Center for Digital Safety & Security, AIT Austrian Institute of Technology GmbH, Giefinggasse 4, 1210 Vienna, Austria; sebastian.chlup@ait.ac.at (S.C.); christoph.schmittner@ait.ac.at (C.S.)
[2] Institute of Computer Technology, Technical University of Vienna (TU Wien), Gusshausstrasse 27-29, 1040 Vienna, Austria; nahla.el-araby@tuwien.ac.at or nahla_alaraby@cic-cairo.com
[3] Electrical and Electronics Engineering Department, Canadian International College (CIC), Cairo 11835, Egypt
* Correspondence: abdelkader.shaaban@ait.ac.at

Abstract: Implementing applicable security measures into system engineering applications is still one of the most challenging processes in building secure infrastructure. This process needs to consider a variety of security attributes to support securing system components against numerous cyberattacks that could exploit vulnerable points in the system. The redundancy in these attributes is also another challenge that could degrade system functionality and impact the availability of the system's services. Therefore, it is crucial to choose appropriate security properties by considering their ability to address cyber threats with minimal negative impacts on the system's functionality. This process is still subjected to inconsistencies due to ad- oc determinations by a specialist. In this work, we propose a novel algorithm for optimizing the implementation of security mechanisms in IoT applications for the agricultural domain to ensure the effectiveness of the applied mechanisms against the propagation of potential threats. We demonstrate our proposed algorithm on an IoT application in the farming domain to see how the algorithm helps with optimizing the applied security mechanisms. In addition, we used THREATGET to analyze cyber risks and validate the optimized security attributes against the propagation of cyber threats.

Keywords: security measures; potential threats; attack propagation; IoT; cybersecurity; security standard

Citation: Shaaban, A.M.; Chlup, S.; El-Araby, N.; Schmittner, C. Towards Optimized Security Attributes for IoT Devices in Smart Agriculture Based on the IEC 62443 Security Standard. *Appl. Sci.* **2022**, *12*, 5653. https://doi.org/10.3390/app12115653

Academic Editor: Stefan Fischer

Received: 27 April 2022
Accepted: 26 May 2022
Published: 2 June 2022

1. Introduction

Cyber-physical production systems (CPPSs) are one of today's major technological driving factors. A wide range of current technologies, e.g., industrial automation, the Internet of Things (IoT), robotics, big data and cloud computing are incorporated within current CPPSs [1]. CPPSs consist of widely dispersed components which each contribute to the overall industrial systems. Therefore, the IoT is a novel upgrade of industrial automation control systems (IACSs). It benefits from the communication between billions of devices and enables sophisticated data processing methods over the Internet [2]. Formerly, corporate networks for production planning, ordering of materials or job scheduling were separated from IACS networks. Due to the required communication ties to the outside world, this top-level company information technology infrastructure was already vulnerable to external attacks. Moreover, IACS networks encompassing corporate networks and proprietary IACS networks have been replaced with commercial-off-the-shelf (COTS) equipment which is based on Ethernet TCP/IP. Therefore, cyber risks also affect the security of IACS networks. Furthermore, total system complexity has skyrocketed in recent years, and new threats are being introduced, jeopardizing the controllers within IACS networks, as these COTS components were never designed to work with them. In many circumstances, replacing them with attack-proof controllers is neither possible nor supplied by the machinery provider. Moreover, this would also disrupt the industrial the continuous operation of the production process. Multiple components are interacting with each

other and communicating over networks or the public Internet. This means cybersecurity risks, potential exploits and potential failures. Therefore, a company must ensure that the confidentiality, integrity and availability (CIA) of data or system components are secured at all times in order to guarantee functionality and privacy at all times. Threat modeling has become an important tool in identifying potential threats to a system, including risk management and mitigation measures. In recent years, AIT has developed an automated approach towards threat modeling in the cyber-physical domain and the spreading of threats throughout a technological system, based on a system model, a threat database and security attributes.

In this paper, we will not only explain the concept of threat modeling but even go one step further and identify the low-hanging fruit within the analyzed model and enhance the results with the requirements of the IEC62443 standard. This is to identify certain "gate" components where a threat can be mitigated before being propagated throughout the parts of the system. Thus, we introduce an approach to find the minimum number of security measures yielding the maximum security impact by considering the effectiveness of the optimized security attributes against multiple types of cyber incidents. Consequently, we seek to apply security measures at relevant points within the system, and thus mitigate the risk for the following components while omitting the necessity to implement the same security measure at all of the components. Formal verification (B-method) was used to evaluate this approach and ensure that no violations for security measures over the whole system exist. B-method can verify the model's consistency through extracting proof obligations that must hold through all operation states. Security measures were embedded in the B-model as invariants.

2. Materials and Methods

2.1. Cyber Threats

We live in a highly networked culture in which human–computer interaction is essential. Intrusion can have a negative impact on safety, finances, operations, and performance. Our technology is becoming increasingly complex, and as a result, more difficult to maintain. The larger the number of technological components, the greater the attack surface [3,4]. As a result, our products necessitate a high level of agility in security. The static approach to security measures and analysis at design time is out of date and cannot keep up with the constantly evolving new generation of threats as they get more diverse and complex.

Agriculture is distinct from industrial control systems due to outdoor field areas, which demand the implementation of different types of security measures. Cybersecurity standards and related regulations are insufficient to introduce an agriculture infrastructure that is secure from current and future cyberattacks [5]. Kristen et al. [5] introduced initial recommendations for addressing cyber vulnerabilities and securing an agriculture domain.

We regard cybersecurity as a vital component of system development lifecycles, especially in critical infrastructures. Cybersecurity is in charge of safeguarding assets from cyberattacks such as integrity breaches, data leakage, and other destructive actions. Most workplaces have changed to a remote work force infrastructure; the worldwide COVID-19 restrictions contributed heavily to this [6]. In order to design a secure system, it is necessary to prevent future cyberattacks by learning from previous cyber incidents. Therefore, we consider threat modeling as an effective method for discovering system security risks and vulnerabilities [7]. It incorporates risk management approaches and the methods for threat identification which we elaborate on in the following sections.

2.2. Risk Management

The goal of risk management is to detect, assess, and evaluate hazards in a given setting and then address them. This method is intended to lessen the risk for a given system [8].

Adding, configuring and maintaining security in an application always remains a trade-off between effort, effects, exploitability, and the costs associated with it. As a result, risks must be addressed at all levels of an organization's structure. Who is responsible for decision making and how choices are made must be transparent [9].

As the environment and parts of the application may undergo changes and adaptions, risk management is regarded as an iterative process. New threats develop on a regular basis, and established threats may evolve or even vanish. As a result, the system under examination must be regularly monitored as new hazards to its operation are identified. Furthermore, a good risk management strategy aids in the discovery of mitigation methods and informing choices to enable prompt reactions. This is why it is crucial to integrate risk management in conformance with the system's objectives and to discover ways to tackle conflicting objectives. Risk management consists of two major constituents, risk assessment and risk treatment, which are discussed in the following Sections [9].

2.2.1. Risk Assessment

The risk assessment process can be divided into three stages: risk identification, risk analysis, and risk evaluation [9,10].

Risk Identification deals with the determination of risks or threats that are interfering with an application's objectives. It explores potential risk factors, vulnerabilities, and threats [9].

Risk Analysis is conducted once risks have been identified. It focuses on the exploitability of risk related events and their impacts on the system. However, the outcomes of such an analysis may differ depending on the application domain, viewpoints, and individual backgrounds [9].

Risk Evaluation builds on the results from the risk analysis. It is necessary in identifying which risks must be addressed and how they must be addressed in terms of the techniques used [9]. However, there may be various solutions to a certain challenge. A strategy addressing even multiple weak spots may be suggested. The purpose of a risk assessment is to aid in decision making. While certain hazards require more investigation, others may not demand specific activity. Consequently, the assessment may disclose circumstances that do not satisfy the intended goals, and as a result, these goals must be re-evaluated.

Once the risk assessment procedure has been completed, measures must be implemented for the evaluation results to take effect. This is accomplished through a procedure known as risk treatment.

2.2.2. Risk Treatment

Once potential risk-reduction measures have been developed, professionals must decide how to address specific risks. This entails assessing the efficacy of various treatment methods. If the residual risk is unacceptable, other treatment options may be examined, or additional treatment methods will be applied [9].

When deciding on the risk treatment, four options may be considered [11]. Figure 1 holds an illustration of abstract risk treatment measures inspired by ISO 27005 [12].

One may decide to avoid a risk when it has a high likelihood or a high potential impact. This could be done by modifying the system under consideration (SuC) in a way that eliminates the risk's cause, involving potential removal of services.

It is also possible to transfer the risk, e.g., to other organizations. In cases where the likelihood is low but the impact remains high, this may be the preferred solution (e.g., insurance or third-party providers).

Sometimes it may be sufficient to accept the risk if likelihood and impact are low, and the consequences are therefore considered negligible.

The final option for risk treatment is the mitigation of risks. If the impact is low but the exploitability is regarded as high, a modification with certain countermeasures can reduce the impact or the exploitability even further.

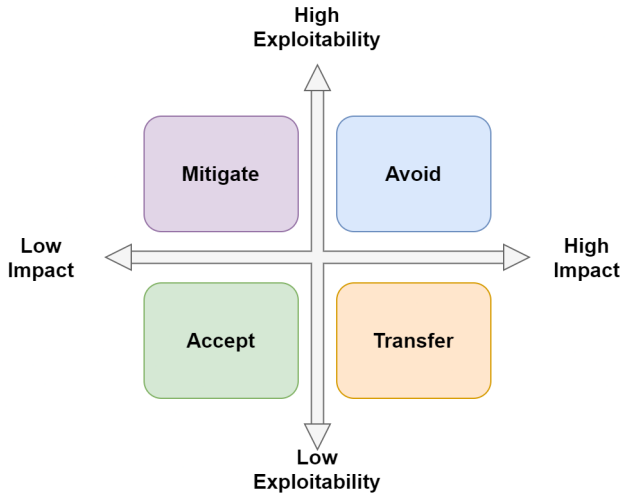


Figure 1. Risk treatment options.

What must be considered is that the outlined methods for risk treatment are not universal. Sometimes, transferring or avoiding the risk is not an option, as it would interfere with the intended objectives. Due to the work necessary for a successful attack, the likelihood of the system under evaluation being subject to it is sometimes nearly negligible, which may result in risk acceptance, despite a significant potential impact. However, the presented approach is applicable to a variety of cases.

2.3. Security Requirements According to ISO/IEC62443

The primary purpose of the IEC 62443 series is to provide a framework for addressing existing and potential cybersecurity vulnerabilities in industrial systems. For the whole system lifecycle and all layers of network infrastructures, it provides security risk management [13]. These include partitioning the system into security zones, specifying a security level for each zone, and describing security capabilities that enable a component to be integrated at a particular security level (SL). The IEC 62443 standard includes thirteen documents organized into four major categories: General, Policies and Procedures, System, and Component [14]. The first two categories include concepts, use cases, policies, and processes related to industrial control system security (ICS security). Technical requirements for networks and components are defined by the other two groups, namely, System and Component [15]. The standard proposed to build a set of distinct security boundaries is called a security zone; each has a set of common security requirements defined to achieve a particular security target. Another type of zone proposed by this standard is called a conduit, which is responsible for aggregating communication channels and building secure mediums between zones [16].

2.3.1. IEC 62443 Categories

The standard is based on four main categories; these categories are described in the following section.

Category1—General

The first category of this series is General, which includes discussion and topics common throughout the entire series [14]. The second category of this series is Specific. The International Electrotechnical Commission (IEC) Technical Specification 62443-1-1 provides the terminology, concepts, and models for IACS security. The descriptions of terminology and acronyms used during the IEC 62443 standards are provided in IEC/TS 62443-1-2. There are seven foundational requirements (FRs) presented in this standard as follows [17]:

FR1: Identification and Authentication Control (IAC): The primary goal of the identification and authentication control is to verify a user's identity before providing permission to access a system or a resource. Depending on the circumstances, this permission could be

granted to humans, processes, software, or hardware devices that require authorization to access a system [18].

FR2: Use Control (UC): This fundamental requirement ensures that only authorized users have access to the system. Its purpose is to prevent unauthorized system activities by verifying that the requested permission has been granted before enabling any entities to initiate communicating with the system [18].

FR3: System Integrity (SI): Individuals, procedures, software, and hardware are all examples of authorized entities who may attempt to compromise any component of a system's security integrity [18].

FR4: Data Confidentiality (DC): In order to prevent the illegal disclosure of data over communication channels or data storage in repositories, data confidentiality is an essential consideration [18].

FR5: Restricted Data Flow (RDF): Security zones and conduits for communication channels are used to restrict the flow of unnecessary data by establishing security borders, which are also known as security zones [18].

FR6: Timely Response to Events (TRE): Create reports to respond to any malicious activities detected on a system [18].

FR7: Resource Availability (RA): Protect against various kinds of denial-of-service attacks so that a system remains operational [18].

Category2—Polices and Procedures

This category deals with the security of the IACS, and it provides security requirements that can be applied to determine the amount of protection provided by operational IACS [14]. The IEC 62443-2-1 specifies who owns an asset for IACS and provides the parameters for developing and maintaining a cybersecurity strategy. This series covers the capabilities of the protection system that are required to ensure the proper operation of an IACS. IEC/IS 62443-2-2 defines a methodology and framework for an IACS for assessing defense in accordance with the security level and ensuring that the corresponding processes are implemented. The third series in this category is IEC/TR 62443-2-3, which provides information about the exchange of information from asset owners to product suppliers. The next series provides information about the IACS service provider and asset owner [17].

Category3—System

The IEC 62443-3-1 standard gives an overview of the advantages and drawbacks of currently available network security solutions. The IEC 62443-3-2 standard addresses the assessment of security risks and the design of network architectures and systems. Part 3-3 of the standard defines broad system security requirements, with an emphasis on ensuring that system performance is not compromised during the process of fulfilling these requirements [19].

Category4—Component

It comprises two documents that make up the Component group. IEC 62443-4-1 specifies the development procedure for ICS products to restrict the number of security vulnerabilities in control systems solutions. It is specified in the IEC 62443-4-2 standard how to secure the separate components of an ICS network in terms of their technical requirements.

2.3.2. Zones and Conduits Concept

The IEC 62443 security standard emphasizes performing a security analysis of a manufacturing facility. The facility is broken up into sections that have been given the name "security zones" [20]. In addition, data flows among connected security zones are suggested in the standard via another type of zones for communication channels; these channels are called conduits [16]. The standard outlines the zones and conduit requirements, abbreviated as ZCRs, for the system under consideration. The SuC entails defining a collection of IACS together with any assets associated with it in order to conduct a risk analysis. IEC 62443

describes the procedures to be followed in order to set up zones and conduits, and their connections to ZCRs [14,21].

According to the IEC 62443-3-2 [21], the ZCRs are described as follows:

- **ZCR1—identification of the SuC:** An identification of the system under consideration (SuC) shall include a specified description of the security limits and an identification of all access points to the SuC.
- **ZCR2—high-level risk assessment:** In order to detect unmitigated risks, a high-level risk assessment of SuC is carried out. This clarifies the worst-case scenario that SuC presents. The assessment assists in the grouping of assets in distinct zones and conduits. High-level risk can be quantified using a risk matrix to define the relationship between the likelihood and the impact values. There are five different levels of parameter values for the likelihood and impact values. In the following, the the likelihood levels are listed:
  - **Level 1**: Trivial
  - **Level 2**: Minor
  - **Level 3**: Moderate
  - **Level 4**: Major
  - **Level 5**: Critical

  The next listing shows the possible impact values:
  - **Level 1**: Remote
  - **Level 2**: Unlikely
  - **Level 3**: Possible
  - **Level 4**: Likely
  - **Level 5**: Certain
- **ZCR 3—Partition the SuC into zones and conduits:** This phase splits up the complex overall system into separate zones and conduits. Figure 2 illustrates an example of splitting the system into distinct zones and a conduit.

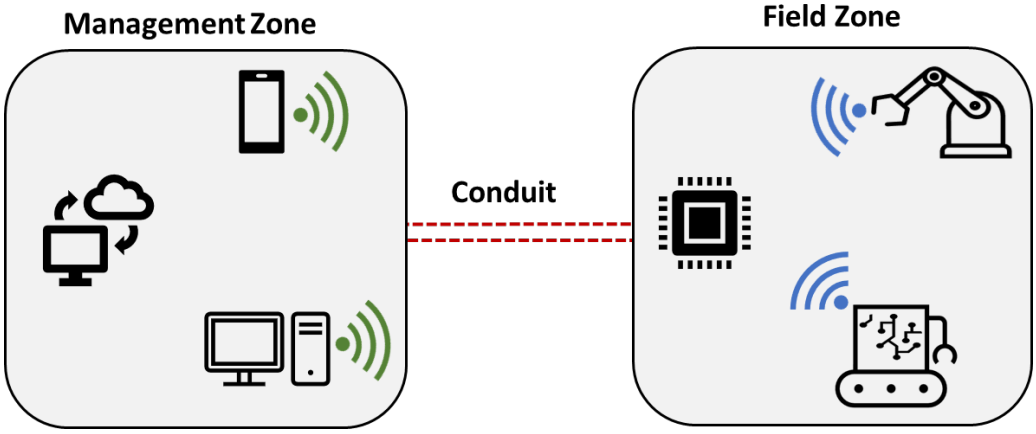


Figure 2. Example of SuC partitioning in zones and a conduit.

Figure 2 illustrates components connected wirelessly to provide a particular function. Two security zones are defined to represent a security boundary containing all components that must be protected from multiple cyberattacks. The management security zone represents a system boundary with a set of common security properties to protect monitoring and storage devices, such as cloud servers and computer. The second security zone describes an aggregation of field components, such as actuators and electronic control units (ECU), that need to be protected. Between these two zones, all communication challenges are defined as a part of a conduit that securely creates a protected medium for data transmission.

- **ZCR 4—document cyber security requirements, assumptions and constraints:** This is the last phase to assess the cyber risk for each zone and conduit to individually evaluate target security levels [22].

In the course of our research, we investigated the utilization of the IEC 62443 security standard in terms of guidance for protecting a system model from multiple forms of cyberattacks. The ontology approach was proposed before as one of the most effective solutions for building complete tractability between potential threats (i.e., security issues) and security-related requirements (i.e., security solutions driven by standardization). Shaaban [14] introduced an ontology-based cybersecurity framework that is capable of automating the tracking of threats and related security attributes until the main security objective and its associated security requirements can be specified. In order to address existing security issues, the framework derives a set of security requirements for handling identified potential cyber threats to protect system components/assets. It builds relationships between various entities in the hierarchical ontology model. These relationships are then used to deduce security requirements, which justify why a certain group of security requirements was chosen to address a specific vulnerability in the system. These relationships are constructed in accordance with the security attributes that are collected from threats, components/assets, and security requirements in order to estimate previously established relationships and create new ones, as outlined below [14]:

- *Components/Asset* $\longrightarrow$SecurityAttributes: "A component/asset has security attributes to protect it from any type of cyberattack scenarios"
- *SecurityAttributes* $\longrightarrow$Threat: "Security attributes could be vulnerable points that an attacker can exploit by multiple perspectives, which are defined as potential cyber threats"
- *Threat* $\longrightarrow$Components/Asset: "A threat impacts a component/asset"
- *Threat* $\longrightarrow$*SeverityLevel*: "Each threat has a degree of risk level, that represents the severity level of its cyber risk"
- *SeverityLevel* $\longrightarrow$*SecurityLevel*: "Severity level has an equivalent security level driven by standardization that handles security issues"
- *SecurityLevel* $\longrightarrow$*SecurityRequirements*: "Security requirements are classified according to multiple security levels (i.e., ranging from 1.0 to 4.0)"
- *SecurityRequirements* $\longrightarrow$SecurityAttributes: "A set of security properties can identify security requirements to describe security measures applied to achieve particular security objectives"
- *SecurityRequirements* $\longrightarrow$Component/Asset: "According to that, security requirements are chosen to address/mitigate a threat that aims to protect component/asset"
- *SecurityRequirements* $\longrightarrow$Threats: "Security requirements are chosen to address a threat"

The framework gives a complete, trackable method for proceeding from cyber threats to security requirements. It is an advanced method for automatically estimating an appropriate set of security-related requirements to protect the system components, address potential cyber threats, and mitigate related risks.

2.4. Theory of Threat Modeling

Threat modeling has evolved into a valuable tool for detecting potential safety and security problems. Furthermore, it aids in finding vulnerabilities in the system architecture and can disclose potential design mistakes. *"Threat modeling should be regarded like any other aspect of the design and specification process"*, says Torr [23].

Threat modeling is an iterative process and should be applied during the whole lifecycle of the system, at design, implementation, testing, and maintenance phases. Each of these phases may yield consequences for the system under consideration and can therefore necessitate modifications, which in turn will have an impact on the overall system and its security. AIT follows an approach involving a system model in diagram form and a threat model (threat database) which are compared against each other, based on sophisticated anti-patterns to identify misconfigurations of the system leading to potential vulnerabilities. The result is a threat catalog depicting vulnerable components and related threats. Moreover,

this threat catalog is built utilizing risk assessment methods based on expert knowledge. The following sections discuss the components of threat modeling in more detail.

2.4.1. System Model

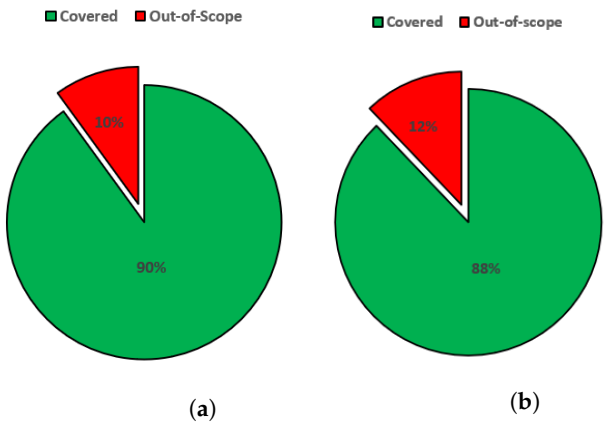
The system model is represented in the form of a data flow diagram (DFD). A DFD can graphically represent a technical system and its security attributes. This DFD provides a description of the components, and their interconnections and setup. The modeled data may be utilized to do an analysis on a specific component or a set of components [23]. Even the act of converting an application's concept or real-world information into a graphical representation can already affect the system's architecture and identify potential design faults.

2.4.2. Threat Model

One of the essential parts of threat modeling is building an up-to-date threat database that consists of a wide range of cyber threats based on the state-of-the-art. This database supports the risk analysis process to deeply investigate all possible threats that could be initiated from one or multiple existing security vulnerabilities in the system model. The threat database holds the threat model which contains known flaws, vulnerabilities, and weaknesses to be used for analysis in order to detect threats inside the system model. Potential dangers can be found by automatically comparing the threat model with the system model [3]. Using a database for storage and a rule-based technique for anti-pattern representation, a suitable threat model can be defined. Anti-patterns are used to mimic common configuration problems and are described by domain experts with a domain-specific language (DSL). The phrase anti-pattern refers to a pattern that should not be present in the system. They reflect well-known setup concerns, such as communication over trust boundaries, a lack of encryption while using wireless connection, or a missing of tamper protection in case of physical attacks.

In our study, we built a comprehensive threat database for IoT and CPS application domains based on the IEC 62443 security standard. The newly developed database consists of many potential threats inspired by the standard to give a complete image of the present and future security issues. This database mainly focuses on parts 3-3 (i.e., system requirements) and 4-2 (i.e., component requirements). Figure 3 illustrate the number of developed threats according to the coverage of these two parts (i.e., system requirements—part 3-3 and component requirements-part 4-2) of the standard.

According to our estimations, we addressed over 90% of the security requirements that could be used to define a set of potential threats from the IEC 62443-3-3 standard, as illustrate in Figure 3a. In addition, about 88% of the security requirements in IEC 62443-4-2 are utilized to identify a set of potential threats that may be encountered, as depicted in Figure 3b.



(**a**) (**b**)

Figure 3. Threat coverage of developed threat database according to the IEC 62443 security standard. (**a**) The coverage of threat database according to IEC 62443 part 3-3 (i.e., system requirements). (**b**) The coverage of threat database according to IEC 62443 part 4-2 (i.e., component requirements).

The threat modeling approach can be summarized in five main steps [3]. Figure 4 illustrates these steps in terms of how the threat modeling approach includes designing the whole system with all security-related information until all potential threats are addressed.

The diagram shows that the process of threat modeling begins with building the entire system model, including all of the relevant security information that must be integrated into the design of the system. After that, the next step is to perform a threat analysis in order to determine the possible threats that already exist and how they can exploit any security vulnerabilities that may be present. In order to provide clear knowledge of the best risk treatment plan for addressing identified threats, it is essential to evaluate every threat that has been identified. In addition, all of the identified threats can be mitigated by keeping the system model up to date with appropriate security measures. Consequently, the threat analysis (i.e., step 2) shall be performed to examine the effectiveness of the previously applied security measures and determine new potential risks could be propagated due to the newly updated measures or if any existing ones need more security concerns.

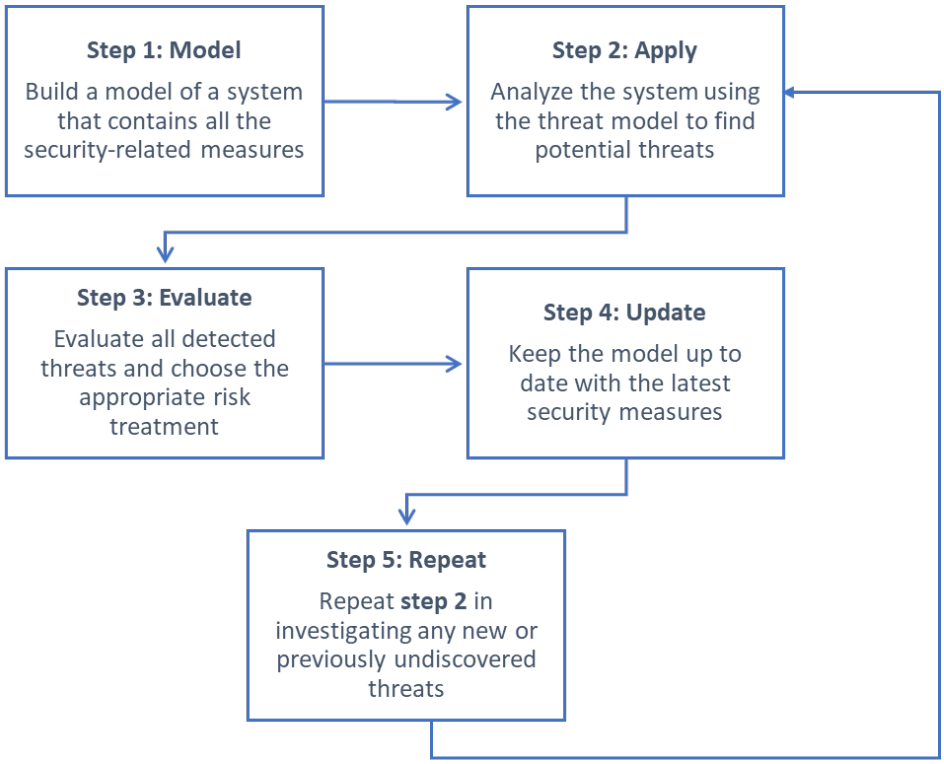


Figure 4. Main steps of the threat modeling approach.

2.4.3. STRIDE

The basis of threat modeling is a brainstorming methodology referred to as STRIDE. STRIDE is the abbreviation of the **S**poofing, **T**ampering, **R**epudiation, **I**nformation Disclosure, **D**enial of Service, and **E**levation of Privilege. It was invented in 1999 and adopted by Microsoft (www.microsoft.com, accessed on 15 May 2022) in 2002 [24] and is used to identify threats for different threat categories. These threats categories are defined as follows [7]:

- **S**poofing: Attempting to get unauthorized access via a false identity.
- **T**ampering: Aiming to modify data with an unauthorized method.
- **R**epudiation: Denying an action that a legal/illegal user conducts.
- **I**nformation disclosure: Revealing confidential data.
- **D**enial of service: Making a specific service, system, or application unavailable.
- **E**levation of privilege: A user with restricted access rights receives more elevated privilege than they should.

STRIDE can be used together with a the data-flow diagram (DFD) to describe the system entities, events, and the multiple levels of system zones [24]. As discussed in [3], there are five basic annotations are utilized to clarify the system elements using DFD:

- **Process:** represents the system elements, input, output, and actions.
- **Data store:** represents a data storage or databases, which indicate the sources of the system's data.
- **Data flow:** represents the flow of the information between the different system entities that could represent a communication protocol.
- **External interactions:** indicates external elements that interact externally with the system elements.
- **Trust boundaries:** are used to divide the system structure into separate trust zones.

While developing software or hardware applications, the STRIDE approach helps to identify potential threats during the all system lifecycle phases. When it comes to executing threat modeling, STRIDE has two different approaches: per-element and per-interaction [25].

STRIDE Per-Element

This approach observes the threats according to applicable elements in the diagram. It concentrates on a certain set of threats for each specified system element. According to Table 1, Microsoft uses this table as a core of its security development lifecycle [26,27].

Table 1. Microsoft's STRIDE-Per-Element [26,28].

	S	T	R	I	D	E
External Entity	X		X			
Process	X	X	X	X	X	X
Data Flow		X		X	X	
Data Store		X	?	X	X	

The "X" in Table 1 represents the class of threat that the element category could trigger. "?" means that the logging data store element is applied in addressing the repudiation, but in some cases, the logs could be used as a way for a repudiation attack. Consider the following scenario: there is a network data flow, and an attacker has access to the same network. In this situation, attackers can read, change, or broadcast a flood of packets in order to prevent a specific service from being performed [26].

STRIDE Per-Interaction

The second approach of the STRIDE method that identifies threats against the tuples: origin (i.e., source), destination (i.e., target), and interaction. This approach gets the same number of threats as STRIDE-per-element. However, the STRIDE per-interaction is more manageable because the threats could be more straightforward to comprehend than the STRIDE-per-element. This approach requires a reference chart of software to obtain the results [26,27].

2.4.4. Threat Analysis

An analysis may be performed once the system model and threat model have been completed. The system model is compared to the threat model to identify and reveal threats defined within the threat model. As a consequence, a list of threats targeting certain components, communication channels, or their compounds has been created. The threat catalog provides information about the threat´s impact and likelihood, and the STRIDE category. These data may be utilized to assess the identified dangers and provide the basis for risk management. It aids in determining which dangers—and as a result, risks—should be addressed, and which should be avoided, transferred, or accepted.

2.4.5. Security Attributes

When it comes to information security, security attributes are defined as abstractions representing an entity's fundamental properties or characteristics in terms of information protection. Security attributes are generally related to the internal data structures (documents, buffers, etc.) within the information management system. The information security policy is supported by security attributes, which enable the fulfillment of access control, data flow, data processing, distribution instructions, and support additional characteristics of the information security policy [29].

A set of security attributes is selected from the IEC 62443 security standard to define a set of the most common security measures that need to be part of system components. The attacker is always looking for multiple malicious activities to exploit existing security measures to reach malicious goals. Therefore, the threat modeling approach could support discovering these security vulnerabilities in the system design phase to predict similar cyberattack scenarios in the actual operation of the system. Table 2 gives an overview of the selected security attributes based on the IEC 62443 security standard in order to describe a set of security measures for protecting system components.

Table 2. An excerpt of the proposed security attributes for system components [29].

Attribute	Description
Authentication	Verifying the identity of a user, process, or device
Authorization	Dedicated management of roles and access right to allow access to resources on a target system.
Input Validation	Input validation helps to ensure correct and accurate inputs and prevent attacks such as cross-site scripting and a variety of injection attacks.
Input Sanitization	Input sanitization helps to correct invalid or security critical inputs to prevent potential injection attacks.
Tamper Protection	Protection against physical attacks.
Secure Boot	A method of securely booting the system with only OS relevant operations. No third party applications are executable.
Encryption	Encrypting the data to prohibit information disclosure and to protect data confidentiality.
DDoS Mitigation	Mechanisms to prohibit DDoS attacks. This means allowing only a certain amount of traffic, reducing the attack surface, defining firewall rules etc.
Intrusion Detection	Applying network scanning tools, end-point protections, dedicated intrusion detection or prevention systems.

2.4.6. Security Optimization Algorithm

Protecting the network systems of big enterprises introduces a variety of major difficulties due to the complexity of the configurations. The management of these networks demands analysis techniques that are both robust and comprehensive. An administrator of a network needs to be aware of both the configurations that are susceptible to attacks and the technologies that can be used to protect the networks [30]. It is essential to provide sufficient security mechanisms to protect the system from multiple security issues. Today's technology suppliers are well aware of the significance of taking precautions to protect users' data, which is why there has been an increase in the number of security solutions available on the market [31]. However, due to the high cost of security measures, most customers choose the less expensive options rather than pay the additional cash for more robust versions. This cost can be quantified in cash for the security modules. However, additional overhead expenses are associated with performance and power consumption [32].

In addition, there is a significant part played by security optimization, which involves doing an analysis of the security system of an organization, locating any vulnerabilities, and plugging them using appropriate security measures [33]. However, the main challenge lies in maintaining a high level of security within a particular system while simultaneously optimizing security measures without compromising the measures that have been optimized. Almohri et al. [30] presented a probabilistic graph model and algorithms for studying the security of complicated systems with the end goal of minimizing the chance of successful cyberattacks. This research was based on an optimization technique known as sequential linear programming, which has been extensively used and researched in a wide variety of engineering applications.

During our investigation into this topic, we primarily focused on the outcomes of the threat analysis. These outcomes assist in deeply investigating the existing security issues that need more security solutions. Based on these findings, we determined common security attributes applied for system components that need to be optimized to maximize their resistance to the various kinds of threats currently present within systems' networks. Furthermore, we propose an optimization algorithm that examines each component in a system and aggregates security attributes, if applicable, in order to resist any attack propagation that could spread among multiple connected components within a particular network. However, adding the security attributes to all components is not very efficient. Therefore, we introduce an algorithm for identifying the minimum number of security attributes while still covering all attributes required by the components. The proposed algorithm is defined as follows:

1. M is the THREATGET model.
2. C is the component.
3. SP is the security property.
4. $M =< C, SP >$.
5. connectsTo(c1,c2): "c1 as a source component connects to c2 as a target component."
6. protectedBy(c,sp): "c is protected by a set of security properties."
7. protected(c) = {sp | protectedBy(c,sp)}.
8. incomingCovered(c) = $\bigcap_{c1.connectsTo(c1,c2)} covered(c1)$.
9. covered(c) = $(incomingCovered(c)) \bigcup protected(c)$.
10. final(c) = $protected(c) \setminus IncomingCovered(c)$.

The algorithm is mainly based on the results from the THREATGET tool. Therefore, M in *line 1* represents the model of THREATGET, which includes a set of components, their interconnections, their security properties/attributes, and a group of potential threats for each affected component. The algorithm checks the data flow for pairs of connected components in order to determine the security attributes that need to be covered in each system component for the data flow path. Therefore, *line 6* represents *c1* and *c2* as a source component connected with a target component. As previously discussed, each component has a set of security attributes defined as defensive measures against multiple cyberattack scenarios. Therefore, the protectedBy (i.e., *line 7*) indicates that a set of security properties are defined for the current component *c*. Furthermore, the algorithm creates a set of current security attributes by which the component is *protected*. The algorithm creates a set of attributes that are determined based on the intersection of all covered security attributes of all previous data flows (i.e., the *incomingCovered* set), as illustrated in *line 10*. In *line 11*, the algorithm creates another set of attributes based on the union of all previously textitcovered sets with the ones from the current components (i.e., *protected*). *Line 12*, represents the algorithm only selecting the security attributes that are not already covered by the prior system's component in a particular data flow (i.e., *protected* set minus *incomingCovered*).

2.4.7. Formal Verification

Formal proofs increase the trustworthiness of verified systems. The state-of-the-art shows various attempts to formally verify security properties. Ruet in [34] provides the formal security proofs of three standards that are formally verified using the proof assistant

EasyCrypt. Andronick et al. [35] presented a method for verifying the security properties of source code embedded on smart cards, through combining functional verification at the source code level and the verification of high-level properties on a formal model built from the program and its specification. Model checking is used in [36] to develop techniques that support the development of tools to solve analysis problem instances in trust management policy. In hardware, formal verification is also needed. The work presented in [37] proposes a method to formally verify the correctness and the security properties of a network on chip (NOC) router in order to provide the proper communication functionality and to avoid NOC attacks. Detailed surveys on the use of formal verification for enhancing the security in different levels across the system stack exist in [38–40]. It is shown in the literature that formal methods and verification provide improved results in resilience against adversaries. In this work, we chose the B-method as a formal verification method to prove that the security properties of the optimized system still hold after optimization, and accordingly, that the proposed algorithm is not deteriorating the system security.

Incorporating formal models into the design process results in more robust systems. In order to increase trust in the system's functionality, researchers study a wide range of modeling and verification technologies. The fundamental goal of the verification process is to investigate a model for the accuracy of functional requirements stated as formal properties. The role of formal verification is to ensure the system being designed will satisfy certain properties. Formal verification ensures the correctness of the system in early design stages and consequently improves its reliability. Verification, validation, and hazard analysis procedures on critical systems are usually carried out in separate stages of system development and by different teams of engineers. However, formal verification can be integrated with design tools to verify certain properties are satisfied at different stages and also at run-time. Theorem proving and model checking are the two most prevalent methods for automatic formal verification. The formal description can be used to guide further development activities; moreover, it can be used to verify that the requirements for the system being developed have been entirely and precisely specified. A variety of formal methods and notations available are available, such as Z notation, VDM, and B-method. Three categories can be used to classify the formal verification methods—equivalence checking, model checking, and theorem proving [41].

Equivalence checking verifies that two designs have the same apparent behaviors (e.g., an optimized design has the same observable behavior as its earlier version). Two distinct types of equivalence checking for digital systems are available according to the type of circuits either combinational or sequential. For combinational systems, the two outputs are passed into canonical representations, and if the representations are identical, the two systems are equivalent.

In model checking, an automatic process is used to check if a system satisfies a known property. The model checking process starts by describing the system being verified and its required behavior formally. A formal model of the system is build as a collection of finite state machines, and the behavior is modeled as a set of properties (usually defined in terms of temporal logic formulas). Then, the properties' satisfaction levels are checked automatically by exhaustively exploring the state space of the system. The termination of model checking is guaranteed by the finiteness of the model (there is a finite number of states). There are various approaches to model checking. They depend on various technical issues, such as the choice of the language that is used in expressing the system specifications and the representation of the system. Automatic model checking is ideal. If the model was selected properly, the required properties can be verified in one pass. Moreover, available tools for model-checking provide a trace for each error path in the model. The model checker will either terminate with the answer true, indicating the property is satisfied, or give a counterexample that shows why the property is not satisfied. Model checking is not suitable for very large systems, as the state space generally grows exponentially with the size of the transition system. This problem is referred to as state explosion problem. Another drawback of model checking is that it operates only on models, and the actual

implementation might drift from the correct implementation of the model, so the verified model is not equivalent to the actual implementation.

Theorem proving is a pure mathematical or logical approach where the verification problem is described as a theorem in a formal theory. A formal theory is a language in which formulas are written, a set of axioms are developed, and a set of inference rules are used for proving. Theorems can be proved with rules and axioms. A desired property is satisfied if a proof can be constructed from the system axioms and inference rules.

Security properties are all aspects related to the confidentiality of system data. Security policies including authenticity, authorization, secrecy, integrity, freshness, and fair exchange [42] are enforced within a system when security must be verified. Formal verification for security properties verification is an efficient technique to ensuring the security measures are satisfied in various system components and in different design stages [43].

In this work, we used formal verification as part of the evaluation of the proposed algorithm, through ensuring that the security measures hold after applying the algorithm. We used B-method to describe a complete abstract model of the case study presented in this work. Atlier B tool was used to verify the consistency of the described model and to verify surface attack scenarios on the optimized model. In the next subsection, we briefly introduce the B-method.

2.5. B-Method

B-method is a formal specification method invented by Jean-Raymond Abrial [44]. It provides highly precise descriptions of the attributes that specifications demand. The B-method is a model-oriented comprehensive formal method that covers the entire development cycle [45]. The method is based on the mathematical principles of set theory and predicate calculus, and its semantics are given using a variant of Dijkstra's weakest precondition calculus [46]. The B specification consists of a hierarchy of components described using the abstract machine notation (AMN). Each component in a specification represents a state machine where a set of variables defines its state and a set of operations query and modify that state. State transitions are described using generalized substitutions. Constraints on the operation and variable types are described as invariants of a machine. In B-models, abstract machines are the top-level components describing state machines in an abstract way. Refinements represent enriched versions of either abstract machines or other refinements. Implementations represent the ultimate refinement of an abstract machine. The B-method has been shown to be useful in the development of various applications [47,48].

3. Results

This section introduces an example of the agriculture domain as our research case study. The first part represents the risk identification and analysis approaches using the THREATGET tool to identify and rate relevant threats. Outcomes of THREATGET give more detail about the effected elements and the violated security attributes. A set of security attributes is determined for each system component to define which security measures were proposed to mitigate cyber risks. Then, the proposed optimization algorithm is applied to this example to see how these attributes could be minimized without changing the efficiency of the applied security measures. A complete B-model for the system component is described using the B-method. Surface attack scenarios were verified on the optimized model to show that the security measures hold.

3.1. Agriculture Case Study

Figure 2 shows the application of THREATGET and threat modeling on the basis of an agricultural example. The diagram contains a digital twin of automated agricultural components on a generic basis. From bottom to top, the field devices represent components located directly at the fields. These are sensors that conduct humidity, temperature, or PH measurements; or actuators that conduct actions based on these measurements or on a regular basis, such as watering the plants or applying nutrients. Please note that this

diagram represents an abstraction of a real-world system on a generic basis. Therefore, components such as sensors or actuators are only modeled once, since the detected threats would be the same for multiple sensors from a cybersecurity perspective. Specific types of sensors or actuators are out of scope for the presented example. The sensor fusion takes care of correlating measurement data, validating the received data and detecting and correcting discrepancies between different sensors. Once processing at the sensor fusion component has finished, data are forwarded to an edge computing unit, which is a computational device located close to the area of application, which is considered the central processing unit of the whole system. It coordinates multiple field-level components and triggers higher-level actions based on the sensor data provided. These actions are forwarded to the actuator ECU, which then controls the actuators to perform their intended operations. The communication devices, such as the field gateway, serve the purpose of receiving and sensing information from/to that edge computing unit and connect it to a cloud storage for data acquisition, learning, and optimization for the future. Finally, the external environment represents the external interaction with our proposed agriculture case study. This section illustrates that the external interactor communicates with our architecture through two primary paths: cloud storage (i.e., **Path 1**) and sensor (i.e., **Path 2**) data flow. Consequently, we believe that any attack propagation within the system network originates from one of these two paths.

This example is based on the ThreatGet tool, and it includes all of the security attributes required for each system component to build protective security mechanisms for mitigating cyber risks. The results of the tool are introduced in the following section in terms of a set of potential threats, affected elements, and associated security attributes described as system vulnerabilities.

Risk Analysis Outcomes

THREATGET is utilized to identify existing security vulnerabilities that yield negative consequences to the system under consideration. As described in Section 2.4.5, each system component has security attributes that reflect the protection against multiple attack scenarios, such as authentication, authorization, tamper protection, and encryption. In order to provide a set of risk mitigation actions against potential threats, the system architect can specify these characteristics [16]. Therefore, THREATGET checks for violations of these characteristics and indicates if there is a security vulnerability that could allow malicious activity to take place the system model shown in Figure 5.

Based on THREATGET's database discussed in Section 2.4.2, the tool deeply investigated all known security issues. The tool identified 83 threats which were classified into six main categories based on the STRIDE methodology, as discussed in Section 2.4.3. Figure 6 illustrates the classification of all identified threats according to the STRIDE model.

The chart explains that 20 threats were categorized as tampering, whilst 18 threats were classified as information disclosure. The spoofing category was considered to have the highest rate of all identified threats, with 25 threats classified in this category. The repudiation and elevation of privilege categories were considered to have the lowest rates, with only three and two threats classified into each category, respectively.

Based on the outcomes of THREATGET, we investigated which threats had a high impact on system components and related security attributes that were considered a set of existing security vulnerabilities. Table 3 represents a set of some selected threats identified by THREATGET and shows the affected system components and which of the related security attributes are defined as security vulnerabilities in the system, which pave multiple paths for an attacker to perform a set of malicious activities.

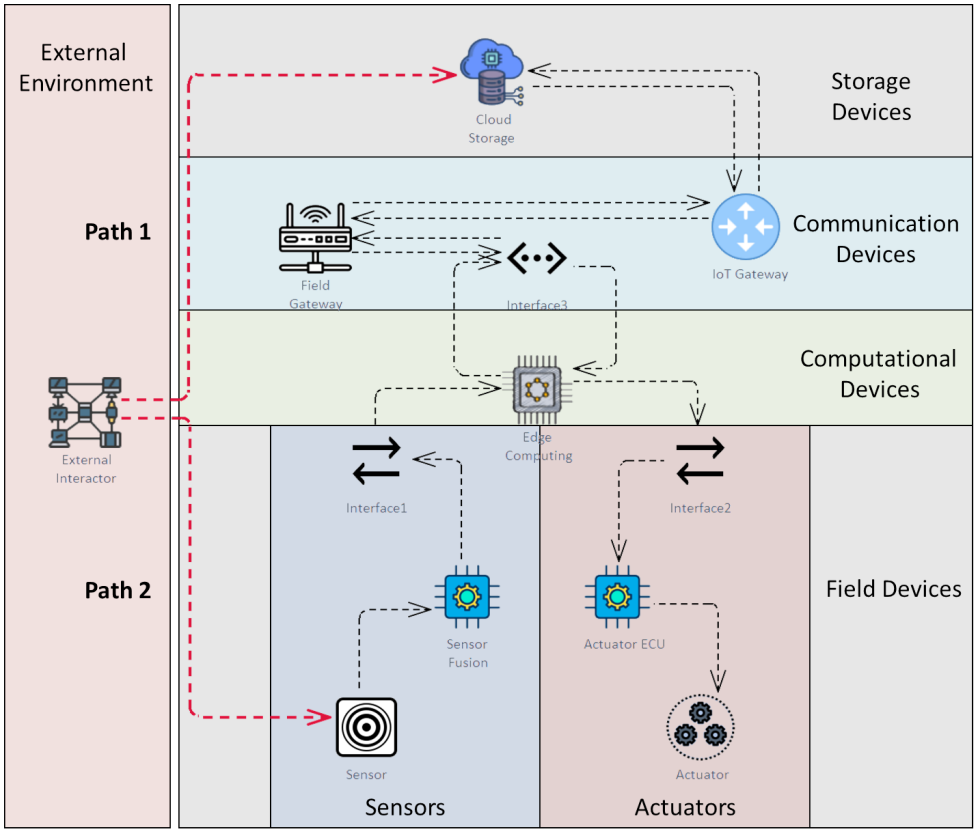


Figure 5. Modeling an agriculture example using ThreatGet.

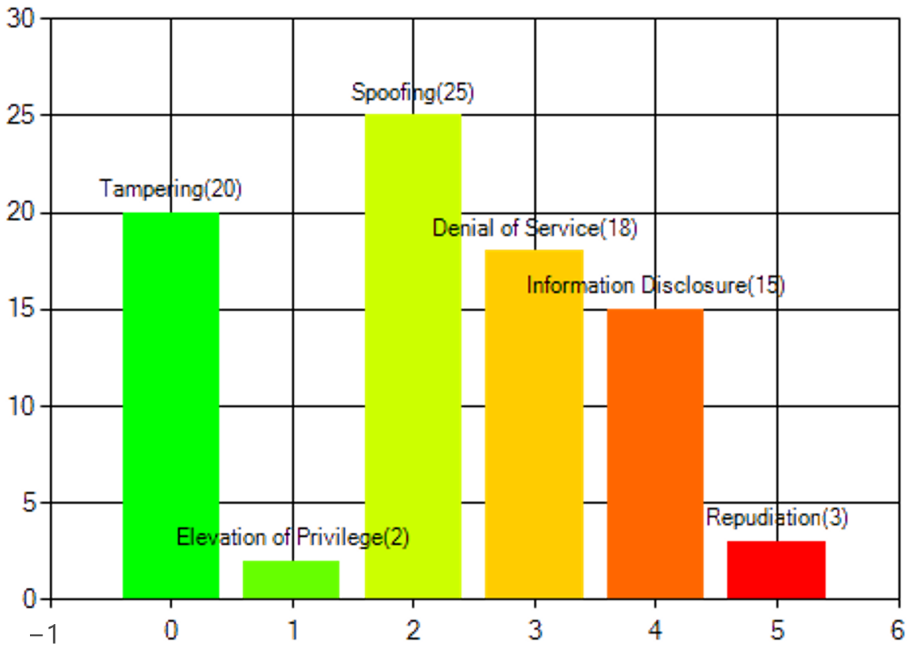


Figure 6. Classification of all identified threats according to the STRIDE model.

Table 3. Some selected potential threats identified by THREATGET.

Threat Title	Affected Components	Violated Security Attributes
Separation of execution environments	IoT Gateway	Input Validation
Update authenticity and integrity for embedded/network devices	Edge Computing	Tamper Protection, Input Validation
Information confidentiality	Interface1	Encrypted, Authorization
Network protection from malicious code	Field Gateway	Input Validation
Automated audit log access	Sensor Fusion	Anomaly Detection

The table shows a sample of the identified threats by THREATGET and gives details on the existing security vulnerabilities for each affected component. For example, the threat "Network protection from malicious code", according to IEC 62443 part 4-2 [13], network devices should be protected against incoming malicious data that could have a harmful impact on the network device and the other system components connected to it. As a result, THREATGET verifies the integrity of incoming data to the Field Gateway system component in order to ensure that the incoming data are legitimate. Consequently, the tool detects that the input validation security attribute is not activated for this component (in this case, Field Gateway) in order to mitigate a similar threat. It is defined in the table that the input validation security attribute is considered to be a violated security attribute in this case and should be addressed.

The full investigation of identified threats is described in Appendix A.

Based on our investigation into the outcomes of THREATGET, we define all needed security attributes for protecting the system model. Table 4, summarizes all affected elements and related security attributes.

Table 4. Summary of all affected system components and related violated security attributes.

Affected Component	Security Attributes
Sensor	Authentication, Secure Boot, Input Validation, Anomaly Detection
Interface1	Authorization, Encrypted, Authentication
Edge Computing	Authentication, Tamper Protection, Input Validation, Authorization, Updates, Secure boot, Encrypted, Anomaly Detection
Interface2	Authentication, Authorization
Actuator ECU	Authentication, Authorization, Secure Boot, Input Validation, Anomaly Detection
Actuator	Authentication
Interface3	Authorization, Authentication, Anomaly Detection
Field Gateway	Activity Logging, Anomaly Detection, Tamper Protection, Input Validation, Secure boot, Input Sanitization
IoT Gateway	Input Validation, Encrypted, Authentication, Activity Logging, Anomaly Detection, Input Sanitization Tamper Protection, Input Validation, Authorization, Secure Boot, Encrypted, Anomaly Detection,
Cloud Storage	Encrypted, Authorization

It is shown in the table which security attributes are required for each system component in order to mitigate cyber threats. We assume that a surface attack on the system model will be conducted through the previously discussed attack paths (i.e., **Path 1** and **Path 2**). This is illustrated in our case study in Figure 5. For this reason, based on Table A1 and the most prevalent security attributes for each component listed in Table 4, two attack path scenarios in which a surface attack will be propagated are anticipated to present. We

consider the edge computing component as the most essential unit and requires sophisticated security measures. Therefore, in the following scenarios, we propose that attack propagation shall proceed from an external unit until reaching the edge computing unit.

Surface Attack: Path 1

- The unauthorized access to the cloud storage component gives an attacker the way to inject malicious code/data into the system network.
- The connected system component (i.e., IoT Gateway in this case) should receive a set of compromised data/code, which lead to multiple negative consequences, such as changing the configuration of any critical system component.
- The authenticity between connected network devices shall be guaranteed, which is considered essential to confirm the authenticity with which the component is communicating.
- In addition, the input sanitization and anomaly detection shall be considered in this attack path in order to detect and prevent the flow of any compromised data/code into the rest of the connected system components.
- The data transmitted through the network to the edge computing unit shall be encrypted and transmitted from authorized service, human, or component.
- Additionally, the edge computing unit needs to check all incoming data from this path to ensure that it does not doubt the integrity of the received content.
- Furthermore, we consider that the system update for the edge computing unit is essential to keep this device up-to-date with all required security measures.

Surface Attack: Path 2

- Sensor components shall authenticate and authorize external interactors in order to ensure the authenticity of the incoming data.
- The sensor fusion plays an essential role in handling incoming data; therefore, input validation and secure booting are needed to mitigate any such similar cyberattack propagation.
- The data shall be encrypted; therefore, Interface1 shall be responsible for encrypting all of the data transmission.
- The edge computing unit may check all integrity of all received data from external environment.
- In addition, the edge computing unit shall be protected against any tampering activities.

Attack Propagation Towards System Actuator. Once the edge computing system has been compromised, we cannot anticipate any additional consequences for the other components of our system network connected to the edge computing system. Therefore, we believe that the system updates for the edge computing unit are essential to keep this device up to date with all required security measures.

- Interface2 needs to verify all connected components' authorization to ensure the validity of all incoming data.
- The lack of authorization could bypass other essential security attributes for the Actuator ECU, such as anomaly detection, secure boot, input validation, and authentication.
- Once an attack successfully reaches the system Actuator ECU, an attacker may gain full control of the component, leading to unexpected outcomes. For example, as described in Section 3.1, the system actuators control the irrigation process or the application of nutrients. Therefore, any unauthorized manipulation could lead to a financially or operationally negative impact.

3.2. Optimization of Security Attributes

In this section, we report optimization of all previously discussed attributes (i.e., Table 4). According to the proposed algorithm, all existing properties are optimized, and the algorithm is concerned with the presence or absence of any single attribute that could be a vulnerable point in the system's design. In order to maximize the overall attributes of a system component and maintain their efficacy against any attack propagation, this method examines each system's component. It determines which of its security properties

are covered by the components that came before it and which properties can be overlooked. Employing attack propagation scenarios based on **Path 1** and **Path 2**, we demonstrate the effectiveness of the finally optimized attributes. Table 5 illustrates all steps, followed by the algorithm to optimize all security properties defined in our agriculture system model.

Table 5. Optimized security attributes in our proposed agriculture system model.

Component	Protected	Incoming Covered	Covered	Final
Sensor	Authentication, Authorization		Authentication, Authorization	Authentication, Authorization
Sensor Fusion	Anomaly Detection, Authentication, Input validation, Secure boot	Authorization, Authentication	Anomaly Detection, Authentication, Authorization, Input validation, Secure boot	Anomaly Detection, Input validation, Secure boot
Interface1	Authentication, Authorization, Encrypted	Anomaly Detection, Authentication, Authorization, Input validation, Secure boot	Anomaly Detection, Authentication, Authorization, Encrypted, Input validation, Secure boot	Encrypted
Cloud Storage			Authorization, Encrypted	Authorization, Encrypted
IoT Gateway	Input Sanitization Activity Logging, Anomaly Detection, Authentication, Authorization, Encrypted, Input Validation, Secure boot, Tamper Protection	Authorization, Encrypted	Activity Logging, Anomaly Detection, Authentication, Authorization, Encrypted, Input Sanitization, Input Validation, Secure boot, Tamper Protection	Activity Logging, Anomaly Detection, Authentication, Encrypted, Input Validation, Secure boot, Tamper Protection
Field Gateway	Activity Logging, Anomaly Detection, Input Sanitization, Input Validation, Secure boot, Tamper Protection	Activity Logging, Anomaly Detection, Authentication, Authorization, Encrypted, Input Sanitization, Input Validation, Secure boot, Tamper Protection	Activity Logging, Anomaly Detection, Authentication, Authorization, Encrypted, Input Sanitization, Input Validation, Secure boot, Tamper Protection	

Table 5. *Cont.*

Component	Protected	Incoming Covered	Covered	Final
Interface3	Anomaly Detection, Authentication, Authorization	Activity Logging, Anomaly Detection, Authentication, Authorization, Encrypted, Input Sanitization, Input Validation, Secure boot, Tamper Protection	Activity Logging, Anomaly Detection, Authentication, Authorization, Encrypted, Input Sanitization, Input Validation, Secure boot, Tamper Protection	
Edge Computing	Anomaly Detection, Authentication, Authorization, Encrypted, Input Validation, Secure boot, Tamper Protection, Updates	Anomaly Detection, Authentication, Authorization, Encrypted, Input Validation, Secure boot	Anomaly Detection, Authentication, Authorization, Encrypted, Input Validation, Secure boot, Tamper Protection, Updates	Tamper Protection, Updates
Interface2	Authentication, Authorization	Anomaly Detection, Authentication, Authorization, Encrypted, Input Validation, Secure boot, Tamper Protection, Updates	Anomaly Detection, Authentication, Authorization, Encrypted, Input Validation, Secure boot, Tamper Protection, Updates	
Actuator ECU	Anomaly Detection, Authentication, Authorization, Input Validation, Secure boot	Anomaly Detection, Authentication, Authorization, Encrypted, Input Validation, Secure boot, Tamper Protection, Updates	Anomaly Detection, Authentication, Authorization, Encrypted, Input Validation, Secure boot, Tamper Protection, Updates	
Actuator	Authentication	Anomaly Detection, Authentication, Authorization, Encrypted, Input Validation, Secure boot, Tamper Protection, Updates	Anomaly Detection, Authentication, Authorization, Encrypted, Input Validation, Secure boot, Tamper Protection, Updates	

Table 5. *Cont.*

Component	Protected	Incoming Covered	Covered	Final
Actuator	Authentication	Authentication, Tamper Protection, Input Validation, Authorization, Updates, Secure boot, Encrypted, Anomaly Detection		

In particular, the algorithm's primary goal is to concentrate on which attributes of system components will be required in order to ensure the system's protection and to preserve the security mechanisms of all system components, which will allow them to resist any attack propagation from an untrusted external component. The table represents all of the stages that the algorithm goes through in order to define a wide variety of security properties for each component of the system. Each security attribute described as a set of protective techniques for protecting a specific component from malicious activity is represented by the first column (protected), which includes all existing security attributes. The second column (i.e., Incoming Covered) represents the intersection of covered incoming edges.The next column (i.e., Covered) includes all of the security attributes that were taken into consideration from the first component in the attack path all the way up to the current component. Consequently, the covered column is defined in order to determine which attributes have already been evaluated before the data flow reaches the current component. The final column (i.e., final) represents the final choice on which security attributes should be included in the current component based on all covered and protected attributes.

3.3. Evaluation of the Proposed Algorithm

A complete abstract model for the system components is described using B-method. Surface attack scenarios described in the previous subsection were verified on the optimized model to show that the security measures holds. The security properties are expressed inside the B-model through defining invariant predicates:

$$((state = authentication\_attack) => authentication\_attack\_signal = TRUE)$$

$$((unauthorized\_acess = TRUE) <=> (state = storage\_attack))$$

$$((authentication\_attack\_signal = TRUE) => (state = update\_sec\_property))$$

Atelier B extracted proof obligations from the described model (48 in the case of surface attack Path 1). Figure 7 shows the proof run on the environment machine that models surface attack Path 1. This path starts with an authorization failure in the cloud storage; this puts the storage in what we call storage attack state. As a result of the storage attack indicated by authorization failure, authentication check is verified in the IOT gateway through calling the authentication check operation in the IOT gateway machine. Some proof obligations generated by the Atelier B prover appears on the right side of Figure 7.

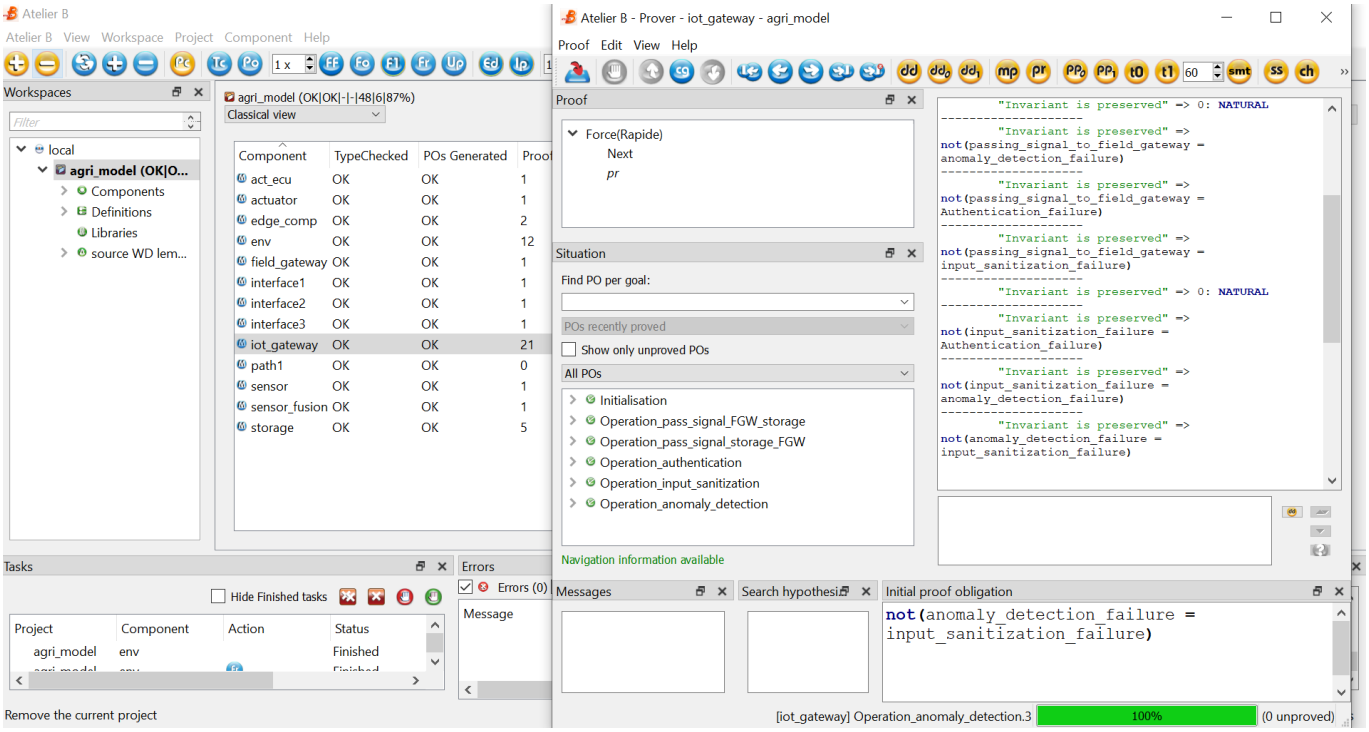


Figure 7. Proof obligations for the surface attack Path 1 scenario.

4. Discussion

In this work, we introduced a standard-conforming risk management approach for cyber threats. We defined different risk treatment options and discussed their applicability based on the likelihood and impact of occurring cyber threats. Moreover, THREATGET, a threat modeling tool developed at AIT, was utilized and enriched with security properties derived from the IEC 62443 series. An automated analysis of the system model based on IEC 62243-conforming anti-patterns revealed potential threats that may result in exploitation of the developed system. In contrast to covering all security measures for every component, an algorithm capable of identifying the minimum number of security attributes while keeping the system security at a maximum was introduced, thereby reducing the implementation effort required. Formal verification ensures the system's correctness and consistency against specifications. Formal verification is based on mathematical and logic proofs, not on simulating execution scenarios. Verifying security measures increases the system's robustness against security attacks. An abstract model of the use case helped to extract the proof obligations that must hold to ensure the system's consistency. Incorporating the minimum number of security attributes in the optimized system as invariants showed that security measures are preserved after applying the proposed optimization.

Author Contributions: Conceptualization, A.M.S. and S.C.; methodology, All; software, A.M.S. and S.C.; validation, N.E.-A.; formal analysis, N.E.-A.; investigation, A.M.S.; resources, A.M.S. and C.S.; writing—original draft preparation, A.M.S. and S.C.; writing—review and editing, All; visualization, A.M.S.; supervision, C.S. All authors have read and agreed to the published version of the manuscript.

Funding: Part of this work has been performed in the PRAETORIAN project, which has funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement 101021274.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors would like to express their sincere appreciation to Thorsten Tarrach from AIT—Austrian Institute of Technology—for his assistance in describing the mathematical formula for the proposed algorithm.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

FR	Foundational Requirements
DSL	Domain Specific-Language
IoT	Internet of Things
CPS	Cyber Physical Systems
DFD	Data-Flow Diagram
ECU	Electronic Control Unit
SuC	System Under Consideration
ZCR	Zones and Conduits Requirements
IAC	Identification and Authentication Control
UC	Use Control
SI	System Integrity
DC	Data Confidentiality
RDF	Restricted Data Flow
TRE	Timely Response to Events
RA	Resource Availability
IAC	Industrial Control System
IACS	Industrial Automation and Control Systems
SL	Security Level
CIA	Confidentiality, Integrity, and Availability
CPPS	Cyber-Physical Production Systems
COTS	Commercial-Off-The-shelf

Appendix A

This appendix gives more detail on the full investigation of THREATGET's outcomes. The following table discusses all potential threats identified by the THREATGET tool for determining cyber risks in our proposed case study, as discussed in Section 3.

Table A1. Potential threats identified by THREATGET.

Title	Affected Components	Violated Security Attributes
Separation of execution environments	IoT Gateway	Inpup Validation
Components secure by default	IoT Gateway	Encrypted
Violation of software/device identification and authentication	IoT Gateway	Authentication
Lack of security capabilities of IIoT network device	IoT Gateway	Activity Logging Anomaly Detection
Breach of control system identification and authentication	Edge Computing	Authentication
Update authenticity and integrity for embedded/network devices	Edge Computing Field Gateway IoT Gateway	Tamper Protection Input Validation

Table A1. *Cont.*

Title	Affected Components	Violated Security Attributes
Permission for information persistence	Sensor Edge Computing Actuator ECU Interface3 IoT Gateway Interface1 Interface2	Authorization
Embedded device updates authenticity and integrity	Interface1 Edge Computing Interface3	Authentication Input Validation Updates
Integrity of booting process for embedded devices	Sensor Fusion Actuator ECU Edge Computing	Secure boot Input Validation
Integrity of booting process for network devices	IoT Gateway Field Gateway	Secure boot Input Validation
Information confidentiality	Interface1 Edge Computing Cloud Storage IoT Gateway	Encrypted Authorization
Automated audit log access	Sensor Fusion Edge Computing Actuator ECU	Anomaly Detection
Network protection from malicious code	Field Gateway IoT Gateway	Input Validation
Machine-readable reporting of current security settings	IoT Gateway Field Gateway Interface3	Anomaly Detection Input Sanitization

References

1. Ma, Z.; Hudic, A.; Shaaban, A.; Plosz, S. Security viewpoint in a reference architecture model for cyber-physical production systems. In Proceedings of the 2017 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), Paris, France, 26–28 April 2017; pp. 153–159.
2. Shahzad, A.; Kim, Y.G.; Elgamoudi, A. Secure IoT Platform for Industrial Control Systems. In Proceedings of the 2017 IEEE International Conference on Platform Technology and Service (PlatCon), Busan, Korea, 13–15 February 2017; pp. 1–6.
3. Schmittner, C.; Tummeltshammer, P.; Hofbauer, D.; Shaaban, A.; Meidlinger, M.; Tauber, M.; Bonitz, A.; Hametner, R.; Brandstetter, M. Threat Modeling in the Railway Domain. In *Reliability, Safety, and Security of Railway Systems. Modelling, Analysis, Verification, and Certification*; Springer: Berlin, Germany, 2019; pp. 261–271.
4. Strobl, S.; Hofbauer, D.; Schmittner, C.; Maksuti, S.; Tauber, M.; Delsing, J. Connected cars—Threats, vulnerabilities and their impact. In Proceedings of the 2018 IEEE Industrial Cyber-Physical Systems (ICPS), Saint Petersburg, Russia, 15–18 May 2018; pp. 375–380. [CrossRef]
5. Kristen, E.; Kloibhofer, R.; Díaz, V.H.; Castillejo, P. Security Assessment of Agriculture IoT (AIoT) Applications. *Appl. Sci.* **2021**, *11*, 5841. [CrossRef]
6. Sobers, R. 134 Cybersecurity Statistics and Trends for 2021. 2021. Available online: https://www.varonis.com/blog/cybersecurity-statistics/ (accessed on 15 May 2022).
7. Abomhara, M.; Gerdes, M.; Køien, G.M. A STRIDE-based threat model for telehealth systems. *Norsk Informasjonssikkerhetskonferanse (NISK)* **2015**, *8*, 82–96.
8. *ISO/TC 22/SC 32*; ISO/SAE DIS 21434 Road Vehicles—Cybersecurity Engineering. ISO—International Standardization Organization: Geneva, Switzerland, 2020.
9. *ISO/TC 262*; Risk Management. ISO 31000—Risk Management—Guidelines. ISO—International Standardization Organization: Geneva, Switzerland, 2018.
10. ENISA. The Risk Management Process. 2020. Available online: https://www.enisa.europa.eu/topics/threat-risk-management/risk-management/current-risk/risk-management-inventory/rm-process/rm-process (accessed on 15 May 2022).
11. ENISA. Risk Treatment. 2020. Available online: https://www.enisa.europa.eu/topics/threat-risk-management/risk-management/current-risk/risk-management-inventory/rm-process/risk-treatment/risk-treatment (accessed on 15 May 2022).
12. *ISO/IEC 27005*, 2nd ed.; Information Technology—Security Techniques—Information Security Risk Management. ISO: Geneva, Switzerland, 2011.

13. *IEC 62443*; Security for Industrial Automation and Control Systems—Part 4-2: Technical Security Requirements for IACS Components. IEC International Standard: Geneva, Switzerland, 2019.
14. Shaaban, A. An Ontology-Based Cybersecurity Framework for the Automotive Domain: Design, Implementation, and Evaluation. Doctoral Thesis, University of Vienna, Vienna, Austria, 2021.
15. *ISA-62443*; The 62443 Series of Standards: Industrial Automation and Control Systems Security. ISA: Sydney, Australia, 2018.
16. Shaaban, A.M.; Jung, O.; Fas Millan, M.A. Toward Applying the IEC 62443 in the UAS for Secure Civil Applications. In *Data Science—Analytics and Applications*; Haber, P., Lampoltshammer, T.J., Leopold, H., Mayr, M., Eds.; Springer Fachmedien Wiesbaden: Wiesbaden, Germany, 2022; pp. 45–52.
17. *IEC 62443*; IEC—SyC Smart Energy. IEC International Standard: Geneva, Switzerland, 2020. Available online: https://syc-se.iec.ch/deliveries/cybersecurity-guidelines/security-standards-and-best-practices/iec-62443/ (accessed on 15 May 2022).
18. *IEC 62443-3-3*; Industrial Communication Networks—Network and System Security—Part 3-3: System Security Requirements and Security Levels. IEC International Standard: Geneva, Switzerland, 2013.
19. ISA-99/IEC 62443, S.L. ISA 99 Security Levels Proposal. 2020. Available online: https://www.scribd.com/document/129590220/ISA-99-SecurityLevels-Proposal/ (accessed on 25 May 2022).
20. Shaaban, A.M.; Kristen, E.; Schmittner, C. Application of iec 62443 for iot components. In Proceedings of the International Conference on Computer Safety, Reliability, and Security, York, UK, 7–10 September 2018; Springer: Berlin/Heidelberg, Germany, 2018; pp. 214–223.
21. *IEC 62443-3-2*; IEC 62443 Security for Industrial Automation and Control Systems—Part 3-2: Security Risk Assessment and System Design. IEC International Standard: Geneva, Switzerland, 2015.
22. Kloibhofer, R.; Kristen, E.; Jakšić, S. Safety and Security in a Smart Production Environment. In Proceedings of the International Conference on Computer Safety, Reliability, and Security, York, UK, 7–10 September 2018; Springer: Berlin/Heidelberg, Germany, 2018; pp. 190–201.
23. Torr, P. Demystifying the Threat-Modeling Process. *IEEE Secur. Priv. Mag.* **2005**, *3*, 66–70. [CrossRef]
24. Shevchenko, N. Threat Modeling: 12 Available Methods. 2018. Available online: https://insights.sei.cmu.edu/sei_blog/2018/12/threat-modeling-12-available-methods.html (accessed on 2 April 2022).
25. Karahasanovic, A.; Kleberger, P.; Almgren, M. Adapting threat modeling methods for the automotive industry. In Proceedings of the 15th ESCAR Conference, Berlin, Germany, 7–8 November 2017; pp. 1–10.
26. Shostack, A. *Threat Modeling: Designing for Security*; Wiley: Hoboken, NJ, USA, 2014.
27. Swiderski, F.; Snyder, W. *Threat Modeling*; Microsoft Press: Redmond, WA, USA, 2004.
28. Shostack, A. *Experiences Threat Modeling at Microsoft*; Microsoft Corporation: Redmond, WA, USA, 2008.
29. Joint Task Force Transformation Initiative. *Security and Privacy Controls for Federal Information Systems and Organizations*; NIST: Gaithersburg, MD, USA, 2013. [CrossRef]
30. Almohri, H.M.; Watson, L.T.; Yao, D.; Ou, X. Security Optimization of Dynamic Networks with Probabilistic Graph Modeling and Linear Programming. *IEEE Trans. Dependable Secur. Comput.* **2016**, *13*, 474–487. [CrossRef]
31. Martínez-Rodríguez, B.; Bilbao-Arechabala, S.; Jorge-Hernandez, F. Security Architecture for Swarms of Autonomous Vehicles in Smart Farming. *Appl. Sci.* **2021**, *11*, 4341. [CrossRef]
32. Better Security, Lower Cost. Available online: https://semiengineering.com/better-security-lower-cost/ (accessed on 15 May 2022).
33. Why You Need Security Optimization in 2021. Available online: https://www.descasio.io/why-you-need-security-optimization-in-2021/ (accessed on 17 May 2022).
34. Baritel-Ruet, C. Formal Security Proofs of Cryptographic Standards. Ph.D. Thesis, Université Côte d'Azur, Nice, France, 2020.
35. Andronick, J.; Chetali, B.; Paulin-Mohring, C. Formal Verification of Security Properties of Smart Card Embedded Source Code. In *International Symposium on Formal Methods*; Springer: Berlin/Heidelberg, Germany, 2005; Volume 3582, pp. 302–317. [CrossRef]
36. Niu, J.; Reith, M.; Winsborough, W.H. Formal Verification of Security Properties in Trust Management Policy. *J. Comput. Secur.* **2014**, *22*, 69–153. [CrossRef]
37. Sepulveda, J.; Aboul-Hassan, D.; Sigl, G.; Becker, B.; Sauer, M. Towards the formal verification of security properties of a Network-on-Chip router. In Proceedings of the 2018 IEEE 23rd European Test Symposium (ETS), Bremen, Germany, 28 May–1 June 2018; pp. 1–6. [CrossRef]
38. Avalle, M.; Pironti, A.; Sisto, R. Formal verification of security protocol implementations: A survey. *Form. Asp. Comput.* **2014**, *26*, 99–123. [CrossRef]
39. Demir, O.; Xiong, W.; Zaghloul, F.; Szefer, J. Survey of Approaches for Security Verification of Hardware/Software Systems. *IACR Cryptol. ePrint Arch.* **2016**, *2016*, 846.
40. Kulik, T.; Dongol, B.; Larsen, P.G.; Macedo, H.D.; Schneider, S.; Tran-Jørgensen, P.W.V.; Woodcock, J. A Survey of Practical Formal Methods for Security. *arxiv* **2021**, arXiv:2109.01362.
41. Wahba, A.M.; El-Araby, N.A. Formal verification of real time distributed systems using B method. *Int. J. Eng. Sci. Technol. (JEST)* **2011**, *3*, 3427–3436.
42. Thapa, V.; Song, E.; Kim, H. An approach to verifying security and timing properties in UML models. In Proceedings of the 2010 15th IEEE International Conference on Engineering of Complex Computer Systems, Oxford, UK, 22–26 March 2010; pp. 193–202.
43. Elaraby, N.; Kühn, E.; Messinger, A.; Radschek, S.T. Towards a Hybrid Verification Approach. In *Software Technologies: Applications and Foundations*; Mazzara, M., Ober, I., Salaün, G., Eds.; Springer International Publishing: Cham, Switzerland, 2018; pp. 367–386.

44. Abrial, J.R.; Hoare, A. *The B-Book: Assigning Programs to Meanings*; Cambridge University Press: Cambridge, UK, 1996; Volume 1.
45. El-Araby, N.A.; Wahba, A.M.; Taher, M.M. Implementation of formally verified real time distributed systems: Simplified flight control system. In Proceedings of the 2011 International Conference on Computer Engineering Systems, Nanjing, China, 24–25 September 2011; pp. 25–32. [CrossRef]
46. Dijkstra, E.W.; Dijkstra, E.W.; Dijkstra, E.W.; Dijkstra, E.W. *A Discipline of Programming*; Prentice-Hall: Englewood Cliffs, NJ, USA, 1976; Volume 613924118.
47. Behm, P.; Benoit, P.; Faivre, A.; Meynadier, J.M. Météor: A Successful Application of B in a Large Project. In *FM'99—Formal Methods*; Wing, J.M., Woodcock, J., Davies, J., Eds.; Springer: Berlin/Heidelberg, Germany, 1999; pp. 369–387.
48. Butler, M. A system-based approach to the formal development of embedded controllers for a railway. *Des. Autom. Embed. Syst.* **2002**, *6*, 355–366. [CrossRef]