ARC-C: Analytical Framework and Software Tool for Automated Risk-Based Cryptoperiod Calculation in Industrial Control Systems

Gabriele Cianfarani

A Thesis submitted to
the Faculty of Graduate Studies
in Partial Fulfilment of the Requirements
for the Degree of
Master of Electrical and Computer Engineering

Electrical Engineering and Computer Science (EECS)
York University
Toronto, Ontario
February 2025

Abstract

Over the past decade, industrial control systems (ICSs) and critical infrastructure (CI) have become prime targets for advanced persistent threat (APT) groups and nation-state actors due to their potential for severe impact. This has resulted in the cybersecurity community increasing their focus on ICS/CI threat modelling and defence.

This thesis examines the crucial role of the internal network reconnaissance stage of ICS/CI attacks, particularly those using the OPC UA standard with encrypted in-transit data. We first introduce a comprehensive attack tree outlining data siphoning strategies and highlight the importance of periodic encryption-key rotation to mitigate risk. Noting the lack of clear cryptoperiod guidelines in industry standards, we then present the Automatic Risk-based Cryptoperiod Calculation (ARC-C) framework. ARC-C aims to optimally determine cryptoperiod lengths based on security risks and operational constraints. We demonstrate its application in two realistic ICS environments: a Water Treatment Plant and an Energy Storage System.

Acknowledgements

Working on this thesis has been quite an exciting journey, one that I have not travelled alone. I'd like to take a moment to thank everyone that helped me reach my destination.

Every step of the way, my supervisor, Professor Vlajic, has been there to ensure that I did not fall off course. From guidance to feedback, she assisted me in refining my ideas, sharpening their focus, and ensuring my arguments held firm. She also helped remould my writing habits to flow more logically and align with the rigor expected of scholarly work. The lessons I have learned go beyond what was done in this thesis and will guide me in my professional career for years to come.

I would also like to thank Thomas Nehring, Robert Noce, and Edgar Wolf, for providing wisdom and insight at key moments in this journey. Their expertise in the field of ICS was invaluable, offering both perspective and feedback which greatly enhanced the depth and quality of my work.

Lastly, I would like to express my gratitude to Professor Litoiu as both a member of my supervisory and examination committee, as well as to Professor Khan joining me at the final stages by serving on my examination committee.

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Cybersecurity and Industrial Control Systems

Cybersecurity has been a concern for users, companies, and governments since the earliest days of computers and communication networks (i.e., the Internet). Nowadays, with the ever-increasing adoption of digital technologies across all industrial and societal sectors, the significance of cybersecurity has been further intensified. Moreover, modern-day cyber threats are no longer limited to simple breaches, but now encompass a wide range of malicious activities - ranging from stealing of sensitive corporate data to serious disruption of industrial and critical infrastructure systems.

In the most general terms, Industrial Control Systems (ICS) are computer/network systems used to control and mange industrial processes. ICSs are integral components of many modern industrial operations where automation and precise control of physical processes are required. For instance, automotive assembly lines use ICSs to automate machinery, ensuring precision and efficiency in the production process. In water treatment plants, ICSs regulate purification, chemical dosing, filtration, and disinfection processes, maintaining the safety and quality of water supplies.

From the operational perspective, ICSs support a range of functions which are designed to operate in a real-time manner and facilitate monitoring and controlling of different industrial processes. ICS environments typically contain numerous interconnected components, and ideally (would) place a priority on secure and efficient communication among these components to prevent disruptions and ensure smooth operations. Unlike traditional IT environments, where confidentiality, integrity, and availability (CIA) of data are of primary concerns, ICS environments - which encompass both IT and Operational Technology (OT) segments - additionally need to focus on maintaining the reliability and safety of industrial processes [1]. Unfortunately, many real-world ICSs still contain outdated legacy components that have been in place for decades, which pose unique challenges in terms of their integration with modern technologies and systems as well as impede the deployment of adequate cybersecurity measures/defences.

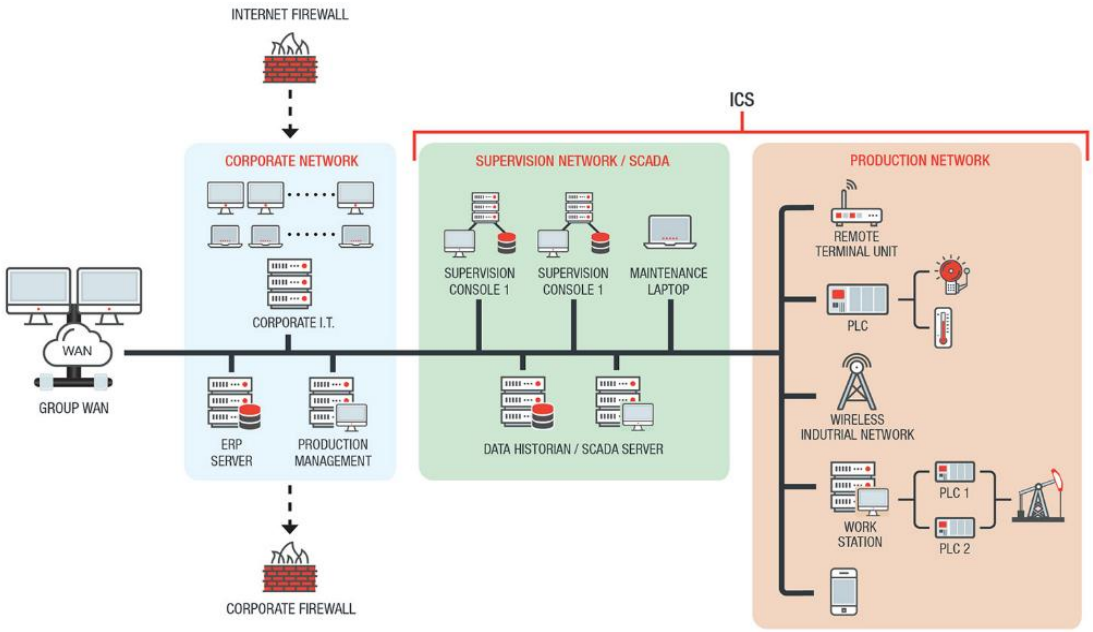


Figure 1: Integrated Network Architecture of Enterprise and ICS [2]

1.2 Data Siphoning/Data Exfiltration in ICSs

Thanks to the advances in the field of cyber threat modelling and intelligence (including the MITRE ATT&CK initiative [3]), we now know that most cyber attacks begin with the so-called stage of 'network and system reconnaissance'. During this stage, the adversary seeks to gather critical information about the target system, including: 1) number, type, status, and role of deployed system components, 2) number and type of deployed network appliances and communication protocols, and 3) potential presence of vulnerabilities in different system and network components. By getting hold of this information, the adversary becomes better equipped to plan and execute subsequent highly targeted and greatly impactful stages of the attack. In the case of a target ICS, this may include compromise of cyber-physical components and/or operational sabotage.

For an adversary that has already obtained a foothold (i.e., covert presence) in the target system, one of the most common ways to get hold of most of the above enlisted information is through gathering and exfiltration[1] of important operational in-transit data exchanged between

[1] As explained in [6], data exfiltration involves an unauthorized party intercepting and extracting data from a system, network, or device [7], [8].

different system components, as illustrated in Figure 2 [4], [5]. It should be noted, however, that for security and/or compliance purposes many modern-day ICSs resort to encryption of their in-transit data. Consequently, successful data exfiltration does not grant the adversary immediate access to the underlying (human-understandable) information contained in the captured/exfiltrated data.

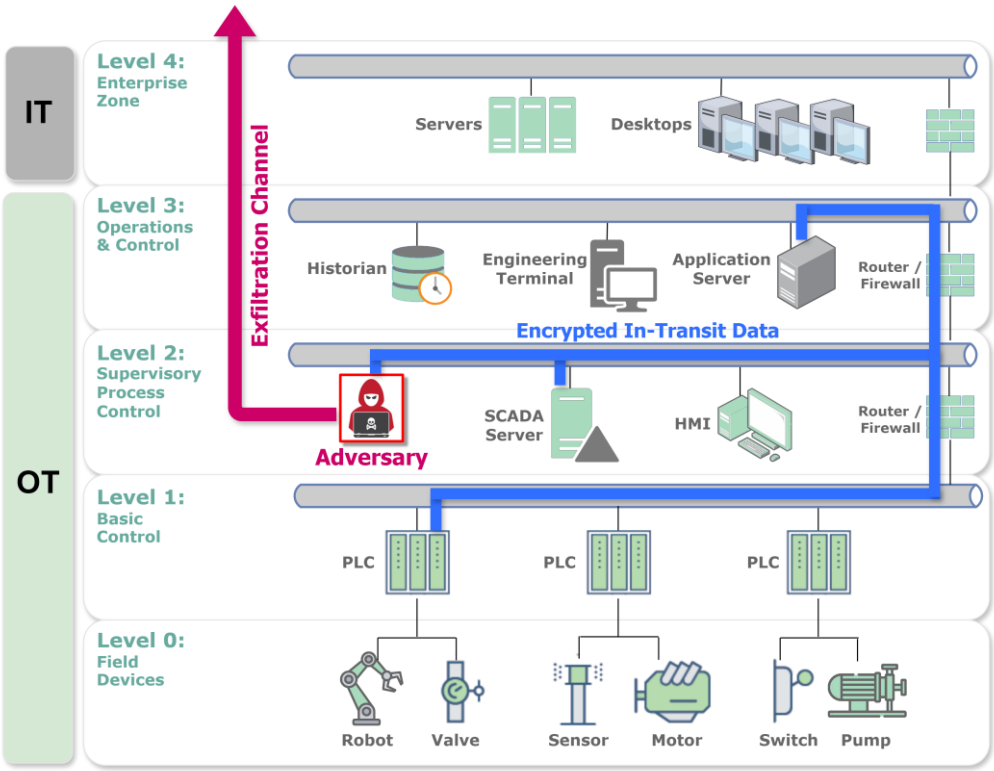


Figure 2: Data exfiltration by an insider adversary in an ICS

In this thesis, we introduce the term Data Siphoning (DS) to refer to the entire sequence of steps and efforts that an adversary needs to perform to access/disclose certain sensitive information of interest (e.g., critical operational data, system configurations, and any other data that could aid an adversary in compromising ICS operations). In cases when the sensitive information of interest is contained within in-transit data exchanged between two or more components/devices of an ICS, and the adversary has already established their presence in the target system, the specific steps of a DS attack include: 1) collection and exfiltration of respective in-transit data (STEP 1 in Figure 3), and 2) additional efforts to acquire the decryption key when the data from STEP 1 happens to be encrypted (Step 2 in Figure 3).

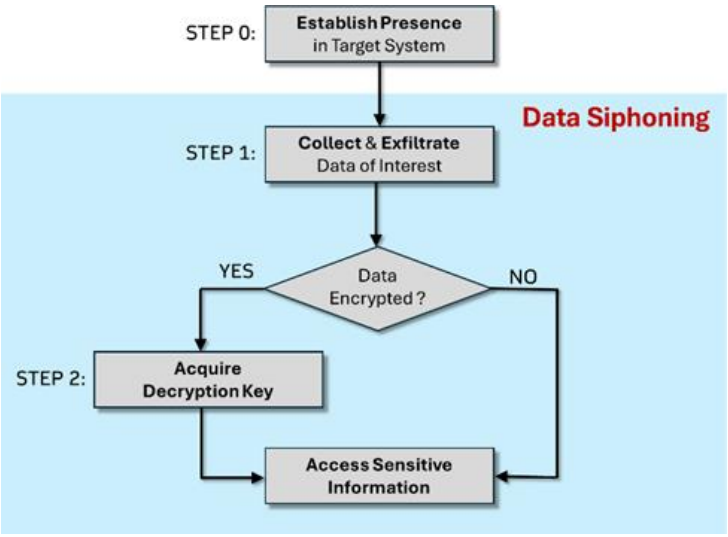


Figure 3: Stages required for a successful DS attack

One of the best known real-world DS attacks on an ICS was the New York Dam intrusion from 2013, in which a nation-state adversary was able to penetrate into the dam's SCADA system and exfiltrate critical data pertaining to the status and operation of the dam (such as water levels and temperature, the status of the sluice gate for controlling water levels, and flow rates [6]). With access to this information, the adversary was well positioned to conduct various subversive and potentially detrimental actions. Fortunately, these actions did not take place only thanks to the fact that parts of the dam's control system had been manually disconnected for maintenance at the time [6]. More recently (in early 2024), and according to [7], another nation-state actor succeeded in compromising several US-based water and electric utility companies, defence-related industrial bases, as well as satellite systems, with the main objective to establish long-term persistence as well capabilities for exfiltration of data pertaining to operational processes, configuration of OT assets, SCADA system configurations, etc.

Unfortunately, despite its critical role in planning and execution of attacks on ICSs, and its increasing real-world prevalence, DS remains a largely overlooked problem in cybersecurity research literature. The work of this thesis aims to fill that gap by building a detailed model of attack that involves data exfiltration and data decryption. At the same time, there are industry standards that do recognize the practical significance and potential dangers of data exfiltration/siphoning but fall short of providing explicit guidelines on how to safeguard ICS environments against their successful execution. In the following section, we provide a brief mention of these standards, which will again be revisited in greater detail within later chapters of this thesis.

1.3 Security Standards, Data Siphoning, and Cryptoperiods

The National Institute of Standards and Technology (NIST) provides a range of guidelines to help organizations secure their systems. Among these, NIST Special Publication (SP) 800-82 [1] and NIST SP 800-57 [8] are particularly relevant for enhancing the security of Industrial Control Systems and dealing with various attacks on ICSs. NIST SP 800-82 standard, titled "Guide to Operational Technology (OT) Security," provides specific guidelines tailored to the unique requirements of ICS environments. NIST SP 800-57 standard, titled "Recommendation for Key Management," addresses the management of cryptographic keys and respective cryptoperiods to minimize data exposure in case of exfiltration/siphoning attacks.

In NIST SP 800-57, a cryptoperiod is defined as “The time span during which a specific key is authorized for use or in which the keys for a given system or application may remain in effect” [8]. A more detailed description of the context in which a cryptoperiod is employed is also provided:

> “A cryptoperiod for a key is assigned for a number of reasons, including limiting the amount of exposure of encrypted data if a key is compromised. Cryptoperiods are usually assigned for a carefully considered period of time or by a maximum amount of data to be protected by the key. Tradeoffs associated with the determination of a cryptoperiod involve the risks and consequences of exposure.” [8]

From the excerpt, it is obvious that the recommendation to limit the amount of data encrypted with a single key is not motivated by a possibility of cryptanalytic attacks, but rather by how much data would be exposed in the event the encryption key actually gets compromised. While NIST SP 800-57 offers extensive general recommendations on key management practices, it does not provide a specific guideline or formula for determining the optimal cryptoperiod in each organization, leaving them to interpret and apply these guidelines on their own. This lack of an explicit cryptoperiod management framework that is based on the actual threats and risks of an organization, particularly within ICS environments, leads us to our main problem statement and what our thesis aims to address; **Industrial Control Systems lack a clear framework for determining optimal cryptoperiods, increasing the risk of key compromise and data exposure**.

1.4 Summary of upcoming sections

The reminder of this document is structured as follows:

In Chapter 2, we review several existing security-related frameworks that are highly relevant to our work and ultimately facilitate the development of our framework/tool for calculation of optimal cryptoperiods. This includes the National Vulnerability Database (NVD) and the MITRE ATT&CK framework.

In Chapter 3, we provide an overview of the OPC UA standard and its use in industrial automation. We also present different OPCA UA architecture models suited for ICS environments, and we highlight the main features and functionalities of the PubSub model and its so-called security groups (which the reminder of this thesis focuses on).

In Chapter 4, we introduce the concept of DS and explain its critical role in launching and execution of more complex attacks on ICSs and propose our comprehensive DS attack tree model, which maps out several alternative strategies the adversary could pursue while seeking to acquire the data's decryption key. Within the context of an OPC UA PubSub environment, we also explain the general role of cryptoperiods. Furthermore, we explain the risks and impacts of poorly adjusted cryptoperiods, especially in a system under a DS attack that cannot be detected and/or adequately prevented.

In Chapter 5, we derive an explicit formula for calculation of the aggregate risk of key compromise across all members of an OPC UA security group. Using this formula, we introduce our novel risk-based framework for optimal and adaptive adjustment of a security group's cryptoperiod that aims to minimize the overall impact of a DS attack (that could not be otherwise prevented) on the given group and the respective system.

In Chapter 6, we present our Automated Risk-based Cryptoperiod Calculator (ARC-C) software tool, which implements the framework described in Chapter 5 and is intended for use in real-world ICSs. This includes an overview of the tool's UI and certain design decisions made for a seamless experience of the ICS operator using the tool to acquire optimal cryptoperiods in their environment.

In Chapter 7, we introduce two experimental test cases involving two different industrial facilities modelled after similar real-world ICS environments - a water treatment facility and an energy storage system. We also present some of the experimental results obtained by applying

the ARC-C tool to these environments, which clearly demonstrate the tool's practical usefulness and versatility.

To enhance the readability of this thesis, the purpose/content of each chapter is depicted in Table 1 - where the chapters in the first column (Existing Research) serve to introduce some foundational knowledge, while the chapters in the second column (Original Contributions) present our novel work and contributions.

<table>
<tr><th>Existing Research</th><th>Original Contributions</th></tr>
<tr><td>Chapter 2</td><td></td></tr>
<tr><td>Chapter 3</td><td></td></tr>
<tr><td colspan="2">Chapter 4</td></tr>
<tr><td></td><td>Chapter 5</td></tr>
<tr><td></td><td>Chapter 6</td></tr>
<tr><td></td><td>Chapter 7</td></tr>
</table>

Table 1: Organization of thesis content by existing research and original contributions

2 Relevant Security Frameworks / Databases

There are several well-known security frameworks/databases which can be utilised to provide a structured foundation for the development of effective cyber defences. Two such frameworks - MITRE ATT&CK and NVD CVSS/CVE - are incorporated into the work of this thesis by: a) serving as a backbone for the logic of our ARC-C analytical framework, and b) by being integrated in the actual ARC-C software tool. In particular, the MITRE ATT&CK framework provides an extensive and well-structured catalogue of information pertaining to tactics, techniques, and impacts of real-world attacks, and as such has allowed us to develop a realistic DS threat model as well as evaluate its various potential impacts on real-world ICSs. On the other hand, the NVD CVSS/CVE framework offers a structured approach to reporting and evaluation of software and hardware vulnerabilities, and ultimately enables us to calculate the risk of encryption key compromise during a DS attack. In the reminder of this chapter, we provide a more detailed overview of each of these frameworks.

2.1 NVD Framework/Database

The National Vulnerability Database (NVD), maintained by the National Institute of Standards and Technology (NIST), acts as a central repository of information about known vulnerabilities in software and hardware products [9]. The database provides insights that can help the cyber security community understand and address potential security risks of various software/hardware vulnerabilities. The actual information contained in the database is based on a standardised method for scoring of different threat aspects for each reported vulnerability.

2.1.1 Common Vulnerabilities and Exposure (CVE)

At the heart of the NVD Database are Common Vulnerabilities and Exposures (CVEs), which are unique identifiers assigned to each of the known/recorded vulnerabilities (i.e., vulnerabilities registered in the database). CVEs allow for a more streamlined naming and tracking of vulnerabilities, as well as more effective communication and collaboration between security professionals and researchers worldwide. The vulnerabilities contained in the NVD Database are submitted by various members of the cybersecurity community and get vetted by the CVE Assignment Team and the CVE Numbering Authorities (CNA), as illustrated in the below figure.

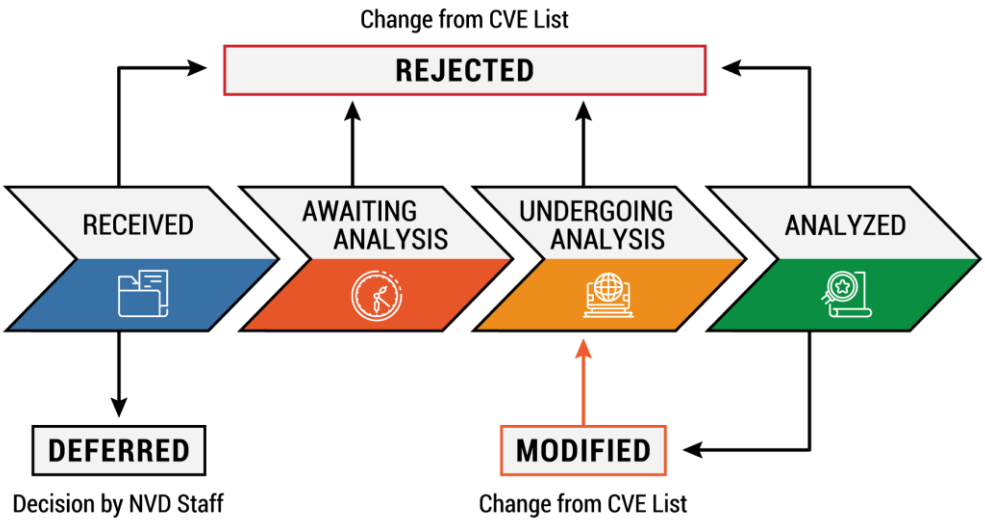


Figure 4: Process of adding and updating CVEs to the CVE List [10]

CVEs are stored and displayed in a structured manner, providing detailed information about each vulnerability. Each CVE entry includes an ID, which is a globally unique identifier, along with a description of the vulnerability, affected software or hardware, impact assessment, and more. The NVD also assigns a severity score to each CVE using the Common Vulnerability Scoring System (CVSS), which is covered in the upcoming subsection.

The NVD also provides various ways to access CVE information, including through its API. The API allows developers and security teams to programmatically retrieve CVE data, granting access to vulnerability information for integration into security tools and vulnerability management systems. The NVD API also provides flexible query capabilities which allows users to search for CVEs based on specific criteria such as a product or a severity. Our use of the API will be discussed in further detail in Chapter 6.

2.1.2 Common Vulnerability Scoring System

To help organisations assess the severity of a known/reported vulnerability, the NVD employs a methodology known as the Common Vulnerability Scoring System (CVSS). CVSS provides a framework for the quantitative and qualitative rating of a vulnerability's severity, considering factors such as the vulnerability's exploitability and its impact on the affected system. Recently, CVSS 4.0 was introduced which aims to improve scoring accuracy and address gaps identified in earlier versions. However, our focus will be on CVSS 3.0, as it is currently more widely supported and extensively deployed by the industry. Additionally, many existing tools and datasets

are still aligned with CVSS 3.0, making it more practical for our current analysis and implementation.

CVSS 3.0 contains three metrics: *Base*, *Temporal* and *Environmental*. NVD uses these scores to assist organisations in making informed decisions about the urgency and priority of addressing a specific vulnerability within their systems.

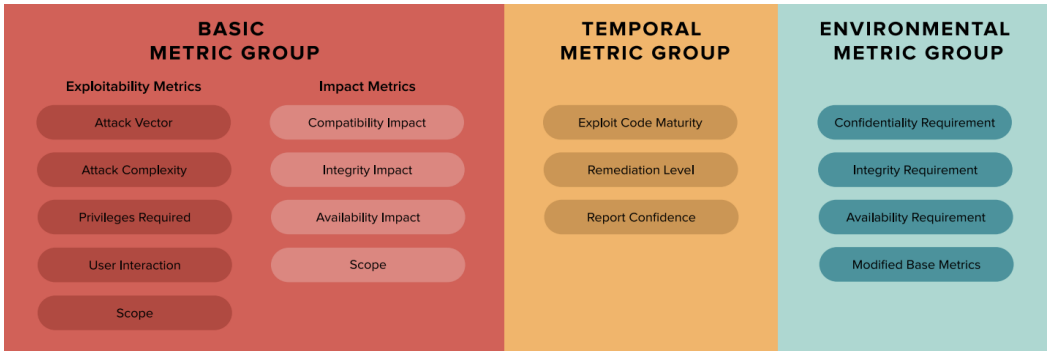


Figure 5: Metric groups for CVSS 3.0 [11]

The vulnerability scores in the *Base Metric Group* are static values, which do not change with time or depend on the environment in which the respective software/hardware component is deployed. These score values are displayed in the NVD when searching CVEs and, as the name suggests, they serve as a base for calculation of the other two metric groups. The scores in the *Temporal Metric Group* describe the current ability for malicious actors to exploit a given vulnerability based on available techniques. In other words, the scores in this group can change over the lifetime of a vulnerability. For example, if a new exploit code for the given vulnerability becomes publicly available, scores pertaining to some metrics in this group would increase. *Environmental metrics* account for the conditions of the system in which the device containing the CVE is deployed. For example, if the CVE exists in a critical infrastructure, the score for this metric would greatly increase.

CVSS Version 3.0 Rating	
Low	0.1 - 3.9
Medium	4.0 - 6.9
High	7.0 - 8.9
Critical	9.0 - 10.0

Figure 6: CVSS 3.0 quantitative to qualitative mapping [9]

Once numerical (i.e., quantitative) scores are calculated, NVD also derives/provides respective qualitative severity ratings, as shown in Figure 6. The purpose of the qualitative ratings is to facilitate the use of CVSS scores in qualitative cyber risk frameworks and tools. For more information on the CVSS metric groups, visit Appendix A.

2.2 MITRE

The MITRE Corporation is an American not-for-profit organisation [3] which plays a pivotal role in the field of cybersecurity, providing resources to assist businesses in assessing, understanding, and mitigating their cyber risks.

In 2013, MITRE introduced a groundbreaking threat modelling framework called ATT&CK (Adversarial Tactics, Techniques, and Common Knowledge), which provides valuable insights into the tactics, techniques, and procedures (TTPs) used by real-world cyber adversaries during different stages of an attack. The ATT&CK framework is structured into matrices that correspond to different platforms and environments, one of which being ICS. Each matrix includes a set of main adversarial tactics and corresponding techniques deployed in the respective environment, as illustrated in Figure 7.

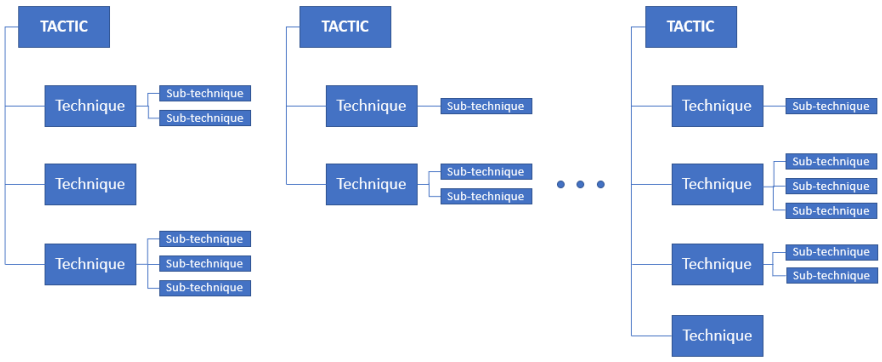


Figure 7: General structure of a MITRE ATT&CK matrix [12]

2.2.1 ICS Impact Tactic

The last tactic in the MITRE ATT&CK ICS (Industrial Control Systems) matrix is Impact, and it specifies the adversary's ultimate goal(s) when conducting an attack [3]. It is designed specifically to assess the potential damages inflicted on an ICS as a result of a security incident/attack. These consequences range from instantaneous operational disruptions (such as immediate loss of control) to long term harm (such as covert surveillance).

The ICS impact categories themselves can be broken down into three main groups: loss, denial, and manipulation, with each depicting different ways adversaries can affect and disrupt an industrial system.

2.2.2 Adversaries

Along with the ATT&CK matrices, MITRE also offers a suite of tools and resources to enhance organisations' understanding of tactics and techniques deployed by real-world adversaries, and then enable them to choose the best suited and most effective defences. In the MITRE ATT&CK framework, adversaries are classified into the so-called Advanced Persistent Threat (APT) groups, which are collectively tracked and identified by various entities and organizations of the global cybersecurity community. Due to the decentralised nature of threat tracking, APT groups may be listed by several names, as various organisations create their own designations. This also means that some overlap may exist where organisations may disagree on a specific activity. To combat potential conflicts, MITRE also includes a separate “Associated Group” tab to identify such overlaps [13]. However, it is important to understand that these overlaps are approximations and not exact matches.

The information provided by MITRE on adversarial groups goes beyond simply categorising them. It also includes detailed mappings of the specific techniques utilised by each group, shedding light on their modus operandi and the tactics employed during their cyber operations. In addition, MITRE's website includes references to original sources, enabling deeper dives into the details of specific attacks, threat reports, or intelligence sources.

3 Overview of OPC UA

Open Platform Communications Unified Architecture (OPC UA) [14] is one of the leading communication standards (i.e., protocol) for ICS and IIoT environments in Industry 4.0. It is developed by the OPC Foundation and has gained popularity due to its unique ability to ensure seamless interoperability and data exchange among devices produced by different vendors, different operating systems and/or via different communication protocols. In terms of its real-world deployment, OPC UA has been integrated into many ICS and IIoT systems across the globe and almost all known types of PLC firmware.

3.1 OPC Classic

OPC UA is not the first iteration of the OPC framework. Being introduced in the 1990s, OPC Classic (previously known as OPC) laid the foundation for industrial communication in the early stages of industrial automation. While revolutionary for its time, it had certain limitations that OPC UA aimed to address and overcome.

One of the primary limitations of OPC Classic was its dependence on Microsoft's COM/DCOM (Component Object Model/Distributed Component Object Model) technology [15]. This made the protocol tethered to the Windows operating system, limiting its ability to be used across different platforms (Figure 8). As industries diversified their hardware and software ecosystems, cross-platform compatibility became a crucial consideration.

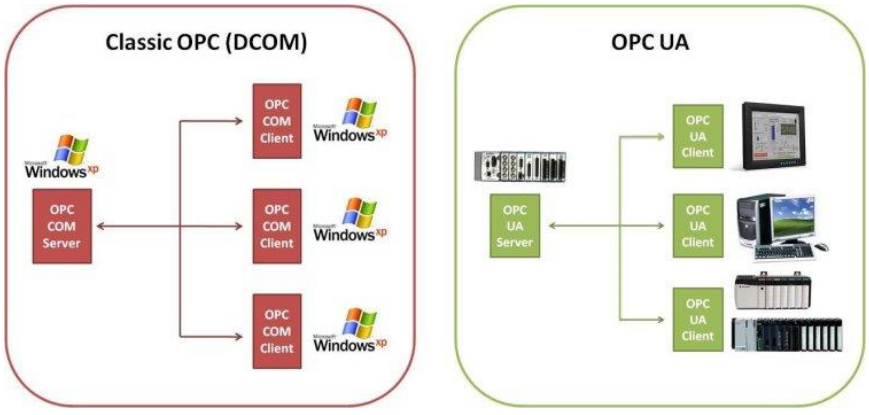


Figure 8: OPC Classic vs OPC UA structure [16]

Security was another area where OPC Classic faced constraints. The security features in OPC Classic were rudimentary, lacking crucial built-in mechanisms like encryption and authentication, relying on the security provided by the Windows OS [17]. This limitation made

OPC Classic less suitable for evolving industrial environments, as its restricted security capabilities increased the risk of issues like unauthorized access and compromised data integrity. Recognizing these limitations, the OPC Foundation began working to develop a protocol which addressed these shortcomings.

3.2 OPC UA

Released in 2006, OPC UA was conceived as a more adaptable, platform-independent, and secure successor to OPC Classic, designed to cater to the more modern landscape of industrial automation. It has started to become the preferred choice for new installations and modernization efforts in the industrial automation sector, claiming over 50% of the global OPC server software market [18].

Among the broader ICS communication protocol market, OPC has been the standard of choice for many big-name companies and foundations. In 2016, Microsoft announced its partnership with the OPC foundation, stating their focus on enabling its users to establish connections with a wide range of manufacturing equipment and software [19]. VDMA, a German company which represents more than 3200 companies in the capital goods industry, also announced a collaboration with OPC in 2017 [20]. This partnership put an emphasis on creating a standard for Industry 4.0, leveraging the OPC UA infrastructure. With major players as well as smaller enterprises engaging in partnerships with OPC to use OPC UA, it highlights its recognition as a foundational standard for industrial environments.

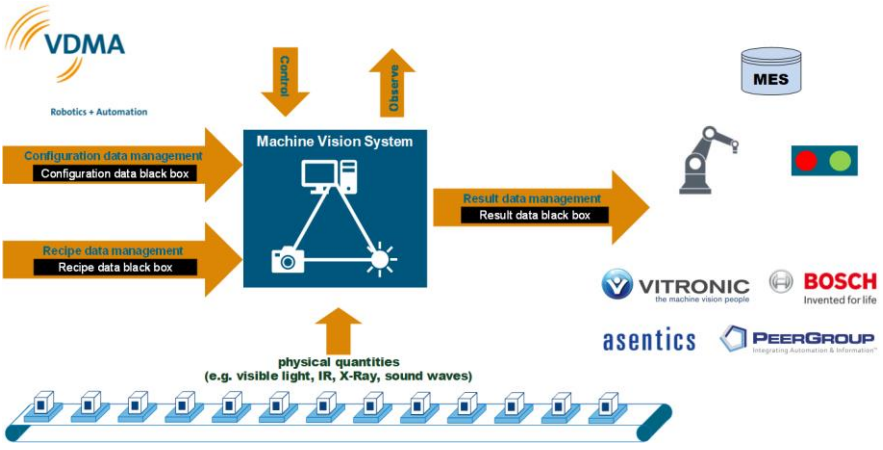


Figure 9: Official VDMA OPC machine vision demonstration [21]

3.2.1 Communication Models

OPC UA offers a robust and secure communication framework that supports both client-server and publish-subscribe (PubSub) communication models [22]. Our research focuses on the

PubSub architecture, as it provides a scalable and efficient method for distributing real-time data across numerous devices and applications — a critical feature in many ICS environments.

There are three primary roles in this model: the publisher, the subscriber, and the broker (see Figure 10). Publishers are responsible for broadcasting data, events, or notifications to the network and can be any device generating data in the system (e.g., sensors). On the other hand, subscribers are entities that express their interest in specific data or events (done by "subscribing" to specific topics) and receive relevant information from publishers. Subscribers can be various applications, systems, or devices that require access to real-time data updates.

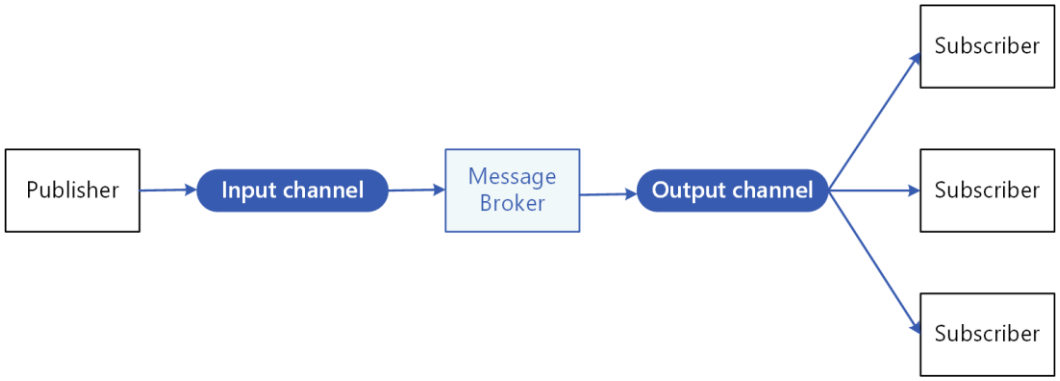


Figure 10: Publisher and Subscriber architecture in OPC UA [23]

The broker serves several essential functions: 1) it receives data from publishers and ensures that it is formatted appropriately for distribution, 2) it maintains a directory of available data and events, enabling subscribers to discover and express interest in particular information via subscribing to topics, and 3) when new data is available on a topic that a subscriber has subscribed to, the broker routes (i.e., pushes) this information to them while ensuring that each subscriber receives only the data they request.

The inclusion of the broker in this model allows for a more organised and efficient data exchange process, reducing the communication burden on publishers (i.e., they send only one message to the broker vs. sending *n* messages to each of the subscribers) and ensuring that data is delivered reliably to subscribers. Due to this, PubSub models are particularly well-suited for applications in which publishers are resource-constrained devices.

3.2.2 Security Groups

OPC UA also contains a suite of functionalities and features related to the so-called security groups. From the OPC UA documentation, a security group is an "abstraction that

represents the security settings and security keys that can be used to access messages from one or a group of Publishers" [24]. The purpose of these groups is to ensure that information is broadcasted only to authorised recipients by enforcing a base level of authentication and confidentiality protection. Namely, to access messages sent to a specific security group, an encryption key is required. In a typical PubSub setting, the publisher will encrypt a message with the encryption key that has been specifically assigned to all members of its respective group, and this key will be used again by the subscriber to verify the message signature and decrypt its content. These keys, along with a list of security group IDs (unique identifiers assigned each to security group), are managed by a system called the Security Key Service (SKS) server.

The Security Key Service (SKS) server acts as the central control for OPC UA keys (including functions such as key cycling and replacement), managing them for all publishers and subscribers. Another important role of the SKS server is mapping security groups (i.e., group members) and their associated roles, as illustrated in Figure 11.

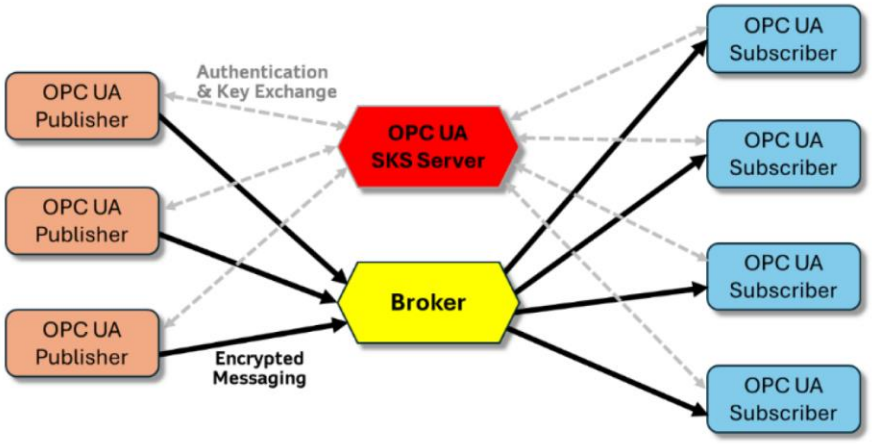


Figure 11: PubSub model with an SKS

In the OPC UA protocol documentation, there are two vital methods implemented via the SKS server: GetSecurityKeys and SetSecurityKeys [25] [26].

GetSecurityKeys (see Figure 12) is used to access and retrieve the keys by each security group member. A security group key becomes invalid after either being revoked and replaced by another key, or if the security group itself is removed. When calling the GetSecurityKeys method, three input values are required:

- **SecurityGroupId:** Security group identifier.

- **StartingTokenId:** Where to begin the key request. In the SKS, keys are referenced using tokens. A token of 0 will point to the current key, tokens less than 0 will point to past tokens, and tokens greater than 0 will point to future tokens.
- **RequestedKeyCount:** The number of requested keys (which need to be returned to the requester) where 0 implies only the current key. A large number may be limited to the maximum number of keys permitted by the SKS.

While GetSecurityKeys allows a security group publishers and/or subscribers to request security keys, the SetSecurityKeys (Figure 13) method is used by the SKS server to distribute the keys back to these (requesting) publishers or subscribers. In addition to providing the actual keys, this method is also used to set some additional important information about the keys, including TimeToNextKey and KeyLifetime.

- **KeyLifetime:** Each key is given a time interval (i.e., lifetime) measured in milliseconds during which it can be used. This interval is also commonly referred to as a cryptoperiod and will be discussed further in Section 4.2. Once the key lifetime has elapsed, the key is considered "expired" and can no longer be used.
- **TimeToNextKey:** Represents the specific time interval measured in milliseconds after which the current key should be replaced with the next key.

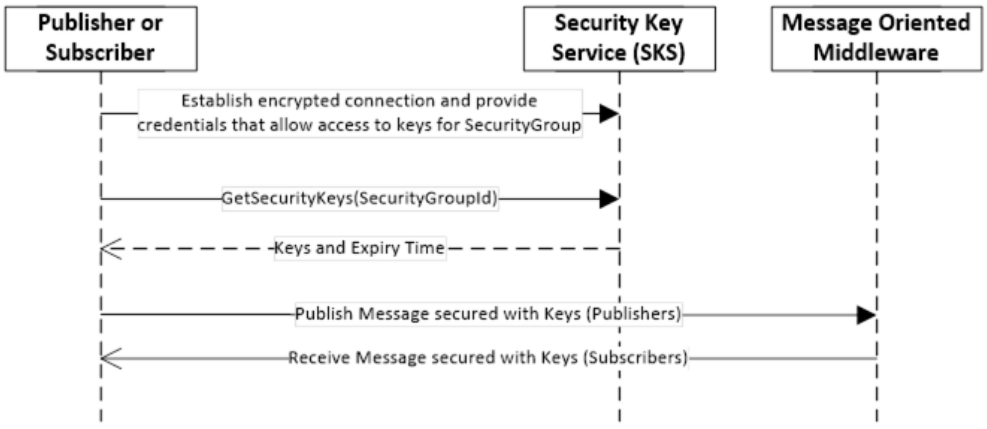


Figure 12: Typical information flow when using the GetSecurityKeys method [27]

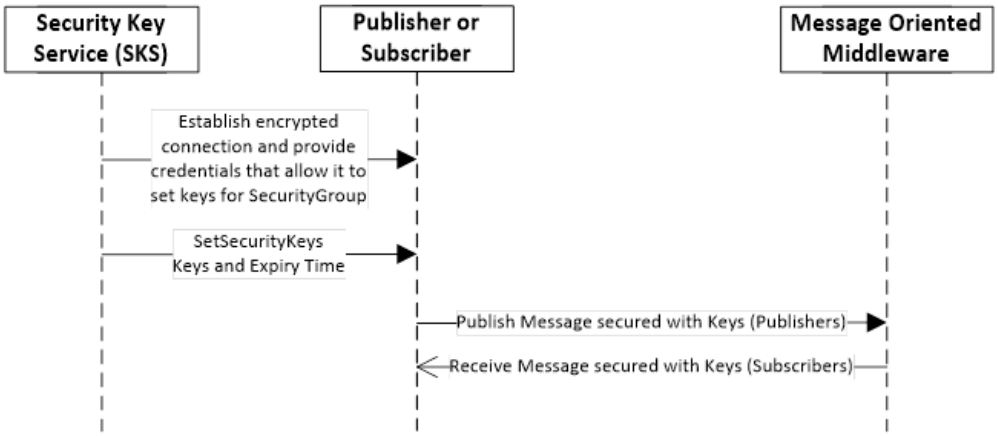


Figure 13: Typical information flow when using the SetSecurityKeys method [27]

As previously mentioned, the PubSub system relies on a message-oriented middleware, specifically a broker, to oversee the message exchange. This is evident in the GetSecurityKeys and SetSecurityKeys diagrams (Figure 12 and Figure 13) where the broker is not directly a part of the handshake with the SKS. Instead, its role is to forward the actual (encrypted) messages between the publisher and subscriber without altering or decrypting their content. In other words, while the SKS facilitates key management by securely exchanging encryption keys with publishers and subscribers, the broker focuses solely on message delivery.

In some cases, a member/device of a security group or their key may be compromised, necessitating dynamic key management. Within the OPC UA framework, two methods are available to address such scenarios:

- **InvalidateKeys:** Used to invalidating the current and all future keys and replace them with an entirely new keyset. Using this method may cause interruptions as keys are immediately invalidated before a replacement is sent. The client calling this function must also have authorisation to modify the SKS groups. While not explicitly defined in the OPC UA documentation, it can be assumed that the client is either the config tool sending the request to the SKS, or the SKS itself sending the request to the members of a security group.
- **ForceKeyRotation:** Unlike InvalidateKeys, this method is less impactful as it forces a key rotation for a single key prior to its KeyLifetime expiration.

3.3 ICS Architecture Model

In general, ICS environments/networks - including those that deploy the OPC UA standard - are structured according to a specific architecture model. Most of these models follow a layered

structure where systems/devices are arranged into distinct levels, with each level defining the specific responsibilities and roles of the devices placed in this level. In this section, we introduce the most widely used ICS model (Purdue), which will be referred to and deployed in the later chapters and analysis of this thesis.

3.3.1 Purdue Model

The Purdue Model, also known as the Purdue Enterprise Reference Architecture (PERA), is a reference model developed in the 1990s [28]. It divides an industrial network into levels, each serving a specific purpose in the control and management of the given industrial process.

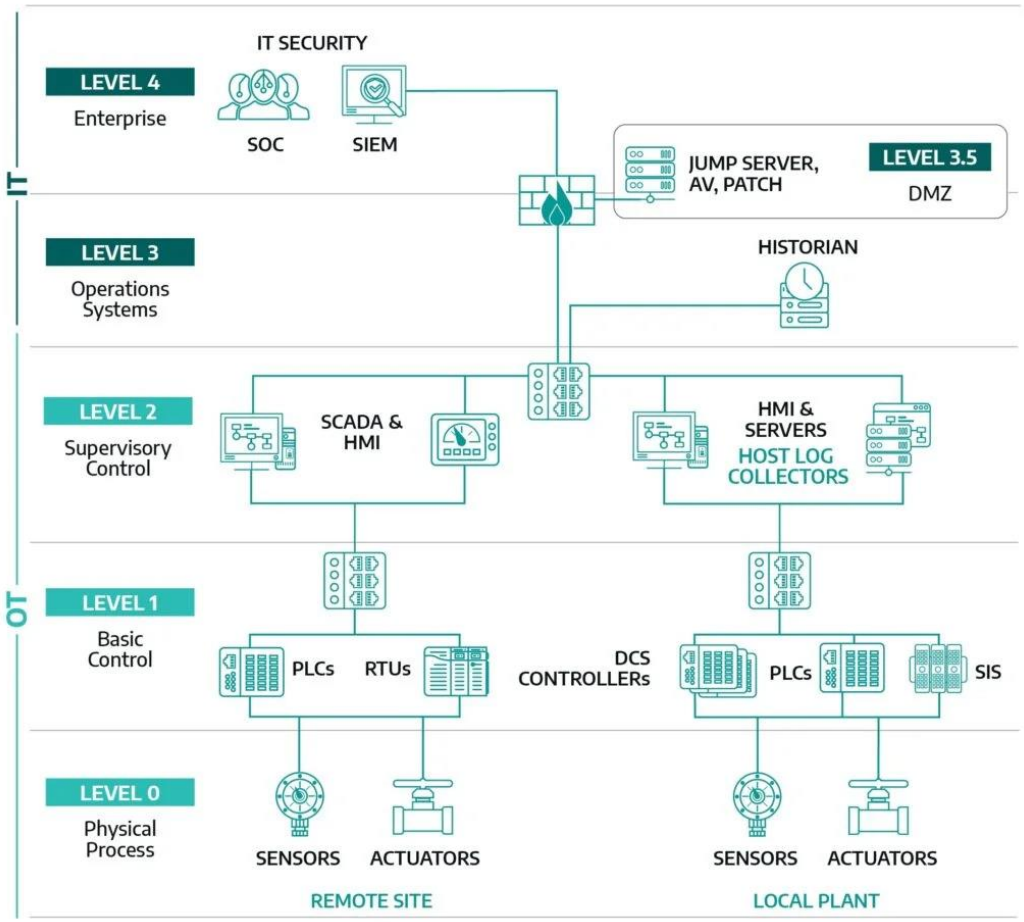


Figure 14: Purdue Model hierarchy [28]

Level 0/1 (Process Control): At the lowest levels are the physical processes, sensors and control devices. Security measures at this level are focused on ensuring the integrity and availability of data from sensors and actuators as well as preventing unauthorised access to control systems and ensuring their proper functioning.

Level 2 (Supervisory Control): Supervisory systems/devices, often referred to as the Supervisory Control and Data Acquisition (SCADA) systems, are found at Level 2. (An OPC UA pub/sub device can also potentially be found at this level.) These systems aggregate data from Level 1. Security at this level involves protecting the SCADA systems from cyber threats and unauthorised control.

Level 3 (Operations): Level 3 is associated with Manufacturing Operations Management (MOM) systems. This level is responsible for coordinating production activities and managing resources. Security measures aim to safeguard production scheduling and coordination processes.

Level 3.5 (DMZ): Between levels 3 and 4 is the Demilitarized Zone, which hosts specialized security appliances such as proxies and firewalls. The OPC UA server and SKS are also placed here. Security at this level serves as an intermediary between IT and OT systems to prevent lateral threat movement.

Level 4 (Enterprise): At the top of the hierarchy is the Enterprise Zone, which encompasses the system's IT infrastructure. At this level, security efforts extend to protect sensitive business data and maintain the integrity of interactions with the broader enterprise environment.

The key system-wide security features of Purdue model are: 1) it stipulates progressively more restricted access to lower network layer, while assuming an upward flow of critical operational data; b) it recommends/envisions the deployment of access control mechanisms, firewalls, and intrusion detection systems at various levels to protect against unauthorised access and potential cyber threats.

Due to its clear and well-structured approach, the Purdue model has been widely adopted as a framework for building of an ICS network architecture and for implementation of security across that architecture. The model helps organisations partition their network into different functional layers, with an appropriate type and level of security deployed at each layer. However, some critics argue that the model may not fully account for the complexities introduced by modern technologies, especially after the introduction of IIoT (which are able to establish direct wireless access to the Internet thereby completely bypassing the upper layers of the Purdue model) [29]. Nevertheless, in most current-day ICS systems, the Purdue model is still considered to be the 'gold standard' of network and security architecture.

3.4 Conclusion

The work of this thesis focuses on ICSs deploying the OPC UA standard due to the standard's widespread popularity, practical versatility, and the important role it plays in Industry 4.0. Additionally, we have chosen to deploy the Purdue model when discussing the organization and deployment of security across an actual ICS network/organization. With these general concepts covered, in the next section we turn our focus on the specific problem of DS attacks on a security group of an PubSub-based OPC UA ICSs. In particular, we explain various strategies available to the adversary when conducting this attack, as well as various ways to mitigate or minimize the impact of the attack.

4 Data Siphoning in OPC UA Systems: Attack Tree Model & Importance of Cryptoperiods

As explained in Chapter 1, many modern-day APT actors prioritize data exfiltration as one of the crucial objectives/steps in their attack strategy – one which, in the case of ICSs, can lead to more effective and precise planning of subsequent stages of an intrusion campaign. However, in systems that deploy encryption, attackers must put additional efforts into the actual extraction of usable information out of the exfiltrated (encrypted) data. As described in Section 1.2, Data Siphoning (DS) refers to an adversary's systematic efforts to collect, exfiltrate, and potentially decrypt sensitive operational (typically in-transit) data from an ICS environment. Given the potential risks associated with a successfully executed DS attack against an industrial system, it is important to build an accurate model of this attack that can ultimately facilitate the development of effective defences. As one of the main contributions of this thesis, we have built a comprehensive attack-tree model of DS against a security group of a PubSub-based OPC UA ICS. (Recall, our research focuses on PubSub OPC UA as this is one of the most popular standards in Industry 4.0.) In Section 4.1 of this chapter, we present the details of our attack-tree model, which depicts different options available to an adversary as well as some key attack execution challenges. Additionally, in Section 4.2, we look at DS from the defender's perspective, and we discuss the important role of periodic encryption-key rotation (i.e., limiting of cryptoperiod length/duration) as an effective way to minimize the ultimate risk and impacts of a DS attack.

4.1 Data Siphoning Attack Tree

The main goal of all attack modelling techniques (AMTs) is to facilitate creation of an effective visual representation of the sequence of steps leading to a successful system compromise (i.e., successful attack execution). AMTs offer benefits to both cyber-security experts and non-experts. For experts, they simplify the analysis of the modelled attack and then facilitate the selection of most appropriate and effective defences. For non-experts, they improve the overall understanding of the modelled attack, including its main stages and possibly different execution strategies, as well as the attack's ultimate objectives.

In today's research literature, attack trees [30] are one of the most popular AMTs. An attack tree is a conceptual tree-like diagram that illustrates the logic and structure of an attack. Attack-tree diagrams are read from the bottom up, with the root node representing the ultimate objective of the attack. Intermediate goals are depicted as leaf nodes, achieved by performing actions

associated with one or more child nodes which can be connected by logic gates (AND, OR, etc.) seen in Figure 15. Each path through the attack tree represents a unique attack vector, guiding the adversary toward achieving the ultimate objective.

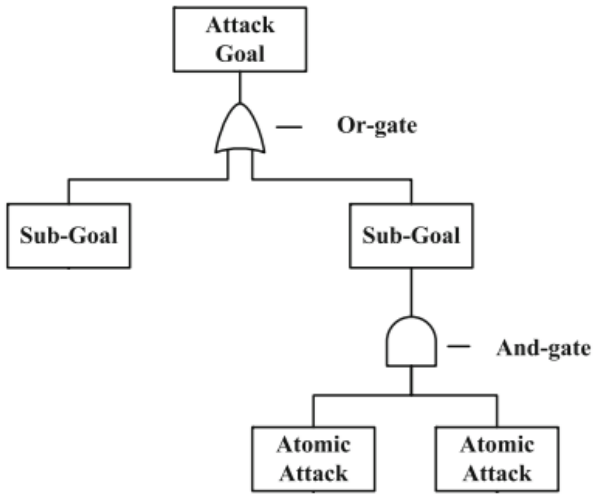


Figure 15: Goal Oriented Tree [31]

4.1.1 Data Siphoning Attack Strategies

As stated in the introduction, our work aims to examine a particular version of DS attack in ICSs – one that targets sensitive **in-transit** operational information exchanged between industrial-system devices. Historically, many communication protocols deployed within ICSs offered little to no protection of data in-transit (data was transmitted in cleartext), thus making DS attacks relatively trivial to execute for an adversary already present in the network [1]. Nowadays, to protect in-transit data against different forms of attacks on confidentiality (including DS), encryption is incorporated in many modern-day ICS protocols.

One of the key assumptions of our work is that the considered DS attack is conducted by an advanced adversary which has already managed to gain access to the target/victim ICS and has established a successful data collection/exfiltration capability. The strategies to achieve these specific objectives (adversary establishing its presence in the target system and being able to collect/exfiltrate data) is beyond the scope of our work.

We have also explained that our work specifically focuses on DS attacks against security groups of PubSub-based OPC UA ICSs. It is important to note that in PubSub OPC UA systems, all messages exchanged within the members of one security group are encrypted with the same symmetric key [32]. Each member (client) of a security group acquires the group key from the so-

called Security Key Server (SKS) through a procedure in which the client must successfully prove its authenticity to the SKS by supplying its digital certificate (including its public key) together with a message signed with its respective private key.

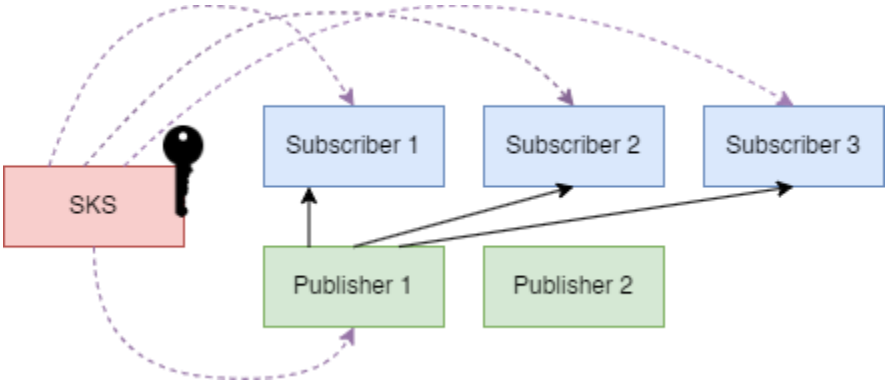


Figure 16: Key transfer between SKS and publishers/subscribers

While the use of a single encryption key for all members of an OPC UA group allows for relatively straight-forward key management with minimal processing overhead, sharing of a single encryption key also carries some inherent risks – especially in the view of DS. As stated earlier, for an adversary that has already gained persistence in the network and has an established data collection/exfiltration capability, data encryption acts as the only remaining defence. It should be noted here that in the case of an OPC UA security group, by gaining access to this key, the adversary would effectively be in the position to compromise the whole group at once – i.e., decrypt every single message sent by every single group publisher.

As for the specific ways an adversary could attempt to get hold of a group key, these are three possible approaches:

A. **Cryptanalytic approach:** An adversary performs cryptanalysis on exfiltrated group date to reveal the key. As will be discussed in Section 4.2.1, NIST SP 800-57 states that using sufficiently strong symmetric encryption algorithms and keys – which many real-world ICS systems already do – would mitigate this attack strategy. *Thus, while possible, this strategy is not likely to be deployed by the adversary.*
B. **Acquire Key by Compromising a Group Member:** A more probable attack strategy would be to try to compromise of one of the security group members, which would then grant the adversary direct access to a security key. There are two types of vulnerabilities which the adversary can try to exploit in one or more group members:

B.1: Software vulnerabilities with impact on system integrity or confidentiality that could facilitate extraction of the group key from the compromised group member either at runtime (from RAM, cache or CPU bus) or at-rest (from ROM memory) [32].

B.2: Procedural vulnerabilities such as inadequate security policies or insufficient employee training, that can also lead to key compromise. For instance, an adversary might obtain authentication credentials from an employee or contractor, allowing them to log into a group member remotely. With the necessary user privileges, the adversary could then extract the group key from the member's runtime or at-rest memory.

C. **Impersonating a Legitimate Client to Obtain the Key Directly from the SKS**: As highlighted in [32], the third method an adversary can attempt is to target an SKS by impersonating a legitimate client and requesting to join the target security group. This attack could succeed if: a) the SKS server lacks strong authentication mechanisms (e.g., does not require clients to provide a signed digital certificate), or b) the SKS does require signed digital certificates, but the adversary manages to obtain both the certificate and the corresponding private key from a legitimate client, possibly through a client-device compromise, exploiting the software or procedural vulnerabilities discussed in approach B.

Approach A can be considered a passive attack (encrypted data is only collected and exfiltrated from the target system for further cryptanalysis). Approaches B and C, however, are active attacks. Besides exfiltration, the adversary is required to proactively detect and attempt to exploit vulnerabilities in the devices of the target system.

4.1.2 Attack Tree

With all main attack approaches identified, we form a comprehensive attack tree representation of DS, as shown in Figure 17.

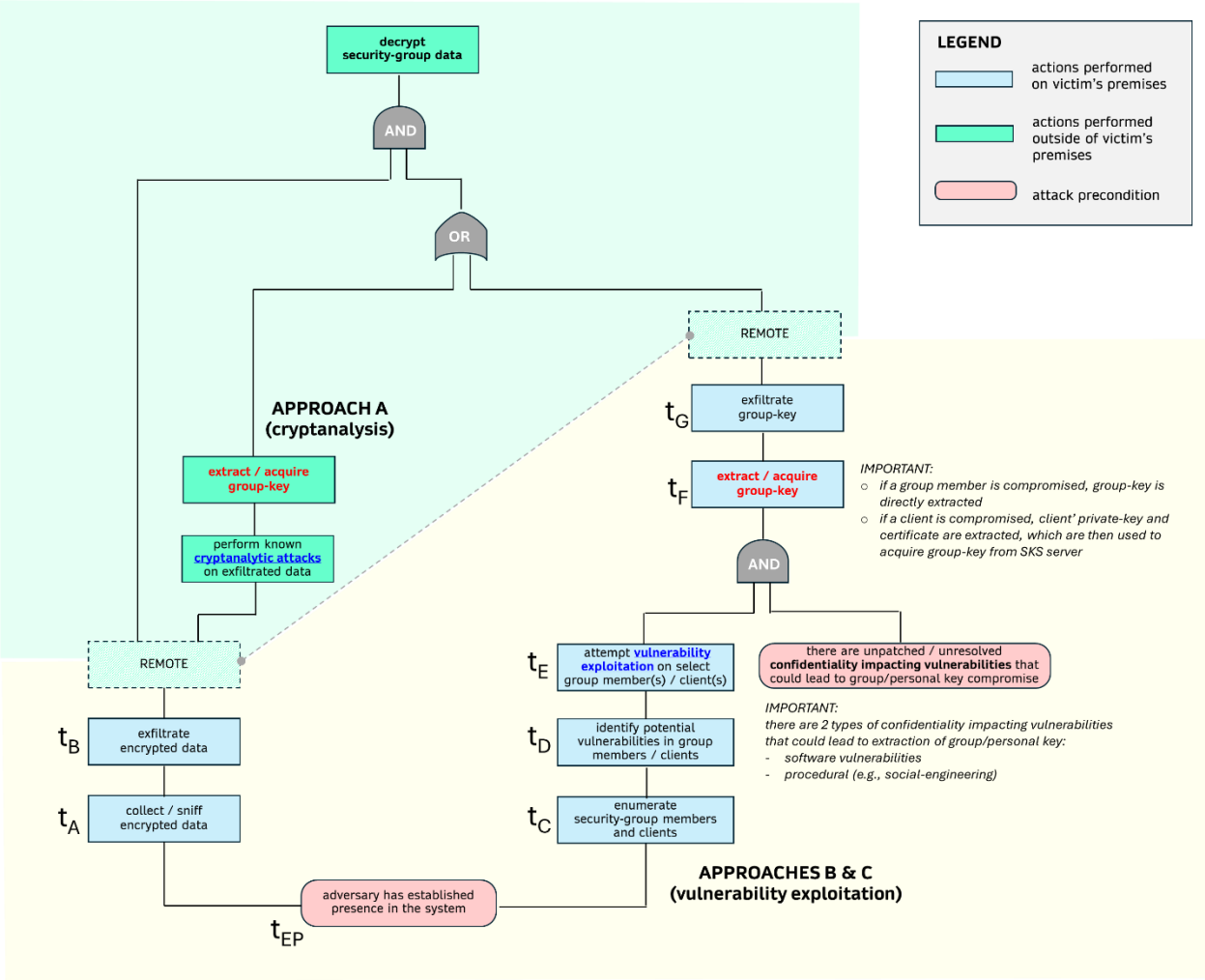


Figure 17: Attack tree model of a DS attack against an OPC UA security group
Time points are defined in Section 4.2.2

Beginning from the bottommost node, the precondition for a successful attack is that the adversary is already present in the target system. Following the leftmost path, the first two steps (collect/sniff encrypted data and exfiltrate encrypted data) are mandatory for attack completion, as indicated by the "AND" operator preceding the root node. The first branch in the path is Approach A (cryptanalysis) which can be conducted remotely. Referring to NIST SP 800-82r3, the standard states "a cryptographic key (used by an organization) should be long enough so that guessing it or determining it through analysis takes more effort, time, and cost than the value of the protected asset" [1]. Assuming the target ICS organisation follows proper security standards, an adversary would not have the interest nor capacity to use this approach as the resources required would far exceed the potential gain.

Pivoting to the rightmost paths, Approaches B and C are to some extent similar to one another, as they both fall under the same umbrella of 'vulnerability exploitation'. Unlike Approach A, the following is true for B and C:

1) These attack strategies must be done on-site and cannot be completed remotely. That is, the adversaries must remain in the network to enumerate, identify and attempt vulnerability exploitation on select device(s).
2) These strategies assume that there are devices (i.e., group members or other clients) with unpatched or unresolved vulnerabilities, which can also include zero-day exploits that are unknown to the operator and do not have a CVE associated with them.

Besides the three outlined approaches for acquiring of the group key, one must also be cognizant of the timeline of the DS attack – both from the adversary's and as well as the defender's perspective. Namely, while a longer time in the victim network will increase the amount of (encrypted) data collected by the adversary, it will also increase the chance of adversary detection. In Section 4.2.2, we will go more in-depth into the various factors that influence the adversary's decision on when to ultimately attempt/execute strategy B or C, relative to the start of data collection and exfiltration process. As the discussion will show, one of the critical factors influencing this decision is the length of security group's cryptoperiod – i.e., the time interval during which one (the same) encryption key is used by the group and before it is replaced with another key.

4.1.2.1 Defensive Strategy

Before we move further into the discussion about the time-related aspects of a DS attack, this section will briefly address the practical significance of the attack tree from Figure 17 for the defender. Namely, the derived DS attack tree suggest that if the defender's goal is to prevent DS from happening, the defender should focus on stopping the adversary from:

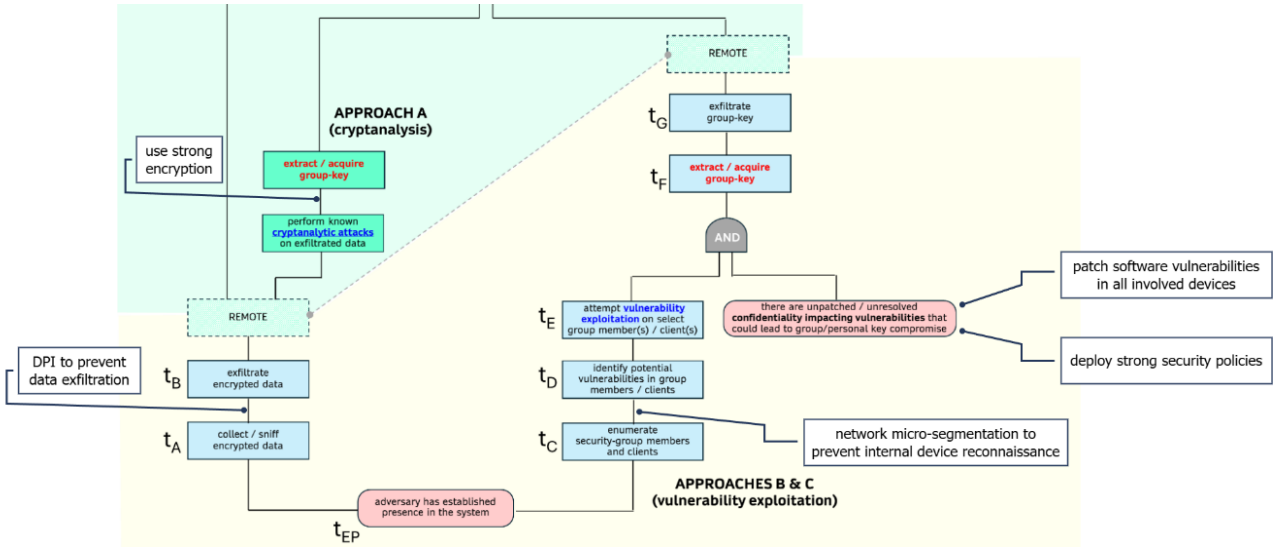
a) 'Moving' along all 'OR' branches of the tree (i.e., all branches that are connected with an OR gate). For example, given that strategy A, B, C are connected with an OR gate, the defender should stop the adversary from moving along (executing) either of the three strategies to fully prevent the DS attack from happening.

b) Moving along at least one of the 'AND" branches of the tree (i.e., one of two branches connected with an AND gate). For example, in the bottom-right part of the tree, there is an AND gate connecting the '*adversary's vulnerability discovery*' with '*precondition that there are*

unpatched vulnerabilities in the system'. So, in this case, as long as one of the branches is 'terminated' (e.g., the administrator/defender ensures that there are no unpatched vulnerabilities in the system) the attack will ultimately fail – even if the adversary manages to successfully move along the other branch (e.g., vulnerability discovery).

Figure 18 illustrates different defensive measures that could be implemented by the defender to prevent the adversary from moving along different branches of the DS attack tree:

- Deep Packet Inspection (DPI)[2] can be employed to prevent data exfiltration by analysing network traffic and blocking data from leaving the system when signs of malicious activity are identified.
- Strong encryption can invalidate cryptanalysis (Approach A) by making it computationally infeasible to break the deployed encryption key(s).
- Network micro-segmentation can limit an adversary's ability to perform internal device reconnaissance (Approaches B & C) by isolating devices and restricting lateral movement within the network.
- Regular patching of all involved devices can eliminate software vulnerabilities and possibility of their exploitation (which could eventually lead to compromise of the group key).
- Deployment of strong security policies can also significantly reduce the probability of device (and, consequently, the group key) compromise.



[2] A security method that inspects the contents of data packets as they pass through a network checkpoint.

Figure 18: Potential defensive techniques for a DS attack

However, it is important to recognise that even if all the above techniques are implemented a system, there is still a chance that the system falls victim to a DS attack. For example, there is always a possibility that the adversary discovers a new zero-day vulnerability in one of the group members (which the system administrator is not aware of and thus cannot patch), and through that vulnerability the adversary eventually gets hold of the encryption key. All of this suggest that in addition to implementing all required defences, organizations should also implement measures which, in the case of unpreventable DS attacks, would minimize the ultimate losses/impact of those attacks. According to NIST SP 800-82r3 [1] and NIST SP 800-57 [8] standards, one way to minimize the potential impact of DS (which cannot be prevented) is through periodic rotation of encryption key(s), which will be explored in the following section.

4.2 Role of Cryptoperiods in DS Attacks

4.2.1 Industry Standards Pertaining to Cryptography and Cryptoperiods

The key lifetime, or cryptoperiod, is defined in NIST SP 800-57 as “the time span during which a specific key is authorized for use or in which the key(s) for a given system or application may remain in effect” [8]. It is assigned for various reasons, with one important purpose being to limit the exposure of encrypted data in the event of a (data/key) compromise - such as in the case of a DS attack.

As for how an organization should go about setting/determining the actual cryptoperiod length for their encrypted data, NIST SP 800-57 provides several general recommendations. Namely, the standard states that for a cryptoperiod to be deemed effective, it should satisfy several criteria, as enlisted below:

- Limit the overall amount of data encrypted with the key, so as to limit the amount of information (plaintext /ciphertext pairs) potentially available to an adversary attempting to reveal the key through cryptanalysis;
- Limit the amount of sensitive-information exposure in the case of the key compromise;
- Limit the time available for the compromise of the key though non-cryptanalytic means, such as compromise of physical, procedural, and logical components of the target system that use/store the key (e.g., compromise of devices that use/hold the key).

As previously mentioned, the standard also points out that the vulnerability to cryptanalytic attacks can generally be minimized by employing strong cryptographic algorithms (potentially even quantum-resistant algorithms) and sufficiently long encryption keys – which could practically render R.1 redundant. The standard further emphasizes that in systems with strong cryptographic algorithms and key sizes, non-cryptanalytic attacks are likely to be more effective and allow adversaries to gain access to encryption key(s) with far less expenditure of time and resources. Thus, in such systems, requirements R.2 (amount of sensitive information encrypted) and R.3 (possibility that adversary retrieves the key from a compromised system component) should be the ones ultimately impacting cryptoperiod selection.

Continuing, the standard delineates three categories in which consequences can be measured. These categories provide a general guideline to assess the potential fallout of a key compromise:

- *Sensitivity of the data being safeguarded:* Higher sensitivity (e.g., operational control commands or critical infrastructure configurations) necessitates shorter cryptoperiods to minimize exposure if a key is compromised.
- *Criticality of related processes:* If a compromised key could disrupt mission-critical operations or industrial processes, shorter cryptoperiods are essential to prevent significant operational impact.
- *Cost of recovering from a key compromise:* As the recovery cost increases, the cryptoperiod should decrease to limit potential damage and reduce recovery complexity.

Unfortunately, while generally useful, the above-mentioned standards and recommendations fall short of providing a clear directive on how to determine the actual (numerical) cryptoperiod duration for an actual encryption key deployed in an actual system. The closest to an explicit guideline are the following two very general recommendations, also from NIST SP 800-57:

- Cryptoperiods deployed for the encryption of large volumes of data should be in the order of *a day or a week*, while in the case of smaller volumes of data should be *up to two years*.
- The (actual) determination of cryptoperiods should “involve the risk and consequences of (key) exposure”.

Now, from the security perspective alone (and in the absence of explicit methodologies and tools to calculate optimal cryptoperiods), one may argue that selection of very short cryptoperiods is the best and most obvious choice – as very short cryptoperiods are likely to satisfy both R.2 and R.3 in **Error! Reference source not found.**. However, while better from the security standpoint, there are many serious technical issues potentially associated with very short cryptoperiods (i.e., very frequent key rotations). For example:

1. **Increased Overhead:** Very short cryptoperiods lead to frequent key rotations, which increases the amount of communication and processing overhead (i.e., amount of control information exchanged to ensure that all keys are received and deployed on time by all communication parties involved). This can be particularly disruptive in ICS as many devices are resource constrained.
2. **Synchronization Risks:** Frequent key rotations demand perfect synchronization between all communicating parties. If any party fails to switch to the new key at the right time, there is a risk of data loss, incomplete transmissions, or general interruptions in communication.
3. **Manual Key Replacement:** In certain environments (or segments of an environment), such as remote or air-gapped ICS, key replacement might require manual human intervention. In these cases, very short cryptoperiods may not be practically feasible due to the logistical challenges involved in physically accessing the device(s) that deploy the keys.

Therefore, a well-chosen cryptoperiod length must strike a balance between reducing security risks and accommodating the technical limitations of the system. In other words, an optimal cryptoperiod must not only minimize the risk of data compromise but should also ensure that performance degradation or operational disruptions are avoided.

4.2.2 Cryptoperiods and Data Siphoning

Analysing the critical time-points related to a DS attack - from both the attacker's and the defender's perspective - can shed more light on an appropriate cryptoperiod length. This section explores how timing influences the attack's outcome, while highlighting important factors that cannot be fully depicted by the attack tree in Figure 17.

4.2.2.1 Adversary's Perspective

From the adversary's perspective, there are five significant time-points related to a DS attack, also shown in Figure 19:

- t_0 – beginning of the data's cryptoperiod;
- t_A – time-point when the adversary starts to sniff/collect encrypted group data;
- t_F – time-point when the adversary successfully acquires the group key by compromising one of the group's members;
- t_N – end of the data's cryptoperiod, and when the group switches to a new key;
- t_{Det} – time-point when the attack is detected, and necessary measures to interrupt the attack are taken by the defender.

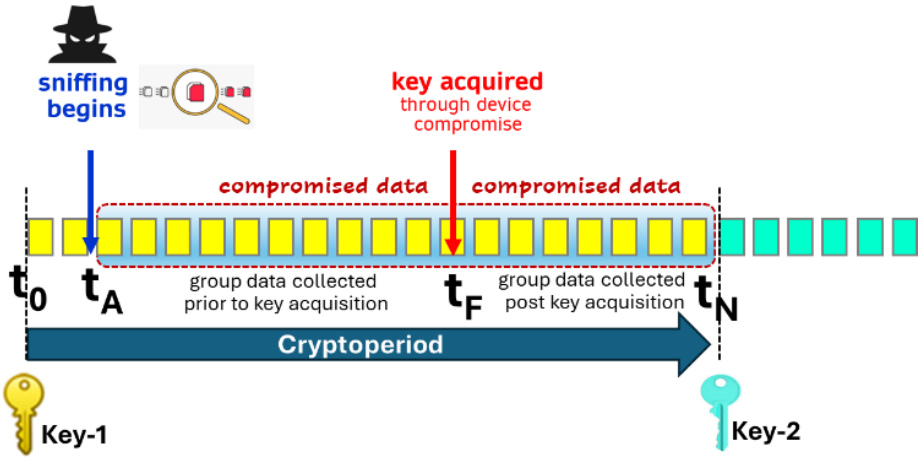


Figure 19: Impact of longer cryptoperiods on successful attack (key acquisition)

The time-points t_0 and t_N mark the beginning and end of a cryptoperiod, and after t_N a new cryptoperiod will begin with its own t_0 and t_N. t_A and t_F are time-points actively controlled by the adversary. An adversary conducting a DS attack would want t_A to be as close as t_0 as possible (with $t_0 = t_A$ being the ideal scenario), so when the group key is finally acquired at time t_F, the adversary would be able to decrypt a whole cryptoperiod worth of data. Note, once the group key is acquired, with this key the adversary will be able to decrypt not only the previously collected data matching the given key (i.e., data transmitted between t_A and t_F), but also any subsequently sniffed/collected data up to the end of the cryptoperiod (i.e., data transmitted between t_F and t_N).

Another important time-related aspect of the attack which the adversary needs to decide on is the actual delay between t_A and t_F – both of which are under the adversary's direct control. Practically, the question here is: how long should the adversary be passively collecting and

exfiltrating the group data before actively attempting to acquire the key by compromising one of the group members or clients? Clearly, if the adversary delays the attempt to acquire the group key too long, he might face the situation where $t_F > t_N$. This would mean that the key ultimately acquired is a new key, and thus of no use for the decryption of data collected/exfiltrated between t_A and t_N. In other words, by delaying t_F too long, the adversary would run the risk of potentially wasting all the effort put into collecting and exfiltrating group data of (what in the end may turn out to be) a previous/passed cryptoperiod.

Conversely, there may be potential advantages for the adversary in delaying t_F. Namely, in a reasonably secured system, any attempt to proactively engage with and/or compromise any of the system components also runs the risk of triggering the system's intrusion detection and defences, which in turn could potentially jeopardize the attack's very mission/execution. From the perspective of a DS attack in Figure 19, this implies that the adversary may not only want to delay t_F relative to t_A (so as not to trigger any intrusion alarms too early), but they should also proceed with attempts to acquire the key very cautiously to minimize the probability of attack detection and containment, and thereby delay t_{Det} as much as possible.

4.2.2.2 Defender's Perspective

From the perspective of DS defender, the worst-case attack scenario is the one where the adversary manages to siphon (collect and exfiltrate) all the data from a potentially long cryptoperiod, as well as successfully obtain the key for this data, before being detected. Two possible measures the defender could take to prevent such a worst-case scenario and minimize the overall attack impact include:

a) Step up the system's intrusion detection efforts to bring t_{Det} as close to t_F as possible, and thus interrupt/stop the attack as early as possible. This could be achieved through close monitoring of group members and other clients in the systems for the earliest indicators of compromise.

b) Consider reducing the length of cryptoperiod(s) so as to reduce the amount of data that actually gets compromised once the adversary successfully acquires the encryption key – i.e., limit the amount of data encrypted with each particular key, as illustrated in Figure 20.

As for measure b), the decision on how much to shorten the cryptoperiod should be carefully made. As previously mentioned, while better from the security point of view, more frequent key rotations and short(er) cryptoperiods imply more communication and processing overhead for the SKS server as well as their respective security group members and could cause serious synchronization and performance issues in the system. Ultimately, a well-chosen cryptoperiod should be such that it adequately reduces the risk and impact of a potential DS attack on the target/victim organization, and at the same time complies with the performance expectations and limitations of its security group and the system as a whole.

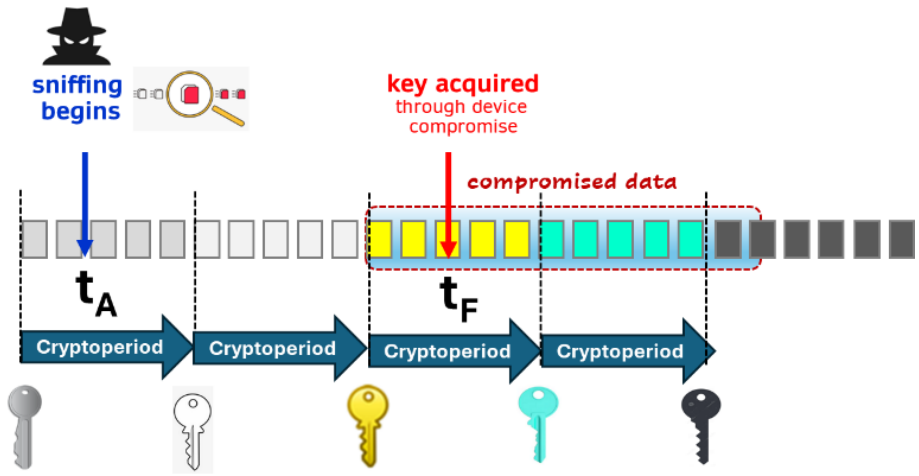


Figure 20: Impact of shorter cryptoperiods on attack outcome

4.3 Conclusion

In this chapter we have presented our comprehensive attack tree model of DS attacks on a security group in a PubSub-based OPC UA industrial system. The model depicts different possible attack strategies, as well as the actual conditions and sequence of steps that each strategy is dependent on. Also in this chapter, while looking at DS from the defender's perspective, we have highlighted the importance of periodic key rotation (i.e., the use of relatively short cryptoperiods) as a measure that can minimize the ultimate impact of a successful DS attack on the target organization. While the existing security standards provide general insights into key rotation, or limiting of cryptoperiod lengths, they lack precise guidance on how cryptoperiod(s) assigned to an actual system should be determined/calculated. The next chapter addresses this gap by presenting our novel risk-based methodology for calculation of optimal cryptoperiods in real-world ICSs.

5 Risk-Based Framework for Determining Optimal Cryptoperiods for Security Groups in OPC UA Environments

After the discussion of Chapter 4, where we analysed various aspects of cryptoperiod management as outlined in the NIST standards and the OPC UA documentation, one thing is clear – the actual risk associated with the compromise of an encryption key (and, ultimately, disclosure of the respective data) should be the main parameter driving/controlling the length of this key's cryptoperiod. Intuitively, it should be easy to see that in high-risk situations, (e.g. where the key in question is used to encrypt critical data/information), the respective cryptoperiods should be short(er), and vice versa. However, numerous questions pertaining to this conclusion remain, including: How 'shorter' should the cryptoperiod be? What the actual relationship between the risk and cryptoperiod should look like? Does the risk impacting the optimal cryptoperiod length depend only on the nature of encrypted data, or it is impacted by other system parameters and conditions? Unfortunately, all these questions remain largely unanswered not only in the relevant industry standards, but also in the existing research literature.

One of the main objectives of our work was to answer the aforementioned questions and develop an explicit framework for optimal adjustment of cryptoperiods assigned to in-transit data of a real-world ICS. In this chapter, we present our novel risk-based methodology for calculation of optimal cryptoperiod(s) assigned to an OPC UA security group. The methodology aims to strike the right balance between adequate system security and practical feasibility of deploying short(er) cryptoperiods. According to our knowledge, this is the first research study attempting to address as well as solve the problem of optimal cryptoperiod calculation in an industrial PubSub OPC UA environment. Though, the general methodology presented in this chapter can be readily extended and applied to any other type of industrial-control, critical-infrastructure or broader IT systems.

5.1 Risk-Cryptoperiod Relationship

5.1.1 General Risk-to-Cryptoperiod Relationship

It was explained in Chapter 4 that the main industry standard pertaining to the management of encryption keys (NIST SP 800-57) provides some general guidelines on recommended '**upper-bound**' (i.e., maximum) cryptoperiod lengths across different encryption/application scenarios – as shown in Figure 21. Note, the scenario considered in this work - DS attacks on encrypted in-transit data, where one symmetric encryption key is shared among multiple PubSub OPC UA security group members - corresponds to *Key Type 9* in Figure 21. That is, the longest recommended cryptoperiod for such a scenario is 1 year, which we will annotate as $T_{CP\text{-}max}$.

At the same time, NIST SP 800-57 recognizes that in an ideal scenario, a cryptoperiod length should be shorter than $T_{CP\text{-}max}$, and ideally adjusted to the actual risk level(s) of the respective environment (or, in the case of a PubSub OPC UA system, to the risk level of the respective security group).

Key Type	Cryptoperiod	
	Originator Usage Period (OUP)	Recipient Usage Period
1. Private Signature Key	1-3 years	
2. Public Signature Key	Several years (depends on key size)	
3. Symmetric Authentication Key	≤ 2 years	≤ OUP + 3 years
4. Private Authentication Key	1-2 years	
5. Public Authentication Key	1-2 years	
6. Symmetric Data Encryption Keys	≤ 2 years	≤ OUP + 3 years
7. Symmetric Key Wrapping Key	≤ 2 years	≤ OUP + 3 years
8. Symmetric and asymmetric RNG Keys	Upon reseeding	
9. Symmetric Master Key	About 1 year	
10. Private Key Transport Key	≤ 2 years[13]	

Figure 21: Suggested cryptoperiod for key types [8]

In our work, we additionally recognize that the adjustment of a cryptoperiod length should also have a well-defined '**lower-bound**', which is determined by the system's performance and/or

implementation requirements. For example, in a system where many devices require manual encryption-key updates (e.g., an ICS with remote or air-gapped segments), it may be determined that under no circumstances the cryptoperiod of a security group should be shorter than 24h, forcing daily rotation of encryption keys. We will annotate such lower-bound on cryptoperiod length as $T_{CP\text{-}min}$.

Based on the above, and in the context of a DS attack, we conclude that the cryptoperiod of a security group ($T_{CP\text{-}SG}$) should be set somewhere in the range [$T_{CP\text{-}min}$, $T_{CP\text{-}max}$], depending on the estimated risk associated with a potential compromise of this group's data due to the group's key compromise. We annotate this risk as $R_{SG\text{-}KC}$, and we illustrate the recommended relationship between $T_{CP\text{-}SG}$ and $R_{SG\text{-}KC}$ (where $T_{CP\text{-}SG}$ is controlled by $R_{SG\text{-}KC}$, or rather $T_{CP\text{-}SG} = f(R_{SG\text{-}KC})$) in Figure 22. The details of $R_{SG\text{-}KC}$ calculation are presented in the upcoming sections.

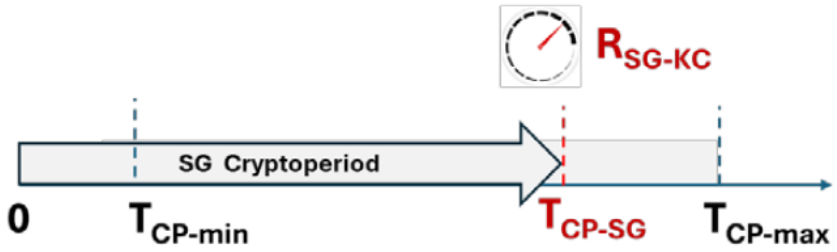


Figure 22: Impact of key-compromise Risk on $T_{CP\text{-}SG}$ in relation to $T_{CP\text{-}min}$ and $T_{CP\text{-}max}$

Now, when it comes to the actual relationship between $T_{CP\text{-}SG}$ and $R_{SG\text{-}KC}$ (i.e., the nature of $R_{SG\text{-}KC}$ impact on $T_{CP\text{-}SG}$) we propose two different types of scaling: linear and exponential.

1) **Linear scaling**, seen in equation (1), begins with a baseline minimum cryptoperiod ($T_{CP\text{-}min}$). It then adds an increment that is proportional to (1 – $R_{SG\text{-}KC}$), up to a maximum of ($T_{CP\text{-}max}$ – $T_{CP\text{-}min}$). In such a relationship, as $R_{SG\text{-}KC}$ increases, the cryptoperiod linearly decreases, ensuring that higher risks are mitigated by more frequent key rotations. Conversely, lower risk levels allow for longer cryptoperiods, optimizing operational efficiency while maintaining security.

$$T_{CP-SG} = T_{CP-min} + (1 - R_{SG-KC}) \times (T_{CP-max} - T_{CP-min}) \quad (1)$$

A potential concern with using the linear approach from (1) is that the same amount of risk change (e.g., risk increase of ΔR) will result in the same amount of cryptoperiod change (e.g., cryptoperiod decrease of ΔT) regardless of the risks' absolute value. As a result of this, with linear scaling – as shown in Figure 23 – even very high-risk levels (e.g., 0.8 or 0.9) will end up producing rather long cryptoperiods (74 or 37 days, respectively), which may appear inadequate for such critical levels of 'insecurity' in the system.

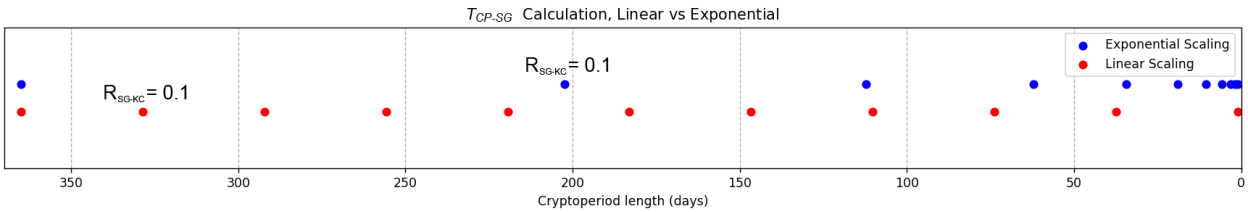


Figure 23: Linear vs Exponential scaling assuming $T_{CP\text{-}min}$ = 1 day and $T_{CP\text{-}max}$ = 1 year. $R_{SG\text{-}KC}$ is annotated with blue/red dots, and increases in intervals of = 0.1, left to right.

2) An alternative way of formulating the relationship between $T_{CP\text{-}SG}$ as $R_{SG\text{-}KC}$ is by using **exponential scaling**, depicted in (2).

$$T_{CP-SG} = T_{CP-max} \times \left(\frac{T_{CP-min}}{T_{CP-max}}\right)^{R_{SG-KC}} \quad (2)$$

The exponential scaling approach starts with a baseline cryptoperiod ($T_{CP\text{-}max}$) but then deploys an exponential factor to adjust the cryptoperiod length based on the risk level ($R_{SG\text{-}KC}$). Given that the ratio $T_{CP\text{-}min}$ / $T_{CP\text{-}max}$ is always less than 1 (since $T_{CP\text{-}min}$ < $T_{CP\text{-}max}$), raising this ratio to the power of $R_{SG\text{-}KC}$ results in more dramatic drop in $T_{CP\text{-}SG}$ as $R_{SG\text{-}KC}$ increases, compared to the linear scaling.

To approximate how (2) actually changes $T_{CP\text{-}SG}$ for every 0.1 increase in $R_{SG\text{-}KC}$, we can deploy the well-known '*approximation using differentials near a known function value*' rule [33] that states:

$$f(x + \Delta x) = f(x) + f'(x) \times \Delta x \quad (3)$$

In our case, with x = $R_{SG\text{-}KC}$ and f(x) = $T_{CP\text{-}SG}$ ($R_{SG\text{-}KC}$) as per exponential scaling function in (2), (3) becomes

$$T_{CP-SG}(R_{SG-KC} + 0.1) = T_{CP-SG}(R_{SG-KC}) + T_{CP-SG}(R_{SG-KC}) * \ln\left(\frac{1}{365}\right) * 0.1 \quad (4)$$

which further implies

$$T_{CP-SG}(R_{SG-KC} + 0.1) \approx T_{CP-SG}(R_{SG-KC}) - 0.59 * T_{CP-SG}(R_{SG-KC}) \quad (5)$$

Finally, we obtain

$$T_{CP-SG}(R_{SG-KC} + 0.1) \approx 0.41 * T_{CP-SG}(R_{SG-KC}) \quad (6)$$

From equation (6), we conclude that the exponential scaling relationship in (2) approximately halves the cryptoperiod length ($T_{CP\text{-}SG}$) every time the risk ($R_{SG\text{-}KC}$) increases by 0.1, which is also depicted in Figure 23 (note that this only applies for exponential scaling using the default $T_{CP\text{-}min}$ and $T_{CP\text{-}max}$ values – for a more comprehensive analysis, visit Appendix B). As a result of this 'rapid' adjustment/shortening of cryptoperiod length as a function of risk, in systems/situations where the risk of security group's key compromise is high (e.g., 0.7 or above), the respective cryptoperiod will be shortened to only a few days. From the security perspective, such a short cryptoperiod (i.e., frequent replacement of encryption keys) will certainly contribute to better attack resistance, or (in the case of attacks that cannot be prevented) lower attack impact – as was previously explained in Chapter 4.

Like linear scaling, exponential scaling does not come without a drawback. Although it puts a strong emphasis on security, it adversely affects operational overhead and efficiency. In certain cases (when risk is low), small changes can drastically reduce the cryptoperiod length, thus resulting in key rotation that may be more frequent than necessary. For example, as can be seen for Figure 23, for a low risk of $R_{SG\text{-}KC}$ = 0.1, the exponential scaling shortens the respective cryptoperiod from 365 to about (only) 200 days.

We will close this discussion by stating that the ultimate choice of $R_{SG\text{-}KC}$ to $T_{CP\text{-}SG}$ scaling should be based on the needs of a particular environment. In environments in which operational efficiency (i.e., minimizing of the operational demands and overhead) is critical, it may be beneficial to adopt the linear approach. In environments where security is critical, the exponential

scaling may be more appropriate. Since system security has been the primary focus of our research, we opt to utilise the exponential scaling approach in the remainder of our work (i.e., for the purposes of our framework). Though, the framework could easily be adjusted to employ the linear scaling approach, if required.

5.1.2 Calculation of Risk Required to Find Optimal Cryptoperiod of an OPC UA Security Group Under DS Attack

Now that we have explored the general functional relationship between $R_{SG\text{-}KC}$ and $T_{CP\text{-}SG}$, and assuming the values of $T_{CP\text{-}min}$ and $T_{CP\text{-}max}$ are provided directly by the operator, $R_{SG\text{-}KC}$ is the only 'missing' (unknown) parameter in equations (1) and (2) required for calculation of the optimal cryptoperiod $T_{CP\text{-}SG}$.

Recall, here, that the ultimate objective of our work is to build a framework for the calculation of optimal cryptoperiod(s) in the context of a DS attack. Hence, in this subsection, we detail our novel analytical approach for determining the value of $R_{SG\text{-}KC}$ in the case of a DS attack on an OPC UA security group, which will then allow us to calculate the group's respective optimal cryptoperiod using equation (2).

We begin our quest for an analytical expression for $R_{SG\text{-}KC}$ by first referring to the well-known general risk formula [34]:

$$R = P \times I \tag{7}$$

where P is the likelihood of harm occurring (i.e., **P**robability of a successful attack execution) and AI is the degree of harm (i.e., the **I**mpact of the attack).

From (7), and considering the nuances of a DS attack on a particular security group (as discussed in Chapter 4), we obtain the following initial expression for $R_{SC\text{-}KC}$:

$$R_{SG-KC} = P_{SG-KC-attempt} \times P_{SG-KC-succ} \times AI_{SG-KC} \tag{8}$$

where:

$P_{SG\text{-}KC\text{-}attempt}$ is the probability that an adversary (assumed to be already present in the network) will attempt to **C**ompromise the encryption **K**ey of the given **S**ecurity **G**roup.

$P_{SG\text{-}KC\text{-}succ}$ is the probability that the given adversary actually succeeds in **C**ompromising the **K**ey of the given **S**ecurity **G**roup. (The proceeding Section 5.2 discusses this probability in more detail.)

$AI_{SG\text{-}KC}$ is the overall **A**dverse **I**mpact incurred on the ICS (i.e., the respective organization) if/once the **K**ey of the given **S**ecurity **G**roup is **C**ompromised, and with it the respective data is exposed to the adversary. (Our methodology for calculation of $AI_{SG\text{-}KC}$ will be explored in Section 5.3.)

Note that our attack tree model from Chapter 4 (Figure 17) considers the worst-case scenario in which it is 100% certain that the adversary with established presence in the target network will attempt to compromise the key, which implies $P_{SG\text{-}KC\text{-}attempt} = 1$. With this assumption/simplification, (8) becomes:

$$R_{SG-KC} = P_{SG-KC-succ} \times AI_{SG-KC} \tag{9}$$

5.2 Methodology for Calculation of Probability $P_{SG\text{-}KC\text{-}succ}$

Referring to Figure 17 in Chapter 4, a DS adversary can attempt to acquire the data's encryption key either through approach A or approaches B & C. Consequently, we can write

$$P_{SG-KC-succ} = P_A \times P_{A-succ} + P_{BC} \times P_{BC-succ} \tag{10}$$

where:

P_A is the probability that the adversary specifically pursues Strategy A (to acquire the key through cryptanalysis), as discussed in Chapter 4.

$P_{A\text{-}succ}$ is the probability that the adversary succeeds in decrypting the exfiltrated data through Strategy A.

P_{BC} is the probability that the adversary specifically pursues Strategy B or C (to acquire the key through vulnerability exploitation of a group member or client), as discussed in Chapter 4.

$P_{BC\text{-}succ}$ is the probability that the adversary succeeds in obtaining the group key though Strategy B or C.

Note that $P_A + P_{BC}$ must result in an output of 1. That is, an adversary which is planning an attack on a system must use either of the three already specified approaches to conduct their attack. For this framework, we assume that P_A is 0, for reasons explained in Section 4.1.1 (either because the system uses a strong encryption algorithm and/or because the adversary does not have sufficient capability for effective cryptanalysis). This implies that the entirety of the attack probability is attributed to Strategies B and C, hence $P_{BC} = 1$. This results in the following simplification of (10):

$$P_{SG-KC-succ} = P_{BC-succ} \tag{11}$$

Section 4.1.1 also discussed that executing Strategy B and C can be accomplished by attempting to exploit a software vulnerability ($P_{VE\text{-}procedural\text{-}succ}$) or procedural vulnerability ($P_{VE\text{-}procedural\text{-}succ}$) in the target system. In this work (our framework), we will consider a determined adversary first attempts to exploit one type of vulnerability, and if that attempt fails, moves on attempting to exploit the other – ultimately trying both, as shown in the below figure (Figure 24). The below figure also shows that in such a case, and regardless of the adversary's actual preferences in terms of the order of steps (i.e., irrespective of the values of $P_{VE\text{-}procedural\text{-}first}$ and $P_{VE\text{-}software\text{-}first}$), the ultimate probability that the adversary succeeds in obtaining the key through Strategy B or C ($P_{BC\text{-}succ}$) boils down to:

$$P_{SG-KC-succ} = 1 - (1 - P_{VE-software-succ})(1 - P_{VE-procedure-succ}) \tag{12}$$

where:

$P_{VE\text{-}software\text{-}succ}$ is the probability that an adversary successfully acquires the group key by exploiting a software vulnerability in a group member or client of the target system (described in Section 4.1.1).

$P_{VE\text{-}procedure\text{-}succ}$ is the probability that an adversary successfully acquires the group key by exploiting a procedural vulnerability within the target system (also described in Section 4.1.1). This includes all other types of vulnerabilities not related to software.

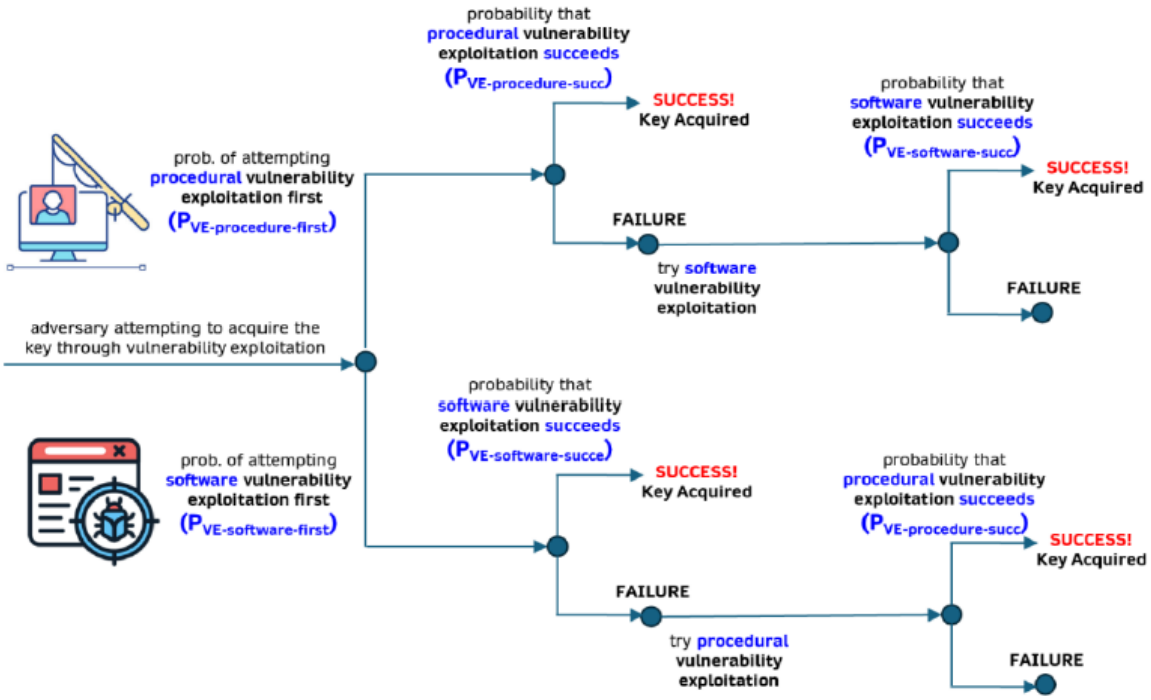


Figure 24: Possible strategies and probabilities of vulnerability exploitation steps

Now, in a real world PubSub OPC UA system, the values of these two probabilities are very closely tied to (i.e., controlled by) a number of parameters that are unique to each specific security group targeted by the DS attack as well as the features of the entire environment, including:

- The number of this group's members and their security posture - specifically the potential for exploitation of their software or procedural vulnerabilities that could lead to a direct compromise of the group's key from their runtime or at-rest memory – as per Strategy B from Section 4.1.1.
- The overall number of clients/devices in the system (other than the group members) and their security posture - specifically the potential for exploitation of their software or procedural vulnerabilities that could lead to a compromise of their private key / digital certificate, which then could lead to request and retrieval of the group's key from the SKS server – as per Strategy C from Section 4.1.1.

In this analysis, we simplify our assumptions by considering that Strategy C (indirect acquisition of the group key via client compromise followed by the key retrieval from SKS) is significantly less likely to be pursued or to be successful compared to Strategy B. Namely, strategy C is far more complex, requiring a multi-step process which can span several security layers,

making it more resource-intensive and time-consuming. These added steps also come with the drawback of being more likely to be detected by security monitoring systems, jeopardising the ultimate success of the entire attack. As a result, we will focus primarily on the impact of software and procedural vulnerabilities within the security group's own members when evaluating $P_{VE\text{-}software\text{-}succ}$ and $P_{VE\text{-}procedure\text{-}succ}$, rather than those of other clients.

In the next two subsections, we present our methodology for determining (i.e., estimating) the values of $P_{VE\text{-}software\text{-}succ}$ and $P_{VE\text{-}procedure\text{-}succ}$ in an actual system.

5.2.1 Software Vulnerabilities and the Probability of Key Compromise Through Exploitation: $P_{VE\text{-}software\text{-}succ}$

As stated in the preceding discussion, there are three aspects which must be considered when calculating the probability of a security-group's key compromise by means of software vulnerabilities existing in the group members ($P_{VE\text{-}software\text{-}succ}$): the number of devices in the group, the number of vulnerabilities (CVEs) associated with each individual device, and the CVE severity rating of each vulnerability (specifically their exploitability and impact). If any one of these values increase, so should the overall probability of key compromise. The extent of this increase depends on various factors, which will be discussed next. However, it is crucial to ensure that the final probability value of $P_{VE\text{-}software\text{-}succ}$ stays within the range [0, 1], maintaining a bounded probability metric.

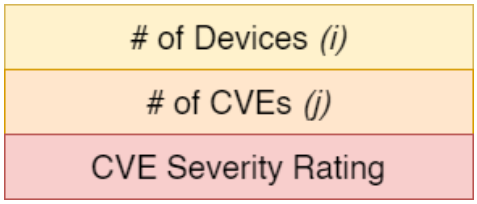


Figure 25: Values directly affecting the probability of compromise

Before we delve deeper into the details of our proposed methodology (i.e., algorithm) for calculation of $P_{VE\text{-}software\text{-}succ}$, we provide a big-picture overview of this algorithm in the below figure.

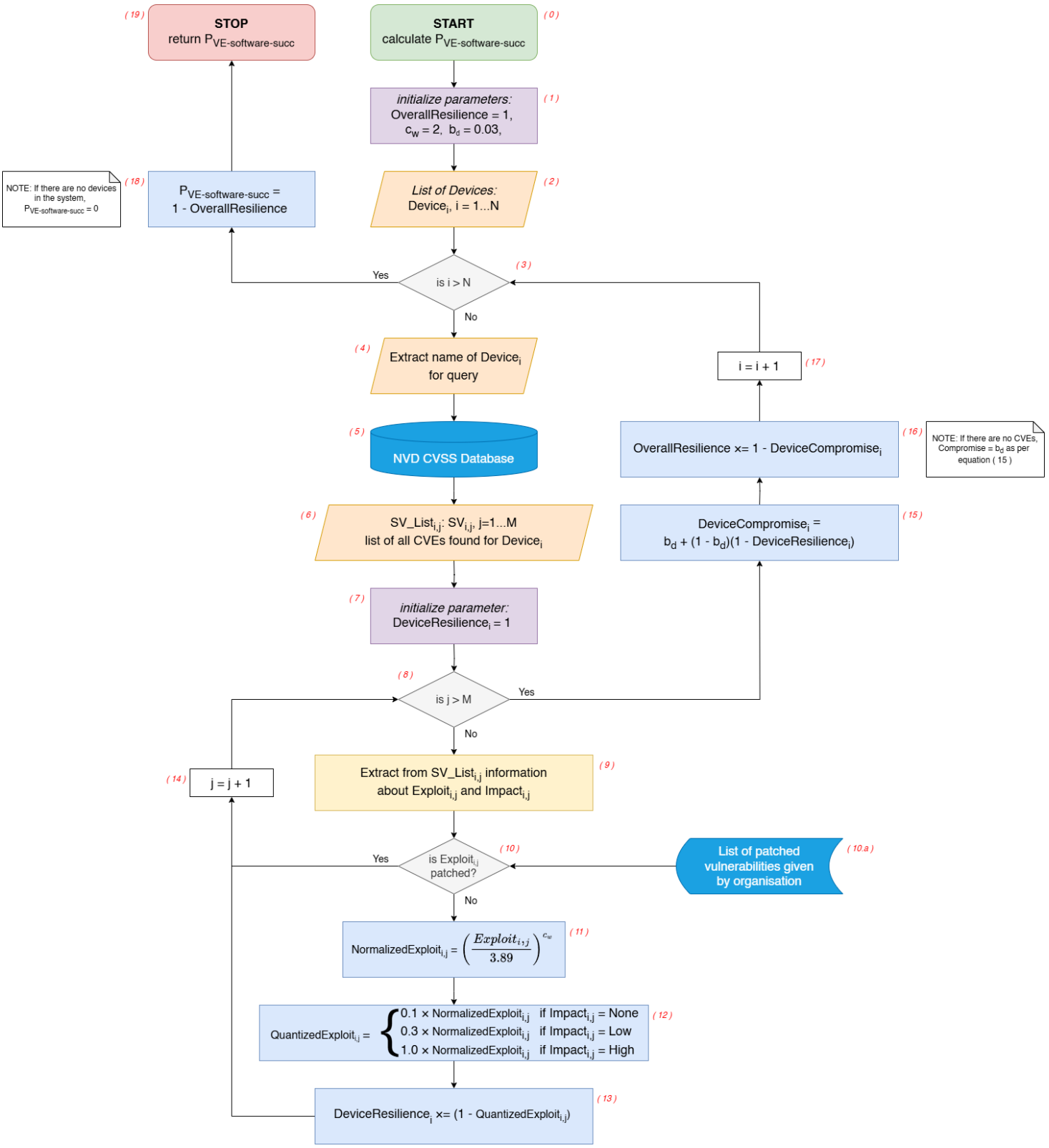


Figure 26: Flowchart for the $P_{VE\text{-}software\text{-}succ}$ calculation

The first set of steps in the proposed algorithm for calculation of $P_{VE\text{-}software\text{-}succ}$ (steps (2) to (6) in Figure 26) perform extraction of raw information about CVSS ratings [10] of all CVEs identified in each of the group members from the NVD CVSS database. In our case, we only deploy information from the CVSS base metric group – exploitability and impact scores (see Section 2.1).

The main 'building block' of the algorithm in Figure 26 is the processing of the Exploitability score ($Exploitability_{i,j}$) for a single CVE on a single device – steps (9) to (13). It should be noted here that the original CVSS Exploitability scores range from [0, 3.9], with a higher number representing a more vulnerable device (i.e., a greater ease by which the given vulnerability can be exploited). Since the final probability we are interested in calculating is required to be in the bounds of [0, 1], in step (11) of the proposed algorithm we normalize $Exploitability_{i,j}$ by transforming it to $NormalizedExploitability_{i,j}$, using equation (13).

It should be observed that in step (11) we apply an exponent (c_w) to the normalized exploitability score, with the aim to deemphasise low scoring vulnerabilities and their impact/contribution to the final $P_{VE\text{-}software\text{-}succ}$ value (see Figure 27). Namely, minor risk/exploitability vulnerabilities are far less likely to influence the adversary's ability to compromise a system, and therefore should have less significant impact on the overall $P_{VE\text{-}software\text{-}succ}$. In our framework we use $c_w = 2$, though other values could easily be used instead if justified by the circumstances of a particular environment.

$$NormalizedExploit_{i,j} = \left(\frac{Exploitability_{i,j}}{3.9}\right)^{c_w} \quad (13)$$

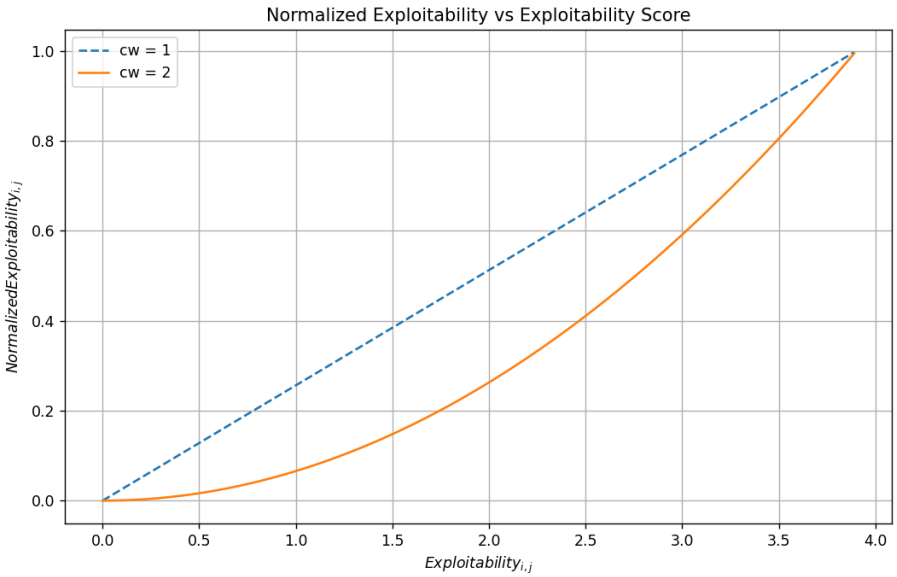


Figure 27: Exploitability normalized based on Equation 11 with two c_w values

In addition to providing an Exploitability score, the NVD CVSS database also provides three different Impact sub-scores for each enlisted vulnerability, which are intended to quantify the impact of successful vulnerability exploitation on: a) confidentiality, b) integrity, and c) availability, of system data. Each score can be "None", "Low", or "High". It should be noted that from the perspective of a DS attack, and for an adversary that aims to acquire the encryption/decryption key from the memory of a security-group member, only vulnerabilities/CVEs with a non-zero confidentiality impact score are of potential relevance. On the other hand, CVEs with non-zero integrity or availability scores are not of much use/relevance, as they cannot directly aid in extraction/compromise of the group's key information from the memory of the respective device. To account for this fact, we further modify the normalized exploitability score, as shown in step (12) in Figure 26, and equation (14).

$$QuantizedExploitability_{i,j} = \begin{cases} 0.1 \times NormalizedExploit_{i,j} \ \ if \ Confidentiality_{i,j} = None \\ 0.3 \times NormalizedExploit_{i,j} \ \ if \ Confidentiality_{i,j} = Low \\ 1.0 \times NormalizedExploit_{i,j} \ \ if \ Confidentiality_{i,j} = High \end{cases} \quad (14)$$

The next important step of our algorithm (step (13) in Figure 26) performs aggregation of quantized exploitability scores (for all CVEs) on one single device ($Device_i$) to determine the

overall probability of software-vulnerability based key compromise from this specific device. To do so, we deploy the concept of "complement probability" and make use of the "multiplication rule to determine the probability of independent events". Namely, complement probability states that the probability of at least one event occurring is the complement of the probability that no event occurs [35].

$$P(A) = 1 - P(A`) \tag{a}$$

In our case, A is the probability that at least one $Device_i$ CVE leading to the key compromise is successfully exploited, while A` is the probability that no such CVE is successfully exploited. Finding P(A`) requires the use of the well-known multiplication rule for independent events, which states that the probability of all (independent) events not occurring is the product of their individual probabilities of not occurring [36].

$$P(A`) = \prod_{i}^{n} P(A_i`) \tag{b}$$

Based on equation (b), we derive and deploy (15) to calculate the resilience of $Device_i$ ($DeviceResilience_i$), or how likely it is that the encryption/decryption key does NOT get compromised from this particular device.

$$DeviceResilience_i = \prod_{j=1}^{M} (1 - QuantizedExploiability_{i,j}) \tag{15}$$

Finally, to calculate the probability that the group key DOES get compromised through exploitation of software vulnerabilities on this particular device (with $DeviceCompromise_i$), we use a modified form of complimentary probability stated in equation (a) – as shown in (16) and step (15) in Figure 26. Note that our equation (16) begins with a baseline parameter[3] ($b_d = 0.03$), which is introduced to allow devices with no known CVEs (i.e., with $DeviceResilience_i = 1$) to still

[3] A baseline value of 0.03 was chosen as we found it to be the best fit for our test-cases, however, the value can be adjusted for different environments and scenarios.

marginally impact the overall $P_{VE\text{-}software\text{-}succ}$. In other words, our motivation for the use of this parameter is the fact that even devices with no CVEs currently reported in NVD CVSS databases could still potentially 'harbour' zero-day vulnerabilities with non-negligible exploitability and confidentiality-impact scores, and thus should (even if just marginally) contribute to the overall $P_{VE\text{-}software\text{-}succ}$.

$$DeviceCompromise_i = b_d + (1 - b_d) \times DeviceResilience_i \quad (16)$$

Once the probability of key compromise for/on each individual $Device_i$ (i=1,..,N) is determined, the probability of key compromise across the entire security group ($P_{VE\text{-}software\text{-}succ}$) is calculated by:

a) finding the overall group 'resilience' to key compromise using the product of individual complementary probabilities – see equation (17) and step (16) in Figure 26, and then

b) finding the complement of the overall group 'resilience' to key compromise – see equation (18) and step (18) in Figure 26.

$$OverallResilience = \prod_{i=1}^{N} (1 - DeviceCompromise_i) \quad (17)$$

$$P_{VE-software-succ} = 1 - OverallResilience \quad (18)$$

5.2.2 Procedural Vulnerabilities and the Probability of Key Compromise Through Exploitation: $P_{VE\text{-}procedure\text{-}succ}$

Unlike probability $P_{VE\text{-}software\text{-}succ}$ discussed in Section 5.2.1, which depends on the number and type of devices forming a security group targeted by a DS attack, the probability of a security group's key compromise through exploitation of procedural vulnerabilities ($P_{VE\text{-}procedure\text{-}succ}$) is far more likely to be similar across different devices / security groups of the same target system. This similarity arises because procedural vulnerabilities are primarily influenced by the overarching security policies and guidelines established by the organization. These policies affect how procedures are implemented and followed across the board. Therefore, $P_{VE\text{-}procedure\text{-}succ}$ can be

treated as a system-wide variable and ideally would be estimated by the system administrator based on their understanding of the strength of security policies deployed across the organization.

Figure 28 depicts our proposed qualitative approach and subsequent quantitative mapping to obtain the information about $P_{VE\text{-}procedure\text{-}succ}$ using the system operator's feedback.

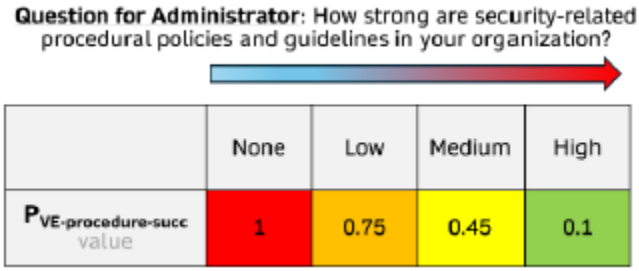


Figure 28: Qualitative-risk approach to obtaining $P_{VE\text{-}procedure\text{-}succ}$ through a system/security administrator

As Figure 28 indicates, in the worst-case scenario (i.e., when there are no security related policies in place), $P_{VE\text{-}procedure\text{-}succ}$ is determined to be 100%. On the other side of the spectrum, if an operator states that their organization uses robust and comprehensive security policies, $P_{VE\text{-}procedure\text{-}succ}$ is set to 0.1, which reflects a slim but still not entirely negligible possibility of human error leading to a potential compromise of the security group's encryption key. A more detailed explanation concerning the qualitative procedure for determining $P_{VE\text{-}procedure\text{-}succ}$ can be found in Appendix C.

5.2.3 Final Look at $P_{SG\text{-}KC\text{-}succ}$

As you may recall, at the beginning of this chapter we identified $P_{SG\text{-}KC\text{-}succ}$ as an important factor for calculation of $R_{SG\text{-}KC}$, which would ultimately allow us to determine the optimal $T_{CP\text{-}SG}$. In this section (Section 5.2), we have further identified $P_{VE\text{-}software\text{-}succ}$ and $P_{VE\text{-}procedure\text{-}succ}$ as two key factors (i.e., probabilities) impacting $P_{SG\text{-}KC\text{-}succ}$ as per equation (12) – which is also shown below. Subsequently, in Sections 5.2.1 and 5.2.2, we have presented our novel algorithm for explicit calculation of $P_{VE\text{-}software\text{-}succ}$ as well as a qualitative procedure for estimation of $P_{VE\text{-}procedure\text{-}succ}$.

$$P_{SG-KC-succ} = 1 - (1 - P_{VE-software-succ})(1 - P_{VE-procedure-succ}) \quad (19)$$

5.3 Methodology for Calculation of Adverse Impact AI_{SG-KC}

With the probability portion of equation (9) addressed, the final step to complete the calculation of R_{SG-KC} – and, ultimately, the calculation of the optimal security group cryptoperiod T_{CP-SG} as per equation (2) – is to determine the Adverse Impact (AI_{SG-KC}) of a DS attack. As explained in Section 5.1, this value represents the negative impact on the organisation in the event of a successful DS attack on one of its security groups (i.e., successful compromise of this group's encryption key).

From the formulation of a DS attack (Chapter 4), the extent of AI_{SG-KC} can be approximated as a product of two important factors pertaining to the data of the targeted security group: data's **I**nformation **R**ate (IR_{SG})[4] and data's **F**unctional **I**mportance (FI_{SG}). In the proceeding subsections, we discuss each of these factors in more detail.

$$AI_{SG-KC} = IR_{SG} \times FI_{SG} \quad (20)$$

5.3.1 Information Rate of Targeted Security Group Data (IR_{SG})

In more precise terms, the information rate of the targeted security group (IR_{SG}) represents the overall amount of 'information of value' that is generated by the group members (as a collective) per time unit and appears on (i.e., is shared through) the group's encrypted channel. During an ongoing DS attack – where the adversary is assumed to be able to collect and exfiltrate all the encrypted group's data - IR_{SG} will also represent the amount of potentially[5] 'lost data' per time unit. In other words, the amount of 'asset' or 'value' that the organization potentially loses per time unit.

[4] Note here, by focusing on 'Information Rate' (which is measured per time unit), we effectively derive the value of AI_{SG-KC} 'per time unit'.

[5] The word 'potentially' is used here because for security group data to be actually compromised, the adversary should be able not only to collect and exfiltrate it, but also to successfully acquire its (de)cryption key.

Now, IR_{SG} itself will depend on two factors: the number of group publishers (i.e., number of group members that actively generate new data/information), and the average information rate per group publisher. A security group with many publishers, and where each publisher generates new information very frequently, will exhibit an overall higher IR. At the same, a group with only a few publishers that generate new information very infrequently will exhibit a very low IR. To simplify the process of determining IR_{SG} in a real-world system, we propose the below qualitative approach[6]. This approach requires only two parameters to be input by the system administrator, or acquired from the system by some other more automated means:

a) the number of security group publishers – $N_{SG\text{-}publishers}$, and

b) the average information rate per publisher – $IR_{per\text{-}SG\text{-}publisher}$.

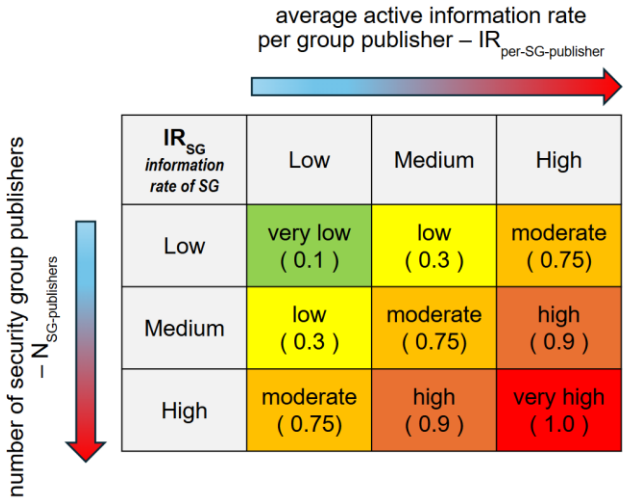


Figure 29: Qualitative approach to determining traffic intensity of a security group (IR_{SG})

The mapping technique in Figure 29 categorizes each parameter into three levels: Low, Medium, and High. By intersecting these levels on the chart, we obtain a qualitative measure of IR_{SG}. For example, a group with a high number of publishers and a high transmission rate per publisher results in very high IR_{SG} = 1, while a group with few publishers and a low transmission rate result in very low IR_{SG} = 0.1. More information about how each mapping was selected can be found in Appendix C.

[6] A qualitative approach is chosen as it is better suited for practical implementation, allowing system operators to estimate IR_{SG} based on observable characteristics rather than requiring precise quantitative measurements, which may be difficult to obtain in complex ICS environments.

5.3.2 Functional Importance of Targeted Security Group Data (FI_{SG})

Another parameter affecting the value of $AI_{SG\text{-}KC}$ is the Functional Importance of group data (FI_{SG}). Unlike information rate, which is fairly trivial to assess, FI_{SG} involves fewer tangible outcomes. This requires an evaluation of the potential damage resulting from data compromise, while keeping in mind that any application-level data sent between devices in an ICS environment can impact several different aspects of this organization's performance (i.e., functioning) at the same time. In practical terms, this parameter is an indicator of the compromised data's importance and the extent at which the compromise would impact different functional objectives the target organization.

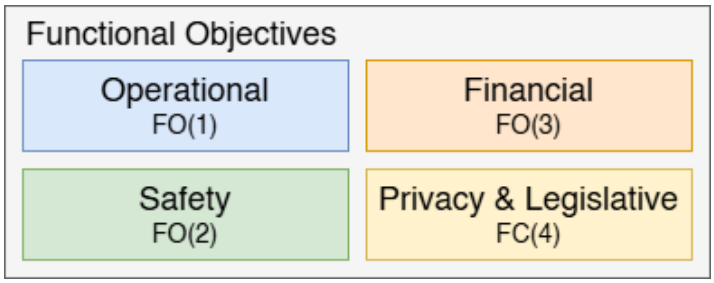


Figure 30: Functional Categories represented in FI_{SG}

Definition of Functional Objectives: Based on the methodologies presented in [37], [38], [39] and [40], **F**unctional **O**bjectives (FOs) of most ICSs can be grouped into the following 4 categories (with the actual importance of each objective varying from one organization to another).

Operational Objectives (FO(1)): Maintain the intended day-to-day operational performance of the system/organization, and prevent any potential misuse, malicious manipulation, or complete loss of system function(s).

Safety Objectives (FO(2)): Ensure the functional safety of the operation, as well as the health and safety of: 1) organization's employees and customers, b) general public, c) environment.

Financial Objectives (FO(3)): Ensure that revenue goals are met while also preventing/minimizing possible losses and costs such as: a) commercial losses due to impaired system function, c) future revenue losses due to reputational damages and/or loss of intellectual property, c) costs of repairing the system and/or replacing damaged components, etc.

Privacy and Legislative Objectives (FO(4)): Achieve and support organization's compliance with regulatory and legislative guidelines, including those pertaining to user privacy and environmental protection.

It should also be pointed out that in certain situations failure to support one of the Functional Objectives can have impact on one or more of other Functional Objectives. For example, failure to comply with government regulations (i.e., privacy and legislative objectives) could result in a range of ramifications pertaining to financial objectives, including monetary fines, loss of market share, reputation loss, etc. Through this interdependency, attacks/incidents that effect multiple Functional Objectives could have cumulative effect in terms of their ultimate attack impact. (Several of the aforementioned functional categories are broad in what they encompass, which can necessitate a more granular approach to assess their impact, as explained in Appendix C.)

Analytically, as shown below, we define FISG as the average functional importance of the security group's data across all four above mentioned Functional Objectives:

$$\mathrm{FI}_{\mathrm{SG}} = \frac{\sum_{j=1}^{4} \mathrm{FO}_{criticality}(j) \cdot \mathrm{FI}_{\mathrm{SG}}(j)}{4} \tag{21}$$

where:

$FO_{criticality}(j)$ is a weight factor that captures the general criticality of FO(j) for the organization's overall performance and business 'bottom line', assuming each $FO_{criticality}(j) \in [0, 1]$ and 1 representing highest possible criticality.

$FI_{SG}(j)$ is the functional importance of the security-group's data for FO(j), j=1, 2, 3, 4. The possible values of $FI_{SG}(j) \in [0, 1]$, with 1 marking maximum possible importance.

Both values, $FI_{SG}(j)$ and $FO_{criticality}(j)$, should ideally be decided/provided by the stakeholder (i.e., an employee with in-depth knowledge of the operation processes and functional priorities of the organization).

To better understand the significance and workings of equation (21), let us consider the following scenario. The organization managing the security group (for which the optimal cryptoperiod needs to be determined) in some aspects of its operation deals with personal customer information, which is protected by the legislature in most countries. Consequently, the organization rates the Privacy and Legislative objectives as High ($FO_{criticality}(4)=1$). However, the data specifically exchanged between the members of the given security group is entirely unrelated to customers (i.e., customer information), suggesting $FI_{SG}(4)=0$. As a result, the Functional

Importance of the group's data and the impact of its compromise on FO(4) would be 0 and could be ignored. Though, the impact of this same data on other Functional Objectives could be non-negligible and should be separately evaluated.

Our proposed qualitative approach to assigning specific values to $FI_{SG}(j)$ and $FO_{criticality}(j)$ (j=1,2,3,4) respectively are illustrated in Figure 31.

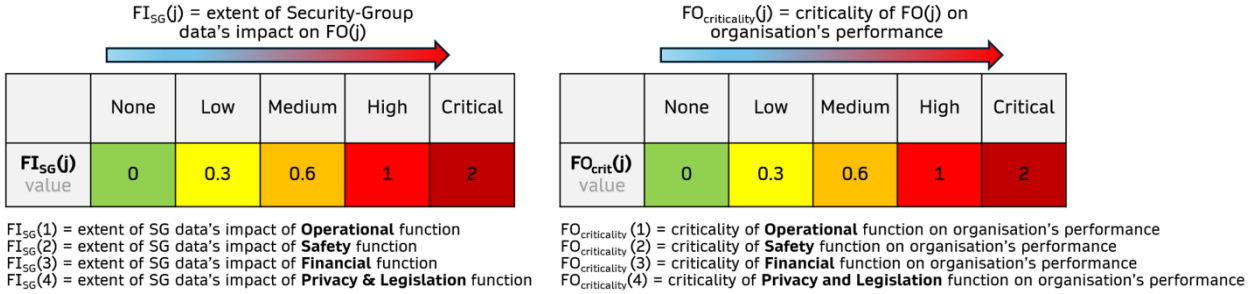


Figure 31: Qualitative approach to determining the importance and extent of Function Categories (FC)

One final note is the inclusion of a "Critical" category (in Figure 31) which exceeds the expected value range of [0, 1]. In some instances, a failure in a single impact category can cause catastrophic damages for an organization. This 'Critical' category addresses scenarios where the impact is so severe that it cannot be adequately represented within the standard range. Normally, an impact is confined within its category, meaning that with four categories and FI_{SG} being the average, a single category can only contribute up to 0.25 to the total impact. However, when a category is marked as 'Critical' (both $FI_{SG}(j)$ and $FO_{criticality}(j)$ for that category are critical), the overall impact becomes 1 immediately. It is crucial to note that FI_{SG} itself remains bounded between 0 and 1, ensuring that even if multiple categories are critical, the final value of FI_{SG} cannot exceed 1. More information about how each value was chosen can be found in Appendix C.

Once IR_{SG} and FI_{SG} are determined, the overall impact can be calculated as:

$$AI_{SG-KC} = IR_{SG} \times FI_{SG} \tag{22}$$

5.3.3 Adjustment of the Information Rate Parameter

$$R_{SG-KC} = P_{SG-KC-succ} \times IR_{SG} \times FI_{SG} \quad (23)$$

The current expression for $R_{SG\text{-}KC}$ (simplified in equation (23)), derived from a rigorous analytical model, initially assumes equal contribution from all parameters. However, our evaluation under specific edge cases indicate that an adjustment is necessary for the contribution of the information rate (IR_{SG}) parameter. The issue arises when the criticality of the transmitted data is underestimated, ultimately resulting in underemphasized risk. In some environments, such scenarios could be particularly dangerous, as even minimal exposure/compromise of critical data could have severe consequences.

As an illustration of the above, consider an environment where it is determined that an adversary has very high chances of successfully compromising some critically important in-transit data ($P_{SG\text{-}KC\text{-}succ}$ = 1, FI_{SG} = 1), but a very low amount of this data is being transmitted at a time (IR_{SG} = 0.1). Plugging the values into (23), this results in:

$$R_{SG-KC} = 1 \times 0.1 \times 1 = 0.1$$

Although it is almost guaranteed that this critical data will be compromised (from $P_{SG\text{-}KC\text{-}succ}$), $R_{SG\text{-}KC}$ is returning a value of only 0.1 – i.e., a value that does not accurately represent the risk associated with the given scenario. This problem can be mitigated by applying a weight function for IR_{SG} that depends on the value of FI_{SG}, as shown in equation (24). Based on Figure 32, when data is highly critical, even small amounts being compromised can result in significant losses, making the rate of data transfer less significant relative to the actual data sensitivity (i.e., impact). On the other hand, as the data becomes less impactful, its actual transfer rate starts gaining more importance.

$$IR_{SG}{}^{w}, w = 1 - FI_{SG} \quad (24)$$

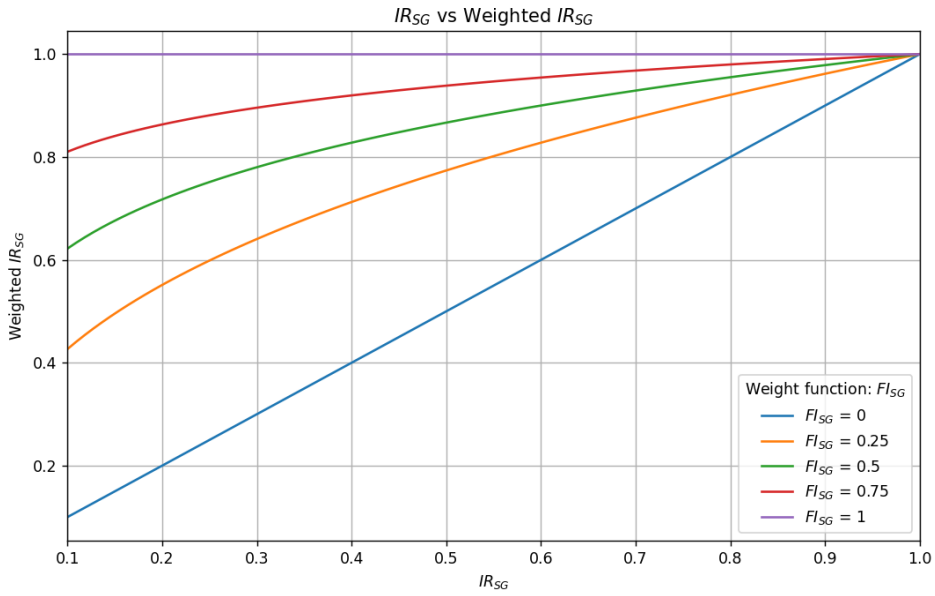


Figure 32: The effect of the weight function on FI_{SG}

5.3.4 Final Look at AI_{SG}

The second half of the risk calculation ($R_{SG\text{-}KC}$) relies on adequate understanding of the impact parameter - AI_{SG}. This section (Section 5.3) expanded upon this concept, defining IR_{SG} and FI_{SG} as two key attributes which influence the impact. Section 5.3.1 provided a quantitative approach for estimating IR_{SG}. Section 5.3.2 provided a more involved quantitative approach for estimating FI_{SG}, including the defining of four Functional Objectives (FO) in an organization. Lastly, a modification is applied to the IR of the formula in Section 5.3.3, covering for specific edge cases which may be encountered in practice and produce undesirable results. The final formula for AI_{SG} based on (20) and (24) can be found below.

$$AI_{SG-KC} = {IR_{SG}}^{w} \times FI_{SG}, w = 1 - FI_{SG} \tag{25}$$

5.4 Completed Expression & Conclusion

This section provides a summary of each step in finding the optimal cryptoperiod ($T_{CP\text{-}SG}$) for a security group.

Firstly, we begin by assigning a lower ($T_{CP\text{-}min}$) and upper ($T_{CP\text{-}max}$) bound to the

cryptoperiod duration using an exponential scale. The value is reliant on the overall risk of the group ($R_{SG\text{-}KC}$) (Section 5.1.1):

$$T_{CP-SG} = T_{CP-max} \times \left(\frac{T_{CP-min}}{T_{CP-max}}\right)^{R_{SG-KC}} \quad (26)$$

$R_{SG\text{-}KC}$ is calculated based on the probability of a successful key compromise ($P_{SG\text{-}KC\text{-}succ}$) and adverse impact if/once the key is compromised ($AI_{SG\text{-}KC}$) (Section 5.1.2):

$$R_{SG-KC} = P_{SG-KC-succ} \times AI_{SG-KC} \quad (27)$$

The probability of key compromise has two parameters: 1) the probability of software compromise ($P_{VE\text{-}software\text{-}succ}$), which is a value unique to each member in the security group, and 2) procedural policy strength ($P_{VE\text{-}procedure\text{-}succ}$), which is a global value for the entire security group (Section 5.2.3):

$$P_{SG-KC-succ} = 1 - (1 - P_{VE-software-succ})(1 - P_{VE-procedure-succ}) \quad (28)$$

Likewise, the adverse impact also has two parameters: 1) the information rate in a network (IR_{SG}), which is based on the number of publishers in a network and average transmission rate, and 2) functional impact (FI_{SG}), which is based on the functional importance of a security group's data and functional objective criticality for an organisation's overall performance (Section 5.3):

$$AI_{SG-KC} = IR_{SG} \times FI_{SG} \quad (29)$$

Lastly, we modify IR_{SG} by assigning it a weight function dependent on FI_{SG} to ensure that specific edge cases do not distort the cryptoperiod formula (Section 5.3.4).

$$IR_{SG}{}^{w}, w = 1 - FI_{SG} \quad (30)$$

As such, the final formula for $R_{SG\text{-}KC}$ combining (27), (28), (29), and (30) is:

$$R_{SG-KC} = 1 - \left(1 - P_{VE-procedure-succ}\right) \times \left(1 - P_{VE-software-succ}\right) \times IR_{SG}{}^{w} \times FI_{SG}$$
$$w = 1 - FI_{SG} \tag{31}$$

Following an in-depth analysis of security standards as well as attack vectors related to DS in Chapter 4, this chapter consolidates the findings into a single, succinct formula. Each variable in the formula was carefully selected and weighted to reflect the key factors related to risk analysis and provides an adaptable approach to calculating optimal cryptoperiods which can be tailored to specific ICS environments. The following chapter will discuss the implementation of the given formula, detailing the steps taken to integrate it within our software tool, ARC-C.

6 Overview of ARC-C Software Tool

With the cryptoperiod formula established in Chapter 5, the next step in our process is to design a software tool that integrates the cryptoperiod calculation logic (i.e., formula (30) from Chapter 5) while also having a user-friendly interface. We name this tool **A**utomated **R**isk-Based **C**ryptoperiod **C**alculator (ARC-C). Namely, a well-designed interface is critical to bridging the gap between the complex calculations performed in the back-end of ARC-C and the operator's ability to effectively configure and understand the results. As there are several unique parameters for an operator to consider, we must ensure that our user interface (UI) is both *intuitive* and *responsive*. This includes clearly organizing input fields, providing real-time feedback on parameter adjustments, and adding helpful tooltips/documentation to guide users through more complex implementations (i.e., collecting device information for the $P_{VE\text{-}software\text{-}succ}$ calculation).

In addition to the front-facing (i.e., interface) objectives in the design of ARC-C, we also had to account for some special requirements of its internal logic, including the integration of the NVD API (briefly described in Section 2.1) to facilitate the imports of relevant software-vulnerability information. This chapter provides a high-level analysis of the functional and design decisions that guided the creation of ARC-C, with a more detailed exploration of the code available in Appendix D.

6.1 ARC-C Overview

The successful implementation of ARC-C requires not only careful consideration of user experience (UX) principles, but also the selection of a robust development framework to bring the interface to life. To achieve this, we rely on the PySide6 library - the official set of Python bindings for Qt libraries[7] - which offers powerful tools for developing sophisticated and responsive GUIs [41]. This library allows us to incorporate a variety of customizable UI elements to create a user-friendly interface.

Building on the capabilities of PySide6, ARC-C is designed so that an operator can see all key parameters on screen at the same time, removing the need to navigate through cumbersome submenus. While this may initially appear overwhelming, the functions and parameters are aligned in a way that ensures a logical and intuitive flow. As seen in Figure 33, each aspect of the

[7] Qt is a cross-platform application development framework for desktop, embedded and mobile. [69]

cryptoperiod calculation is self-contained. To assist with the overall understandability, each UI element has a tooltip. If an operator hovers over an element, detailed information regarding the element's purpose and functionality is provided. For more complex actions, a separate information panel can be accessed as well.

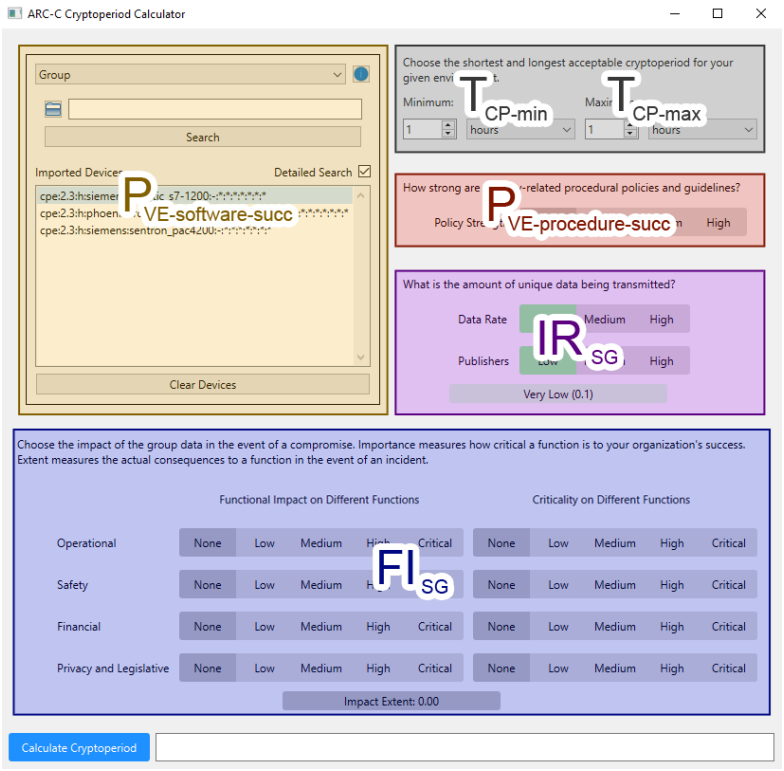


Figure 33: ARC-C User Interface labelled

The UI can be separated into two core functions: importing devices, which will be discussed in Section 6.2 (the following section), and setting environmental parameters, covered in Section 6.3.

6.2 Importing Devices Panel

Importing devices is required for the calculation of $P_{VE\text{-}software\text{-}succ}$ (defined in Section 5.2.1), so ARC-C provides users with the ability to do so through the Import Devices panel. While importing devices, the ARC-C tool interfaces with the NVD API [42] and returns the necessary data to assist in said calculations. The import function itself can be broken into 2 components, as

seen in Figure 34. The topmost component is used to import the devices themselves, while the bottom displays the devices which have already been successfully imported.

Figure 34: Importing Devices UI submenu

Before discussing the mechanics of importing devices, we must first look at the overarching logic which enables the importing to work. When a user is asked to input a device, the typical format one would expect would be the product name (i.e., Simatic S7-1200). However, the NVD API database parses data using Common Platform Enumeration (CPE) [42]. CPE is a naming scheme for IT and OT systems, software, and packages. This means we must first process the user-inputted data to match with a CPE listed in NVDs database. There are two ways this can be done; automatically or manually.

- **Automatic:** Our assumption is that most operators will choose to automatically query the database, especially in instances when there are many devices in a security group. By default, searching for a device will return the first non-firmware result provided by NVD. This typically produces the desired result; however, it can occasionally return incorrect devices (e.g., wrong versions or variations).
- **Manual:** For more advanced users, a “Detailed Search” box can be checked for manual querying, located to at the bottom-right of the search button in the Import Devices panel

(Figure 34). If checked, a pop-up window appears during a search (Figure 35) where every queried result is listed. Here, the user can select the exact device they want for calculation.

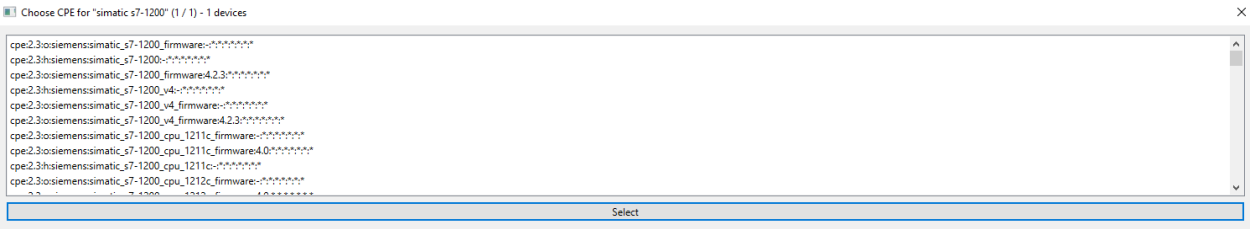


Figure 35: Detailed Search pop-up window for "SIMATIC S7-1200"

6.2.1 Methods of Importing Devices

The ARC-C interface allows three methods of importing devices: individual, group, and manual. The dropdown menu at the top of the UI allows the user to switch between each of the importing methods (Figure 36). The user can request help by clicking the blue information icon to the right of the dropdown which will display an information pop-up.

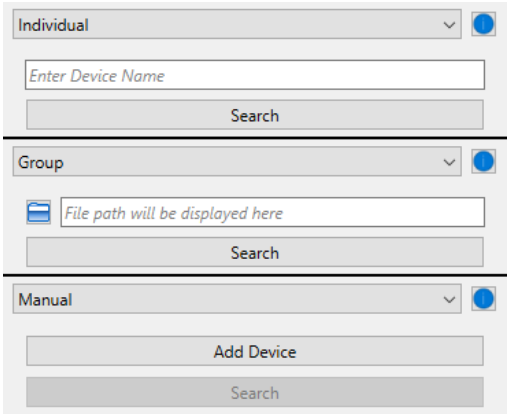


Figure 36: The three modes of device importing

For individual importing, the user is prompted for a single device name. Once inputted, the user can search for the device. If a match is found, the device is automatically added to the "Imported Devices" list. If a match is not found, the user is asked to try again using another name or to provide a more detailed search string.

With group importing, the user is first required to preprocess device data by creating a text file containing a list of all devices, with each device name provided on separate lines. The user can then import the file by clicking the open folder icon. Once the file is imported, the program

iteratively searches the NVD CPE database for each device from the file. At the end of the final query, the devices which were found are added to the list while any devices which were not found are displayed to the user with an error. When conducting a detailed group search, the user will be asked to select a specific CPE for each individual device, one at a time.

Lastly, the user can manually create devices. The NVD API has experienced issues in which it becomes slow to use or in some cases, completely unresponsive. As importing devices is crucial to the final formula, we have included an option to manually create CPEs and assign CVEs to them. As this process does not require the API, the search button is unclickable. Instead, devices are automatically added to the list once they have been created. Adding a device using this method brings the user to a pop-up window (Figure 37). There, they can create a CPE and assign CVEs with an associated Exploitability score and Impact metric.

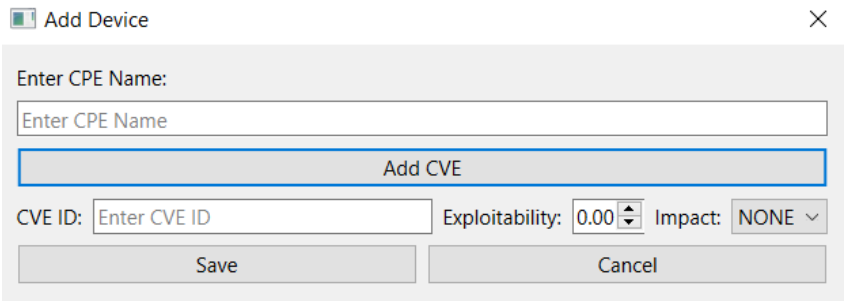


Figure 37: Adding a device when manually importing devices

6.3 Environmental Parameters

The remaining ARC-C UI elements are parameters directly adjusted by the user. Reading the ARC-C UI top to bottom components (Figure 33), the first variables to be set are $T_{CP\text{-}min}$ and $T_{CP\text{-}max}$ (discussed in Section 5.1.1). Setting these values uses a combination of spinboxes and comboboxes[8], seen in Figure 38. The timescale an operator may want to set can vary and therefore requires flexible input options. The spinboxes allow precise numerical entry while the combo box provides a selection between hours, days, and months.

[8] Spinboxes allow users to select a number from a range by clicking arrows to increase or decrease the value, while comboboxes let users choose from a dropdown list of options.

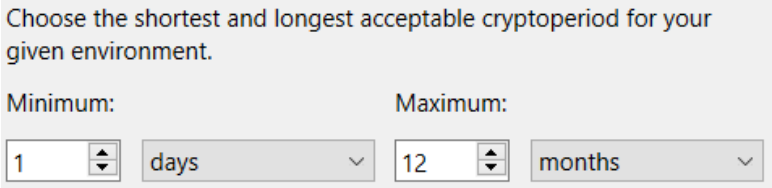


Figure 38: $T_{CP\text{-}min}$ and $T_{CP\text{-}max}$ UI elements

$P_{VE\text{-}procedure\text{-}succ}$ (Figure 39 and discussed in Section 5.2.2) utilises a simple segmented control input. The user can select one of four values, with the colour of the button changing with respect to the severity level as a visual cue.

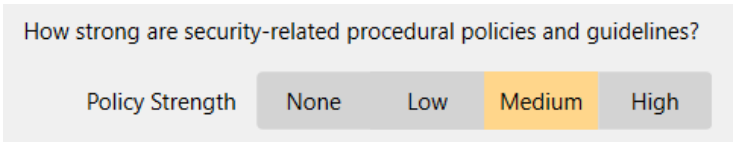


Figure 39: $P_{VE\text{-}proc\text{-}succ}$ UI element with "Medium" selected

The IR_{SG} impact calculation is slightly more complex than the previous segmented controls. As mentioned in Section 5.3.1, two separate input values are mapped to a single output value: the *average rate of unique data sent by publishers* and the *number of publishers*. Under the two segmented control inputs, the mapped output value is displayed (Figure 40). This value gets automatically calculated and updated when the data rate and publisher buttons are changed.

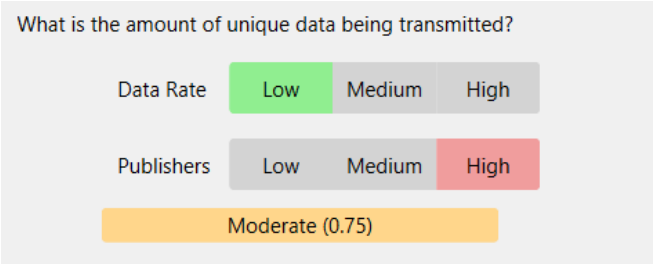


Figure 40: IR_{SG} UI element with "Low" and "High" selected

The last and largest section of the UI is dedicated to $FI_{SG\text{-}KC}$ (Figure 41). Recalling from Section 5.3.2, $FI_{SG\text{-}KC}$ is calculated by multiplying the 'Functional Impact' and 'Criticality' for each Functional Objective (FO). There are four categories in total, represented by the labels on the

lefthand side of the UI. The first set of segmented control buttons are dedicated to Functional Impact and the second set are dedicated to Criticality. The overall Impact is displayed at the bottom. While not used for the purpose of this thesis, ARC-C also has built-in functionality to support subcategories, represented by the arrows next to the 'Financial' and 'Privacy and Legislative' categories. For more information, refer to Appendix D.

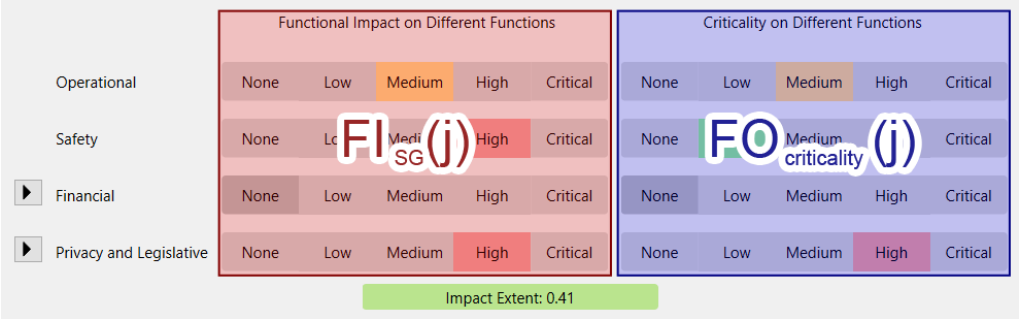


Figure 41: FI_{SG} UI element with various severities selected

6.4 Conclusion

In this chapter, we have outlined the development of our ARC-C tool, highlighting its features and functionality. Further details of ARC-C and its code are provided in Appendix D. The next chapter will showcase the actual use/application of the ARC-C tool, which ultimately demonstrates its capabilities and practical relevance through a series of test cases.

7 Use Cases of the ARC-C Framework

With the cryptoperiod formula defined and the ARC-C tool created (as described in Chapters 5 and 6), in this chapter we present a series of evaluations that are aimed to test and demonstrate the tool's effectiveness across different ICS environments and operational conditions resembling those found in the real-world. Specifically, in these evaluations, by envisioning two different real-world ICSs (a water treatment plant and an energy storage system), and then in each ICS by considering several different test-cases (e.g., with varying levels of security preparedness), we deploy our ARC-C to calculate the optimal cryptoperiod for different security groups operating in those environments. Since this is the first application of our cryptoperiod formula in such environments, it means there is no established benchmark or set of values against which to compare our results. Instead, we rely on general theoretical expectations derived from the NIST standards, as well as a practical common-sense judgment, to evaluate the tool's actual effectiveness.

7.1 Introduction

Real-world ICSs can vary widely in their scale, complexity, and configuration, depending on the respective industry and operational requirements - with some systems consisting mostly of legacy equipment, while others integrate legacy infrastructure with modern functional components. This diversity is further expanded when we consider that even similar general components and configurations may, in fact, deploy different hardware, firmware, or software versions.

To explain our choice of experimental environments for testing of the ARC-C tool and its underlying methodology, we draw on the PURDUE model as a reference framework for representing the hierarchical structure of ICS networks (as discussed in Section 3.3.1). Recall from Figure 14, the PURDUE model consists of five layers, with Layers 0 through 3 specifically covering the ICS (OT) domain, and Layers 4 and 5 being in the IT domain. In our experimental analysis, and for simplicity of the discussion, we will be focusing only on the ICS/OT segments of the target network(s), which means we will be creating example environments and applying our cryptoperiod formula only to their respective Layers 0 to 3 devices. Note, however, it is possible to easily extend the use of ARC-C to incorporate all (or any select subgroups) of devices from the IT layers.

To test the capabilities of the ARC-C tool, two general example environments are considered: a *Water Treatment Facility* and an *Energy Storage System*. These environments are characterized by distinct architectures and device configurations, allowing the ARC-C tool (and the underlaying framework) to be evaluated for its flexibility and effectiveness across different ICS setups. The specific test-cases in these two environments are structured as follows:

Water Treatment Facility	
Test-Case 1	System with Strong Security: In this this test-case scenario, CVEs with a severity score of Medium or above (according to CVSS) have been patched and security groups are created for each function. Multiple OPC UA security group data rates and security policies are considered/tested across three small security groups.
Test-Case 2	System with Weak Security: In this this test-case scenario, only CVEs with a Critical severity score (according to CVSS) have been patched, and one single OPC UA security group which encompasses all devices is created. Multiple security group data rates and security policies are considered/tested across this (one large) security group.
Energy Storage System	
Test-Case 3	System with Moderate Security: In this this test-case scenario, CVEs with a severity score of High or above (according to CVSS) have been patched. Multiple security group data rates and security policies are considered/tested across one large and one small security group.

Table 2: Structure and explanation of upcoming test-cases

The experiments pertaining to our hypothetical water treatment plant (Test-Case 1 and 2) are presented in Section 7.2, while the experiments pertaining to our hypothetical energy storage system (Test-Case 3) are presented in Section 7.3.

7.2 Evaluation of ARC-C in a Water Treatment Facility

7.2.1 Water Treatment Facility: Background

The first ICS environment explored in our analysis is a water treatment plant (see Figure 42). We have chosen to study this particular type of ICS environment due to the real-world

prevalence and critical nature of water treatment systems, as well as the fact that they are a common target of APT attacks (including DS attacks) [6], [43]. Given that these facilities are responsible for providing clean and safe drinking water, it is critically important to ensure that they operate in a reliable and safe manner.

Similar to other industrial environments, the use of remote monitoring and control in water treatment systems gives operators the ability to closely and continuously oversee various aspects and/or parameters of the water treatment operation. This can range from water flow rates, chemical dosing, pH levels, and filtration process. The ability to conduct centralized monitoring and control is particularly invaluable in large water treatment facilities that span large geographic areas.

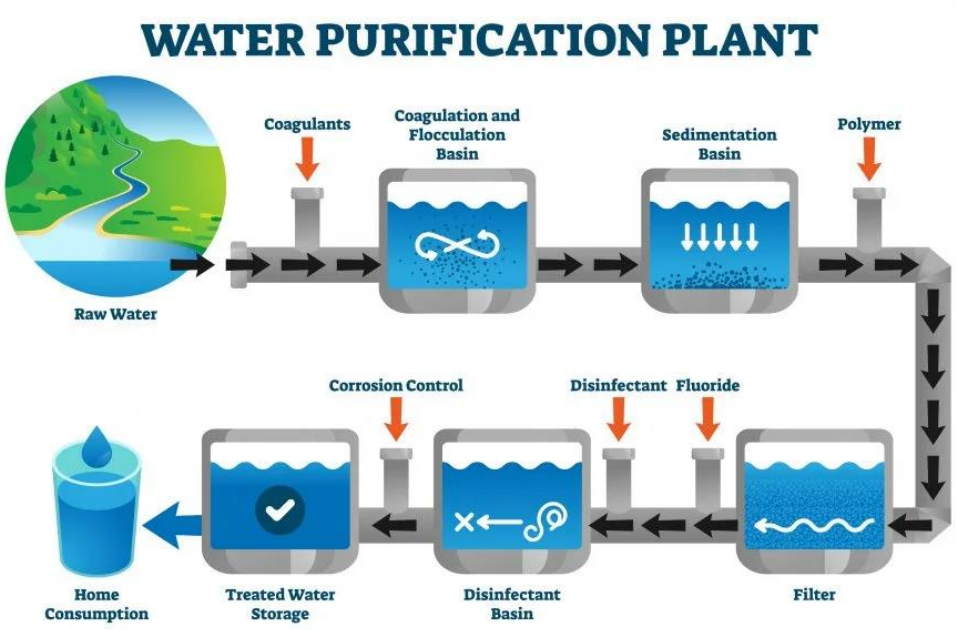


Figure 42: Wastewater treatment process [44]

By applying our ARC-C tool to a hypothetical (but realistic) water treatment plant environment, we are attempting to assess the tool's ability to determine cryptoperiods that strike the right balance between the system's security and its operational efficiency.

7.2.2 Water Treatment Facility: Assumed System Structure and Features

To create an environment reflective of a real-world water treatment facility, we follow the recommendations provided in [45] and [46]. According to these two works, and in the context of

the PURDUE model, the general functionality of the devices placed in layers 0 – 3 of a water treatment plant can be depicted as follows (see below Table):

Layer 0	Field level with physical devices that provide the distribution and delivery of water services, such as pumps and valves.
Layer 1	Process level with controllers that autonomously manage and monitor devices and/or act based on directives from supervisory systems.
Layer 2	Supervisory control level with systems that directly monitor the processes at lower levels (e.g., water delivery and distribution), and includes devices such as Human-Machine Interfaces (HMIs).
Layer 3	Site level with a plethora of computing devices performing different higher-level operations, such as production reporting and terminal services.

Table 3: PURDUE layers 0-3 for a Water Treatment Facility

Paper [46] additionally offers explicit suggestions on how to set up (i.e., organize) the devices in a water treatment environment. Specifically, it recommends that field (Layer 0) and process (Layer 1) devices generally be segregated from one another – except for the connections between PLCs and subset of sensors/actuators that they directly control, as illustrated in Figure 43. As for paper [45], in addition to providing a detailed overview of a water treatment environment, it also discusses the possible utilization of the SWAN framework[9] [45] to optimize the system performance. For simplicity and consistency across all examples (water treatment vs. energy storage systems), our experimental water treatment environment encompasses only the recommendations from [45] that pertain to the more general PURDUE model.

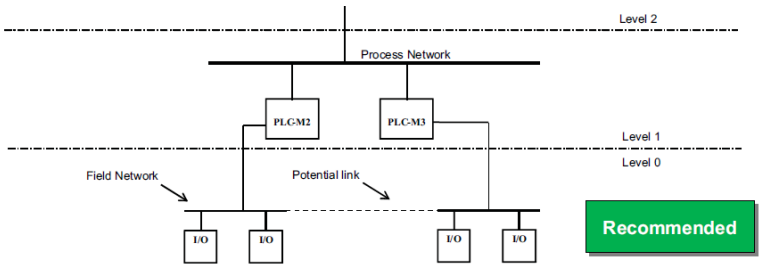


Figure 43: Segregation of Field and Process Networks [46]

[9] The SWAN framework (Smart Water Network) consists of five layers: physical assets, sensing and control, data communication, data management, and data analysis, enabling efficient water system management through real-time monitoring and integration of technology.

On top of the system components and functionalities outlined in [45] and [46], our hypothetical water treatment environment additionally assumes: 1) the use of the OPC UA protocol for communication among system components, and 2) the presence of an SKS server for exchange of respective encryption keys. As for the actual integration of OPC UA within an ICS environment, we have followed the recommendations from [47], where the following OPC UA components are placed in Layer 3: main OPC UA server and database(s), OPC UA workstations, and SKS server.

With this knowledge in hand, we have structured our experimental water treatment plant environment as shown in Figure 44.

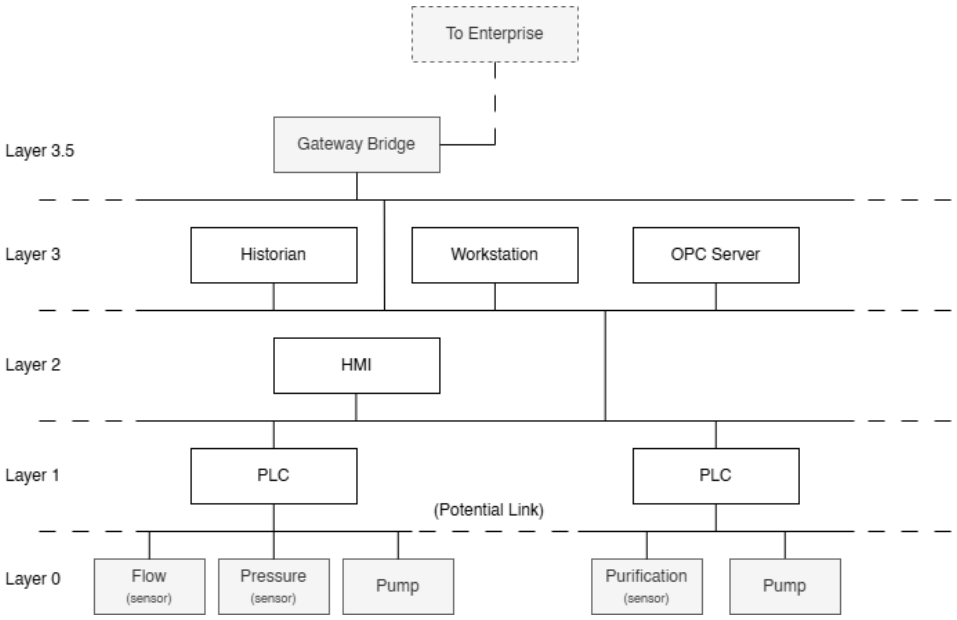


Figure 44: Our Water Treatment Environment

From the PURDUE model perspective, the key features of the environment in Figure 44 are as follows:

- At the field level (Layer 0) of our envisioned water treatment facility, there are five active OPC UA enabled physical devices – 3 sensors and 2 actuators.
- At Layer 1, there are two PLCs which communicate with each of their respective sets of field devices, reflecting the arrangement recommended in [46]. The two PLCs also connect to Layer 2 which hosts a Human-Machine Interface (HMI) terminal.
- Layer 3 consists of a historian, a workstation and the main OPC Server which also hosts the SKS Server. This layer directly connects to Layer 3.5 (OT's DMZ).

- Finally, a gateway bridge in Layer 3.5 connects the ICS environment with the Enterprise environment.

In this sample environment, the leftmost PLC performs the water control function, while the rightmost PLC performs purification control, as shown in Figure 45. **Water Control** involves a flow and a pressure sensor that monitors the current water flow and pressure conditions, which are subsequently communicated with the PLC (at Layer 1) which controls the pump. The pump in turn adjusts the pumps and valves to achieve the desired water flow and pressure through the system. **Purification Control** operates in a similar manner, where the purification sensor reports various measurements to a Layer 1 PLC, which in turn controls the pump that is responsible for adjustment (i.e., achieving) of ideal purification levels.

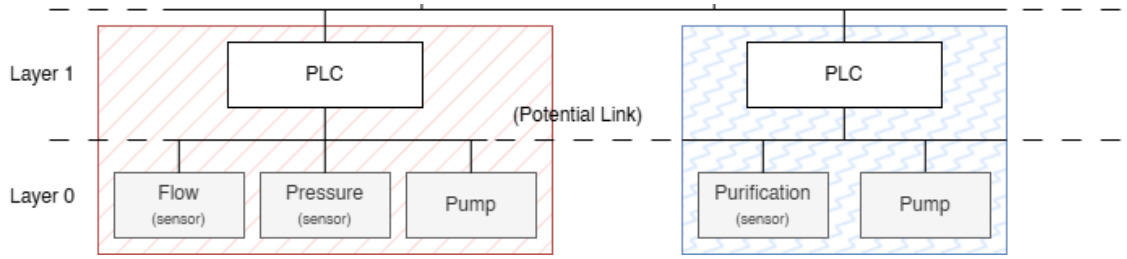


Figure 45: Red – Water Control, Blue – Purification Control

The Layer 2 and 3 devices assist with the floor level processes. Namely, the HMI terminal allows operators to manually override the pumps. The historian collects and stores process data over time for analysis. The workstation is used by operators for high-level actions like performing advanced diagnostics and running management software. Lastly, the OPC server facilitates communication between security groups and oversees their respective key exchanges.

As for the actual make of the devices in Figure 44, we consider a mix of devices from a single manufacturer – Siemens, with the following specific functionalities:

- It is uncommon for Layer 0 devices to have associated CVEs, so for the purposes of our analysis, we will assume them to be generic[10]. An example of Siemens water flow sensor potentially deployed at Layer 0 is SITRANS FM MAG 5100W [48], and it can be easily verified that this sensor does not have any known software vulnerabilities reported in the NVD CVSS database. (For environments which have CVEs associated with Layer 0 devices, ARC-C can easily support their inclusion.)

[10] "Generic" devices are categorized by function alone, allowing for a more flexible assessment of the system without dependency on the unique characteristics or vulnerabilities of individual products.

- The two Layer 1 PLCs are S7-200 SMART devices known for their cost-effectiveness and suitability for simpler control and monitoring tasks [49].
- The SIMATIC Basic device [50] placed in Layer 2 comes equipped with essential HMI functionality, allowing for efficient process visualization and control, as well as seamless interaction/integration with the S7-1200 [51].
- The SIMATIC S7-1200 [51] in Layer 3 has been chosen as the OPC UA server of choice due to its native support for this specific protocol and enough processing power to conduct higher level processes.
- The SIMATIC 2022 device [52] is the Layer 3 historian.
- The SIMATIC IPC 1047e [53] is a Layer 3 workstation with high processing power and reliability, which make it well suitable for industrial applications that require continuous operation and data processing.

Table 4 lists all the devices of the given environment together with their associated CVEs, as reported in the NVD Vulnerability Database [9]. Note that *CVE-2023-51438* has an overall severity score of 9.8, with its exploitability specifically having the maximum 3.9 score, and confidentiality impact being set to HIGH. This is known as a critical vulnerability and (in the context of our ARC-C framework) can alone set the $P_{VE\text{-}software\text{-}succ}$ = 1 if not patched. In the remainder of this section, we examine several different examples pertaining to the environment from Figure 44, with each example considering a different situation in terms of application of patches or their lack of.

	Severity	***Exploitability***	***Confidentiality***
SIMATIC Process Historian 2022			
CVE-2021-3449	5.9	2.2	NONE
CVE-2021-40142	7.5	3.9	NONE
SIMATIC IPC1047E			
CVE-2023-23588	6.3	1.0	HIGH
CVE-2023-51438	9.8	3.9	HIGH
SIMATIC S7-1200			

CVE-2017-2680	6.5	2.8	NONE
CVE-2017-2681	6.5	2.8	NONE
CVE-2017-12741	7.5	3.9	NONE
CVE-2018-13815	7.5	3.9	NONE
CVE-2019-13945	6.8	0.9	HIGH
CVE-2020-28400	7.5	3.9	NONE
SIMATIC HMI Basic			
CVE-2020-7592	6.5	2.8	HIGH
*SIMATIC S7-200 SMART **(x2)***			
CVE-2017-2680	6.5	2.8	NONE
CVE-2017-2681	6.5	2.8	NONE

Table 4: CVEs associated with each device (Water Treatment), CVE-2023-51438 is a critical vulnerability

Finally, before presenting the specific application/test-case scenarios, let us point out that in the context of ARC-C tool and framework, the criticality of different Functional Objectives ($FO_{criticality}(j)$, j=1,..,4, from Section 5.3.2) in our hypothetical Water Treatment Plant have been set as shown in Figure 44, and taking the following into consideration:

- The Criticality of Operational Objectives ($FO_{criticality}(1)$) is set to '**High**', as Water Treatment Plants are generally expected to have continuous uninterrupted operation, and failing to do so would have serious consequence for the communities that depend on them.
- The Criticality of Safety Objectives ($FO_{criticality}(1)$) is set to '**Critical**'. It is common knowledge that water treatment facilities play a pivotal role in the health and safety of general public and environment, with the potential to affect entire communities and ecosystems if compromised.
- The Criticality of Financial Objectives is set to '**Medium**', as many Water Treatment Facilities are publicly owned and do not operate for profit.
- The Criticality of Privacy and Legislative Objectives is set to '**High**'. In 2022, Canadian Government has introduced Bill C-26 (with an amendment called Critical Cyber Systems

Protection Act – CSSPA) [54], which requires the operators of critical infrastructure – water treatment facilities being one of them – to implement a wide range of protections against cyber incidents, including those impacting the confidentiality of their system as well as their data/information. The CSSP also mandates the operators of these systems to ensure that cyber incidents affecting critical infrastructure are adequately detected and reported, and their overall impact minimized. A failure to comply with the CSSPA could result in a monetary penalty of up to $10,000,000 for a corporation, while implicated individuals could face either a monetary penalty or imprisonment. Similar legislations also exist in other countries.

Note that another critical ARC-C parameter – the Functional Importance of the transmitted data (FI_{SG}) - in our hypothetical Water Treatment Plant will depend on the specific role/purpose of each particular security group and will be assigned separately in each individual test-case scenario presented in the remainder of this section.

Criticality on Different Functions					
Operational	None	Low	Medium	High	Critical
Safety	None	Low	Medium	High	Critical
Financial	None	Low	Medium	High	Critical
Privacy and Legislative	None	Low	Medium	High	Critical

Figure 46: Impact weighting ($FO_{criticality}$) for a water treatment facility

7.2.3 Test-Case 1: Water Treatment Facility with Strong Security

7.2.3.1 Water Treatment Facility Device and Security Group Setup

In our first test-case we assume the water treatment facility described in Section 7.2.2 is diligently following and complying with security standards and is up to date with the latest industry guidelines for cybersecurity and operational safety. This includes: 1) separating OPC UA devices into different security groups based on their function; 2) patching all CVEs with a CVSS severity score of Medium or above (as shown in the below table)[11], and 3) implementing strong procedural security policies. With regards to encryption of in-transit data, the operator sets the minimum (acceptable) cryptoperiod to be 1 day and uses the recommended 1-year as the maximum

[11] Patching a vulnerability requires temporarily taking the affected device offline, which may slow or halt production. For environments which require uninterrupted operation (i.e., a Water Treatment Facility), patches should be limited and reserved for serious vulnerabilities.

allowed cryptoperiod duration. By utilizing the ARC-C tool, we are aiming to adjust/optimize the duration of cryptoperiods so that they are commensurate with the actual risk levels of each individual security group.

	Exploitability	***Confidentiality***	***Status***
SIMATIC Process Historian 2022			
CVE-2021-3449	2.2	NONE	PATCHED
CVE-2021-40142	3.9	NONE	PATCHED
Simatic IPC1047E			
CVE-2023-23588	1.0	HIGH	PATCHED
CVE-2023-51438	3.9	HIGH	PATCHED
SIMATIC S7-1200			
CVE-2017-2680	2.8	NONE	PATCHED
CVE-2017-2681	2.8	NONE	PATCHED
CVE-2017-12741	3.9	NONE	PATCHED
CVE-2018-13815	3.9	NONE	PATCHED
CVE-2019-13945	0.9	HIGH	PATCHED
CVE-2020-28400	3.9	NONE	PATCHED
SIMATIC HMI Basic			
CVE-2020-7592	2.8	HIGH	PATCHED
*SIMATIC S7-200 SMART **(x2)***			
CVE-2017-2680	2.8	NONE	PATCHED
CVE-2017-2681	2.8	NONE	PATCHED
Flow Sensor			
Pressure Sensor			
Purification Sensor			
Pump (x2)			

Table 5: Patched CVEs for Test-Case 1 (Water Treatment)

In this test case, we specifically assume that three separate OPC UA security groups are created in this system, shown in Table 6 and Figure 47.

Group 1	Group 2	Group 3
Flow Sensor	*Purification Sensor*	HMI
Process Sensor	*Pump*	Historian
Pump	PLC	Workstation
PLC	HMI	OPC Server / SKS Server
HMI	OPC Server / SKS Server	
OPC Server / SKS Server		

Table 6: Water treatment security groups - Italicized devices are Layer 0

Groups 1 and 2 (process level groups) each support one of the two main subprocess functions: water flow control and purification control respectively. It should be noted that in Layer 1 both groups deploy the identical type of PLC device, while in Layer 2 they deploy the very same HMI terminal unit. Also, both groups rely on the same OPC Server (with SKS Server functionality) to acquire their respective encryption keys. The only notable difference between the two groups is that Group 1 contains one additional Layer 0 device (an extra sensor for flow control).

Group 3 (supervisory level group) exists to support secure communication among the higher-level devices and allow them to indirectly communicate with Layer 0 and 1 devices using the HMI as an intermediary [46].

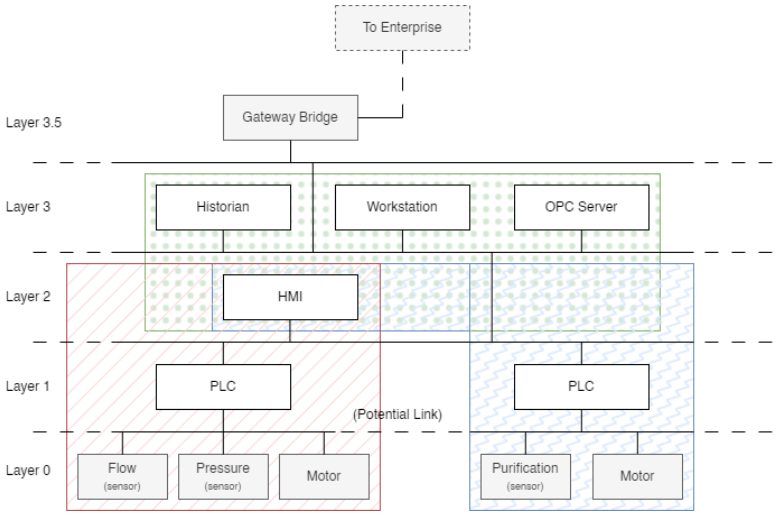


Figure 47: Water treatment security groups – Red = Group 1, Blue = Group 2, Green = Group 3

Using the above information, we will deploy the ARC-C tool to each security groups to derive their unique optimal cryptoperiods. For each of the groups, we will consider five possible Information Rates (IR_{SG}) and all four possible procedural security policy strengths ($P_{VE\text{-}procedure\text{-}succ}$), as outlined in Figure 28 and Figure 29 of Section 5.

7.2.3.2 Cryptoperiod for Security Group 1

Before presenting the optimal cryptoperiod values for Security Group 1 as calculated using the ARC-C tool, let us put in context several critical intermediate parameters/values that need to be calculated by the tool or set by the user/stakeholder (as per the ARC-C theoretical framework from Chapter 6) to derive the final results.

First, using the algorithm for calculation of $P_{VE\text{-}software\text{-}succ}$ (see Section 5.2.1), and based on the CVE related information from Table 5, the probability of software compromise for Group 1 (i.e., its members) is calculated by the ARC-C tool to be 0.17 (17%). This is a rather low value and is in line with the assumed good security practices specifically pertaining to software vulnerability management – i.e., with all critical software vulnerabilities being patched.

Another set of intermediate parameters required by the ARC-C tool to calculate the optimal cryptoperiod are the Functional Impact(s) of a potential group's data compromise on four main Functional Objectives - $FI_{SG}(j)$, j=1, 2, 3, 4 (see Section 5.3.2). As previously explained, Group 1 oversees water flow control, and the purpose of data exchanged among its members is to support this particular function/process. The compromise of Group 1 data - in the specific case of a DS attack (as discussed in this work) - would directly impact only the data's confidentiality and potentially some aspects of system confidentiality. Though, with access to this data, the adversary would be in a position to plan future attacks (i.e., post reconnaissance steps), which could potentially (but only down the road!) affect the Operational and Safety Objectives of the organization.

- For this reason, we set Functional Impact of group's data compromise on these two Functional Objectives ($FI_{SG}(1)$ and $FI_{SG}(2)$) to '**Medium**'.
- At the same time, the impact of group's data compromise would have no to minimal impact on Financial and Privacy and Legislative Objectives, so consequently we set $FI_{SG}(3)$ and $FI_{SG}(4)$) to '**Low**'.

All four $FI_{SG}(j)$, j=1, 2, 3, 4 values, when input into ARC-C tool's GUI, are shown in the below figure.

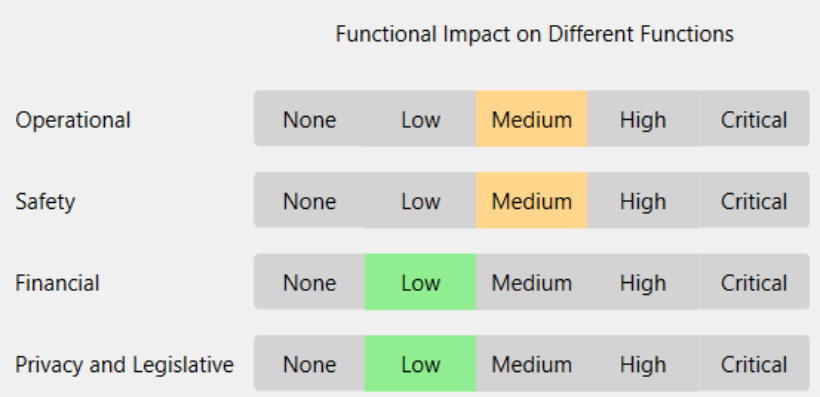


Figure 48: Functional Impact (FI_{SG}) of Group 1 Data Compromise (Water Treatment)

Calculating the total functional impact of this security group - by using both FI_{SG} (from Figure 48) and $FO_{criticality}$ (from Figure 46) - results in FI_{SG} = 0.57.

With all the above values and parameters in place, the ARC-C tool is finally deployed to calculate the optimal cryptoperiod of Group 1, while considering a range of possible values for the estimated strength of procedural security policies deployed by the organization (corresponding to (1- $P_{VE\text{-}procedure\text{-}succ}$)) as well as the Information Rate of the security group (IR_{SG}) (see Table 7). The table is organised so that all values of Information Rate (IR_{SG}) are tested for one single procedural security policy strength (1 – $P_{VE\text{-}software\text{-}succ}$) before moving on to a higher procedural security policy strength. By doing so, we want to explore a range of different practical scenarios (e.g., a security group that sends more/less data in a system that deploy more/less strict procedural policies) and examine how these variabilities affect the optimal cryptoperiod calculated by the ARC-C tool.

1 - $P_{VE\text{-}procedure\text{-}succ}$ *(estimated strength of procedural security in security group)*	IR_{SG}	Cryptoperiod
None	Very Low (0.1)	1 month 7 days
	Low (0.3)	23 days 11 hours
	Moderate (0.75)	14 days 18 hours
	High (0.9)	13 days 8 hours

	Very High (1)	12 days 14 hours
Low	Very Low (0.1)	2 months
	Low (0.3)	1 month 11 days
	Moderate (0.75)	29 days
	High (0.9)	26 days
	Very High (1)	25 days
Medium	Very Low (0.1)	3 months 15 days
	Low (0.3)	2 months 22 days
	Moderate (0.75)	2 months 3 days
	High (0.9)	2 months
	Very High (1)	1 month 28 days
High	Very Low (0.1)	6 months 24 days
	Low (0.3)	6 months 1 day
	Moderate (0.75)	5 months 11 days
	High (0.9)	5 months 8 days
	Very High (1)	5 months 5 days

Table 7: Cryptoperiod for Group 1 (Water Treatment) with variable $P_{VE\text{-}procedure\text{-}succ}$ and IR_{SG}
$T_{CP\text{-}min}$ = 1 day, $T_{CP\text{-}max}$ = 1 year, $P_{VE\text{-}software\text{-}succ}$ = 0. 17, FI_{SG} = 0.57

From Table 7 we observe the following:

- As expected, among all the scenarios depicted in the above table, the shortest optimal cryptoperiod of 12 days and 14 hours corresponds to the sub-scenario where the estimated strength of procedural security policies in the system is 'None' and the security group's Information Rate IR_{SG} = Very High (i.e., significant volumes of data are exchanged between group members per time unit). Without any procedural policies in place, it is almost certain for

an adversarial attack to succeed. It should be noted, however, that the optimal cryptoperiod length in this scenario is still longer that the minimum acceptable cryptoperiod set by the organization (1 day) - largely due to the fact that a compromise of the respective data would have a moderate overall impact on the organization.

- Also as expected, the longest cryptoperiod of 6 months and 24 days corresponds to the sub-scenario where the estimated strength of procedural security policy in the system is 'High' and the Information Rate IR_{SG} = Very Low (i.e., small volumes of data are exchanged between group members per time unit). The fact that the optimal cryptoperiod in this specific scenario is slightly over 50% of the (1-year) cryptoperiod length is likely due to the possible indirect use of the given data by the adversary to plan subsequent attacks with potential impact on the system operation and safety (two functional objectives that are considered to be of 'High' and 'Critical' importance for the organization) as well as the non-negligible probability of compromise.

In summary, given that the probability of software compromise in Group 1 is relatively low, and the adverse impact of Group 1 data compromise is moderate, a moderate length cryptoperiod(s) is a reasonable result. Figure 49 depicts the results from Table 7 in a graphical form.

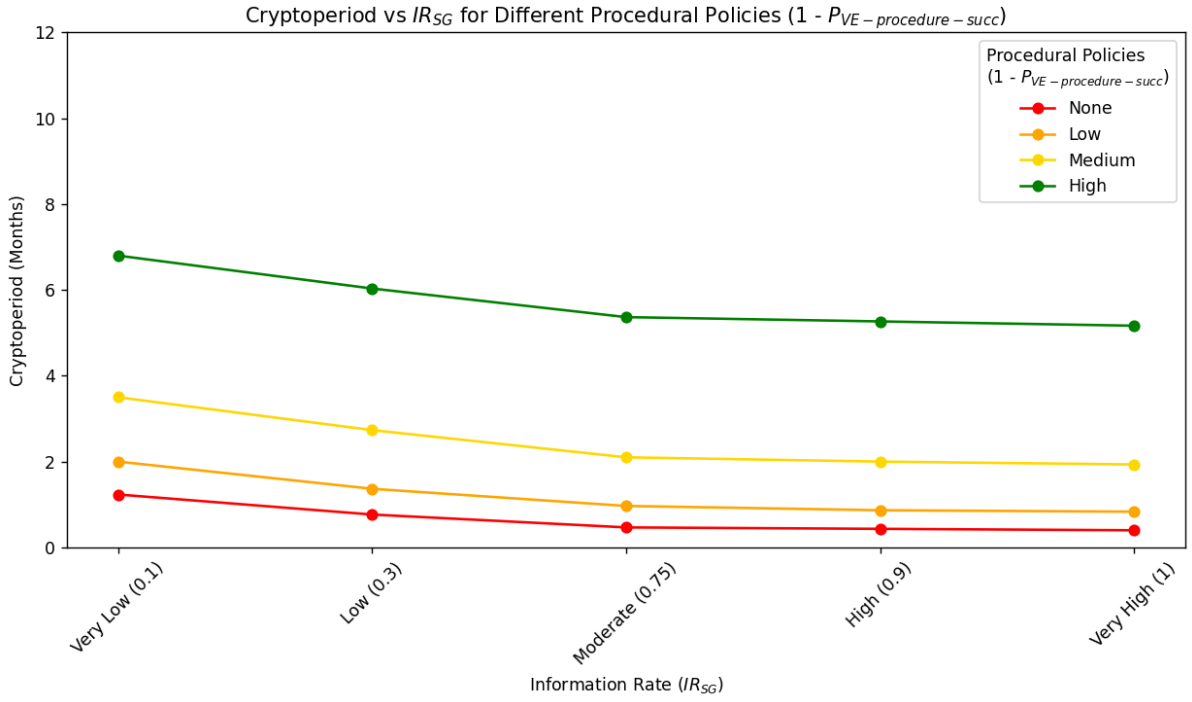


Figure 49: Cryptoperiod for Group 1 with variable 1 - $P_{VE\text{-}procedure\text{-}succ}$ and IR_{SG}
$T_{CP\text{-}min}$ = 1 day, $T_{CP\text{-}max}$ = 1 year, $P_{VE\text{-}software\text{-}succ}$ = 0.17, FI_{SG} = 0.57

7.2.3.3 Cryptoperiod for Security Group 2

In addition to having identical Layer 1 to Layer 3 members, here we also assume that the devices of Group 1 and Group 2 are identically configured, and are transmitting similar types of control data (in this case water-purification control related) in the same time-wise manner. The only distinction between these two groups from the perspective of the cryptoperiod calculation is that Group 2 has one less process level (Layer 0) device. This causes $P_{VE\text{-}software\text{-}succ}$ to be slightly lower, at 0.14 (14%).

As both functionalities (water control for Group 1 and purification control for Group 2) have a similar scope, all four functional impacts (FI_{SG}) also remain the same, as shown in Figure 48.

Running the same tests as for Group 1, we get the following cryptoperiod results, which are very similar in nature as those from Table 8:

1 - $P_{VE\text{-}procedure\text{-}succ}$ *(estimated strength of procedural security in the system)*	IR_{SG}	Cryptoperiod
None	Very Low (0.1)	1 month 7 days
	Low (0.3)	23 days 11 hours
	Moderate (0.75)	14 days 18 hours
	High (0.9)	13 days 8 hours
	Very High (1)	12 days 14 hours
Low	Very Low (0.1)	2 months 1 day
	Low (0.3)	1 month 12 days
	Moderate (0.75)	29 days 7 hours
	High (0.9)	27 days 2 hours
	Very High (1)	25 days 20 hours

Medium	Very Low (0.1)	3 months 19 days
	Low (0.3)	2 months 25 days
	Moderate (0.75)	2 months 7 days
	High (0.9)	2 months 3 days
	Very High (1)	2 months 1 day
High	Very Low (0.1)	7 months 5 days
	Low (0.3)	6 months 14 days
	Moderate (0.75)	5 months 24 days
	High (0.9)	5 months 20 days
	Very High (1)	5 months 18 days

Table 8: Cryptoperiod for Group 2 (Water Treatment) with variable $P_{VE\text{-}procedure\text{-}succ}$ and IR_{SG} $T_{CP\text{-}min}$ = 1 day, $T_{CP\text{-}max}$ = 1 year, $P_{VE\text{-}software\text{-}succ}$ = 0.14, FI_{SG} = 0.57

Comparing the results between Security Group 1 and Security Group 2, there is an interesting pattern to note – stemming from the decrease in $P_{VE\text{-}software\text{-}succ}$ (from 17% to 14%). This decrease in probability lengthens the cryptoperiod in most scenarios and is particularly apparent as the Procedural Policy (1 – $P_{VE\text{-}procedure\text{-}succ}$) gets stronger (None → High). Recalling the formula in Section 5.2, the overall probability ($P_{SG\text{-}KC\text{-}succ}$) is the combined probability of individual events $P_{VE\text{-}procedure\text{-}succ}$ and $P_{VE\text{-}software\text{-}succ}$. For reference, we can compare these values:

1 - $P_{VE\text{-}procedure\text{-}succ}$ *(estimated strength of procedural security in the system)*	**$P_{SG\text{-}KC\text{-}succ}$ - Group 1** $P_{VE\text{-}software\text{-}succ}$ = 0.17	**$P_{SG\text{-}KC\text{-}succ}$ - Group 2** $P_{VE\text{-}software\text{-}succ}$ = 0.14	**Change in $P_{SG\text{-}KC\text{-}succ}$**
None	1	1	0
Low	0.793	0.785	0.008
Medium	0.544	0.527	0.017
High	0.2530	0.226	0.027

Table 9: Comparison of $P_{SG\text{-}KC\text{-}succ}$ between Security Group 1 and Group 2 (Water Treatment)

Table 9 shows that when $1 - P_{VE\text{-}procedure\text{-}succ}$ = None, the value of $P_{SG\text{-}KC\text{-}succ}$ is 1, regardless of the value of $P_{VE\text{-}software\text{-}succ}$. However, as the procedural policies improve, the influence of $P_{VE\text{-}software\text{-}succ}$ on $P_{SG\text{-}KC\text{-}succ}$ becomes more pronounced (the change in $P_{SG\text{-}KC\text{-}succ}$ increases). Since Security Group 2 has a smaller probability of software compromise, this lengthens the cryptoperiod as the environment is more secure (on the side of software vulnerabilities) than Security Group 1.

7.2.3.4 Cryptoperiod for Security Group 3

As previously indicated, Group 3 consists of only Layer 2 and 3 devices – see Table 6. Like Group 1 and 2, Group 3 also does not contain any unpatched CVEs – see Table 5. As a result, for this group, the probability of software compromise is calculated to be 0.11 (11%).

However, the main difference relative to Group 1 and 2 is that Group 3 contains higher-level supervisory control devices, therefore it is expected that more important information is being communicated during this group members' data exchange. Consequently, higher values of Functional Impact of group data compromise on organization' Functional Objectives are assumed. Specifically:

- Functional Impact of Group 3 data compromise on Operation and Safety Objectives ($FI_{SG}(1)$ and $FI_{SG}(2)$) are now set to '**High**'. Namely, achieving access to data exchanged between Layer 3 devices could grant an adversary a wide breadth of knowledge pertaining to system configurations, system operation, and control logic for multiple lower-level processes.
- Functional Impact on Financial Objectives ($FI_{SG}(3)$) increased from 'Low' to '**Medium**', as a subsequent attack can lead to a wider range of financial damages.
- Functional Impact on Privacy and Legislative Objectives ($FI_{SG}(3)$) is also increased from 'Low' to '**Medium**'. An attack on higher level devices extends beyond a minor security breach, potentially leading to more serious repercussions.

These changes drastically increase the ultimate Functional Impact of a potential DS attack on Group 3 (relative to Group 1 and 2) returning a value of FI_{SG} = 0.99 - close to the maximum value of 1.

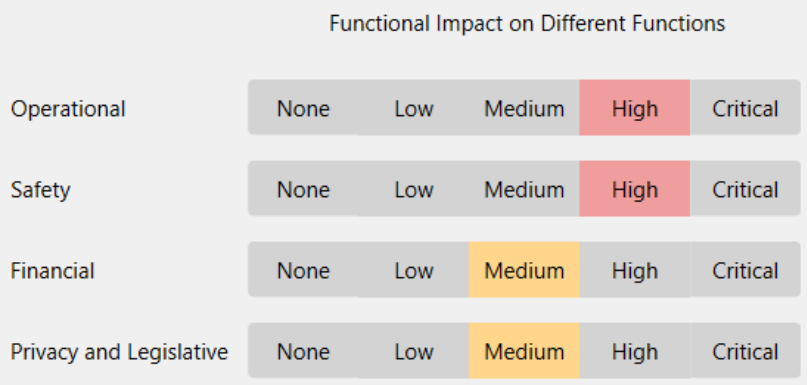


Figure 50: Functional Impact (FI_{SG}) of Security Group 3 Data Compromise (Water Treatment)

The final cryptoperiod results for the Group 3 sub-scenarios can be found in Table 10. As before, the table is organised so that each value of group's Information Rate (IR_{SG}) is tested for a single procedural security policy strength (1 – $P_{VE\text{-}software\text{-}succ}$) before moving on to a higher procedural policy strength.

1 - $P_{VE\text{-}procedure\text{-}succ}$ *(estimated strength of procedural security in the system)*	IR_{SG}	Cryptoperiod
None	Very Low (0.1)	1 day 3 hours
	Low (0.3)	1 day 2 hours
	Moderate (0.75)	1 day 2 hours
	High (0.9)	1 day 2 hours
	Very High (1)	1 day 1 hour
Low	Very Low (0.1)	4 days
	Low (0.3)	3 days 22 hours
	Moderate (0.75)	3 days 21 hours

	High (0.9)	3 days 21 hours
	Very High (1)	3 days 20 hours
Medium	Very Low (0.1)	18 days 13 hours
	Low (0.3)	18 days 8 hours
	Moderate (0.75)	18 days 4 hours
	High (0.9)	18 days 3 hours
	Very High (1)	18 days 3 hours
High	Very Low (0.1)	3 months 21 days
	Low (0.3)	3 months 21 days
	Moderate (0.75)	3 months 20 days
	High (0.9)	3 months 20 days
	Very High (1)	3 months 20 days

Table 10: Cryptoperiod for Group 3 (Water Treatment) with variable $P_{VE\text{-}procedure\text{-}succ}$ and IR_{SG} $T_{CP\text{-}min}$ = 1 day, $T_{CP\text{-}max}$ = 1 year, $P_{VE\text{-}software\text{-}succ}$ = 0.11, FI_{SG} = 0.99

From Table 10 we observe the following:

- The shortest cryptoperiod for this security group is 1 day and 1 hour (when there is no procedural policies and the group's Information Rate is very high)
- The longest cryptoperiod for this security group is 3 month and 21 days (when procedural policy is strong and the group's Information Rate is low)

The possible cryptoperiod values remain similar for different information rates (IR_{SG}) but varies drastically for different procedural policies ($1 - P_{VE\text{-}procedure\text{-}succ}$). This occurs as with FI_{SG} approaching 1 (0.99 in this example), the overall information rate in the system becomes negligible (as discussed in Section 5.3.3).

It should be noted that in all scenarios, Group 3 requires a shorter cryptoperiod compared to Groups 1 and 2 due to the critical functional impact. This is exasperated in environments with weak procedural policies.

7.2.4 Test-Case 2: Water Treatment Facility with Weak Security

Our next example will assume less-than-ideal security conditions. Namely, due to negligence or lack of knowledge, the system administrators fail to follow the basic security practices and miss to implement critical updates. Only the most severe CVE (CVE-2023-51438) is patched, while all the other vulnerabilities enlisted in Table 4 are left unpatched, causing the system to be very vulnerable to potential adversarial attacks. A single security group is created for all devices (which is another indication of poor security related practices), ultimately causing the software compromise score to be 0.91 (91%) – as calculated by ARC-C. The operator sets a minimum cryptoperiod of 1 day and uses the recommended 1-year maximum duration.

With a security group encompassing all devices, an even higher Functional Impact of the group's data than in the previous test cases is expected. To explain the specific implications of adding Layer 0 and 1 devices to a security group already comprising Layer 2 and 3 devices, we will use the Functional Impact values discussed in Section 7.2.3.4 (Security Group 3 of Test-Case 1) as a base for comparison.

- In the case of Group 3 in Test-Case 1 (i.e., a security group solely consisting of supervisory control devices and potentially exchanging some critical system-level data), it was deemed that the Functional Impact of this group's data on both Operation and Safety Objectives ($FI_{SG}(1)$ and $FI_{SG}(2)$) should be set to 'High'. Introducing Layer 0 and Layer 1 devices into the security group would put an adversary in an ideal position to understand the system and conduct future attacks. This would increase $FI_{SG}(1)$ and $FI_{SG}(2)$ to '**Critical'**.
- When it comes to Functional Impact of group's data on Financial objectives ($FI_{SG}(3)$), we determined it results in a 'Medium' impact for supervisory control devices. The impact of a security group's data confidentiality compromise will not change much regardless of the actual scope of the data or the number of devices in the security group. Therefore, $FI_{SG}(3)$ remains at '**Medium**'.

- A successful compromise of a larger group's data may be an indicator of more systematic cyber security problems in the organization, ultimately resulting in more severe Legislative penalties. Consequently, this warrants setting $FI_{SG}(4)$ to '**High**'.

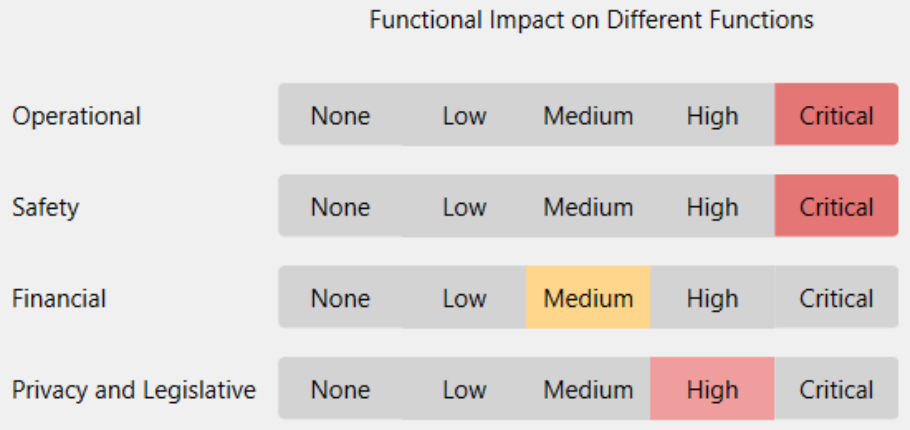


Figure 51: Functional Impact (FI_{SG}) of Supergroup Data Compromise (Water Treatment)

In this scenario, the overall Functional Impact of group's data is 1 (using FI_{SG} (from Figure 51) and $FO_{criticality}$ (from Figure 46)), making it the maximum possible value.

Like Test-Case 1, the test will consider variable traffic intensity (IR_{SG}) rates and different policy strengths (1 - $P_{VE\text{-}procedure\text{-}succ}$). Each sub-scenario is displayed with its associated cryptoperiod length, calculated by ARC-C.

1 - $P_{VE\text{-}procedure\text{-}succ}$ *(estimated strength of procedural security in the system)*	IR_{SG}	Cryptoperiod
None	Very Low (0.1)	1 day
	Low (0.3)	1 day
	Moderate (0.75)	1 day
	High (0.9)	1 day
	Very High (1)	1 day

Low	Very Low (0.1)	1 day 3 hours
	Low (0.3)	1 day 3 hours
	Moderate (0.75)	1 day 3 hours
	High (0.9)	1 day 3 hours
	Very High (1)	1 day 3 hours
Medium	Very Low (0.1)	1 day 8 hours
	Low (0.3)	1 day 8 hours
	Moderate (0.75)	1 day 8 hours
	High (0.9)	1 day 8 hours
	Very High (1)	1 day 8 hours
High	Very Low (0.1)	1 day 15 hours
	Low (0.3)	1 day 15 hours
	Moderate (0.75)	1 day 15 hours
	High (0.9)	1 day 15 hours
	Very High (1)	1 day 15 hours

Table 11: Cryptoperiod for Supergroup (Water Treatment) with variable $P_{VE\text{-}procedure\text{-}succ}$ and IR_{SG} $T_{CP\text{-}min}$ = 1 day, $T_{CP\text{-}max}$ = 1 year, $P_{VE\text{-}software\text{-}succ}$ = 0.91, FI_{SG} = 1

From Table 11, we observe the following:

- The shortest cryptoperiod is 1 day
- The longest cryptoperiod is 1 days and 15 hours

These results seem justifiable, as in an environment with a high probability of compromise and a maximum data importance in terms of impact, it is expected that the calculated cryptoperiod should be close the minimum duration ($T_{CP\text{-}min}$) – which in this scenario ranges from 1 to 1 day and 15 hours.

With a Functional Impact (FI_{SG}) set to its maximum value, IR_{SG} has no impact on the final cryptoperiod. Instead, the formula assumes the worst-case scenario regardless of the rate of data. This explains why the value remains constant across different IR_{SG} values under the same procedural policy. Similarly, as $P_{VE\text{-}software\text{-}succ} \rightarrow 1$ (0.91), the change in procedural policy has a small (but not non-existent) effect on the overall cryptoperiod.

7.3 Evaluation of ARC-C in an Energy Storage System (ESS)

The second experimental ICS environment considered in our work is an Energy Storage System (ESS). ESSs are becoming more prevalent and essential for modern-day power grids, making them prime targets for adversarial attacks [55]. There have already been several notable attacks to power grids. For example, in May of 2023, 22 energy companies in Denmark were hacked when attackers exploited a critical vulnerability in a firewall [56]. In 2019, visibility was lost to 500 MW of wind and photovoltaic sites in the US via a DoS attack on a firewall with an unpatched vulnerability [57].

The integration of ICS within ESS environments allows operators to manage a plethora of functionalities. However, this integration also introduces vulnerabilities, as ESSs rely heavily on communication protocols (i.e., OPC UA) to coordinate operations across its assets. Similarly to the water treatment example, testing ARC-C within this context provides an opportunity to validate its effectiveness.

7.3.1 ESS: Background

Unlike the water treatment environments, there is a wide range of detailed documentation regarding the specific structure of ESSs. To begin, Figure 52 from [58] provides a detailed implementation of an ESS, represented by the PURDUE model, outlining explicit devices and functionalities expected from an ESS.

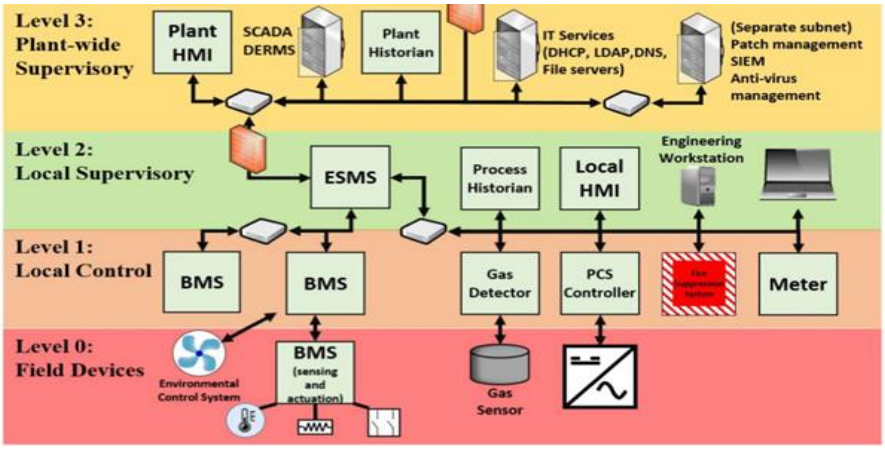


Figure 52: Detailed ESS environment [58]

An important detail to note is that for smaller ESSs, either home or small industrial organisations, the functions of the HMI, Battery Management System (BMS)[12] [59], and Energy Storage Management System (ESMS)[13] [60] can be implemented on the same device. The BMS can also be its own device while the HMI encapsulates the role of an ESMS. At the same time, the Environmental Control and Fire Suppression Systems are not mandatory [60].

For our experiment, we will assume the ESS is for a small organization which does not have duplicate devices for different levels of supervisory control (e.g., there will be only one HMI rather than a local and a plant HMI). Since we will be simulating a small-scale operation, we can integrate the functionalities of the ESMS into the HMI. With this arrangement, we must ensure that the actual deployed device can satisfy the needs of a typical HMI as well as the additional processes required by the ESMS. We can also forgo more advanced mechanisms such as Environmental Control and Fire Suppressions Systems, although this modification will need to be reflected in how the operator determines their Functional Impact (FI_{SG}).

7.3.2 ESS: Assumed System Structure and Features

Our test environment is a simplified version of Figure 52, only accounting for core functions. Viewed through the PURDUE model, the environment's key features are outlined as follows:

[12] Battery Management Systems (BMSs) are devices/systems which monitor and manage battery packs.
[13] Energy Storage Management Systems (ESMSs) are systems that oversees and optimizes the charging and discharging processes of an ESS.

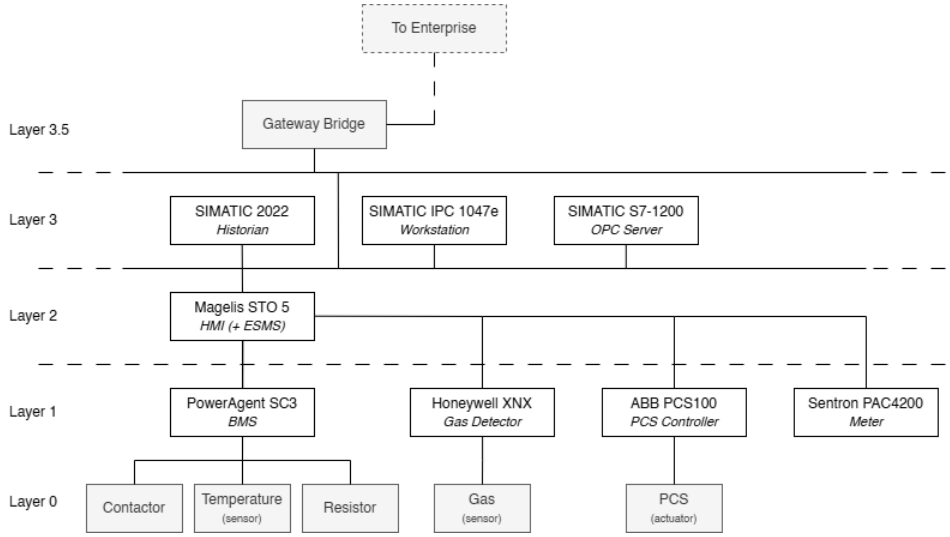


Figure 53: Our ESS environment

- At the field level (Layer 0), there are 5 devices active in the network: 3 sensors, a contactor, and a resistor.
- At Layer 1, we have specialty devices which communicate with each of their respective sets of field devices (excluding the Smart Meter which is a standalone device)
- Layer 2 consists of the multi-function HMI + ESMS, as discussed previously.
- As in the case of the water treatment example, Layer 3 consists of a historian, a workstation and the main OPC Server which also hosts the SKS Server. This layer directly connects to Layer 3.5 (OT's DMZ).
- A gateway bridge connects the ICS environment with the Enterprise environment.

In total, there are three distinct processes being performed in the ESS system from Figure 53, as shown in Figure 54. **Battery Management** is overseen by the HMI and is connected to BMS (Battery Management System). The BMS manages the contactor, the temperature sensor, and the resistor to ensure the optimal use of energy in the battery. **Gas Detection** has a separate device which controls a gas sensor. As gas generation can occur during the battery management and power conversion process, this system alerts an operator if any escapes. **Power Conversion** uses a PCS Controller to manage the actuator (bidirectional inverter) itself. This system mediates bidirectional electrical power flow between ESS and the power grid [58]. There is also a smart meter which displays information about the environment and communicates directly with the HMI.

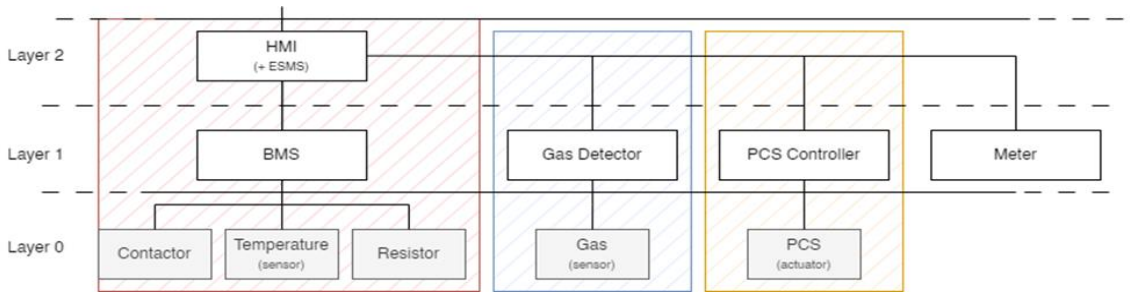


Figure 54: Red – Battery Management, Blue – Gas Detection, Orange – Power Conversion

Unlike the water treatment facility use cases from Section 7.2 that assumed devices used were from the same manufacturer, this use case will instead use a mix of devices from multiple manufacturers:

- We will make the same assumption that the Layer 0 devices have no associated CVEs (Dräger Polytron 7000 is an example of a possible gas sensor) and will assume them to be generic (as described in Footnote 10).
- The PowerAgent SC3 [61] was chosen for its capability to remotely monitor energy storage systems and efficiently manage critical data.
- The Honeywell XNX [62] is valued for its flexibility in gas detection.
- The ABB PCS100 [63] was chosen for providing seamless grid integration, and advanced power control.
- The Sentron PAC4200 [64] was chosen for its capability to monitor electrical parameters and manage power consumption.
- Manufactured by Schneider Electric, the Magelis STO 5 HMI [65] offers an alternative to the SIMATIC Basic at a similar price point.
- The Layer 3 devices remain the same as in Test Case 7.2.

To display the CVEs associated with each device, we will use the same format as used in Section 7.2. Table 12 shows each device's CVEs in our example ESS environment, along with their severity, exploitability, and confidentiality scores.

	Severity	***Exploitability***	***Confidentiality***
SIMATIC Process Historian 2022			
CVE-2021-3449	5.9	2.2	NONE

CVE-2021-40142	7.5	3.9	NONE
SIMATIC IPC1047E			
CVE-2023-23588	6.3	1.0	HIGH
CVE-2023-51438	9.8	3.9	HIGH
SIMATIC S7-1200			
CVE-2017-2680	6.5	2.8	NONE
CVE-2017-2681	6.5	2.8	NONE
CVE-2017-12741	7.5	3.9	NONE
CVE-2018-13815	7.5	3.9	NONE
CVE-2019-13945	6.8	0.9	HIGH
CVE-2020-28400	7.5	3.9	NONE
Magelis STO5			
CVE-2016-8367	5.3	3.9	NONE
CVE-2016-8374	7.5	3.9	NONE
PowerAgent SC BMS			
CVE-2017-6039	5.3	3.9	NONE
Honeywell XNX			
ABB PCS 100			
SENTRON PAC4200			
CVE-2020-13987	7.5	3.9	NONE
CVE-2020-1737	7.8	3.9	LOW

Table 12: CVEs associated with each device (ESS), CVE-2023-51438 is a critical vulnerability

Compared to critical infrastructure like a Water Treatment Facility, it can be expected that a small industrial ESS would contain less critical functional objectives ($FO_{criticality}$). However, being self-owned and managed 'for profit' (i.e., not run by the city), this system comes with its own unique set of challenges (Figure 55), which have justified the following selection of $FO_{criticality}(1)$ parameters:

- The Criticality of Operational Objectives ($FO_{criticality}(1)$) is set to '**Medium'**. While it may be acceptable for an organization to have some minor disturbances or delays in its operation, it would not be acceptable for the system to have its operation significantly impacted or halted for prolonged periods of time.
- The Criticality of Safety Objectives ($FO_{criticality}(2)$) is set to **'Medium**.' Safety is essential, particularly in managing energy storage and associated risks, though the potential hazards are more contained within the immediate operational environment.
- The Criticality of Financial Objectives ($FO_{criticality}(3)$) is set to **'High**.' As a self-owned and 'for profit' organization, the organization's financial stakes are high, while its disruptions or damage would directly impact the owner's investment and profitability.
- The Criticality of Privacy and Legislative Objectives ($FO_{criticality}(4)$) is set to **'Medium**.' While Bill C-26 does not apply in this scenario, general Occupational Health and Safety regulations mandate adequate safeguards to protect employees and prevent hazards [66].

Criticality on Different Functions					
Operational	None	Low	Medium	High	Critical
Safety	None	Low	Medium	High	Critical
Financial	None	Low	Medium	High	Critical
Privacy and Legislative	None	Low	Medium	High	Critical

Figure 55: Impact weighting ($FO_{criticality}$) for a small industrial ESS

The Functional Importance of transmitted data (FI_{SG}) within the example ESS will be evaluated individually for each Security Group scenario presented in the following section.

7.3.3 Test-Case 3: ESS with Moderate Security

7.3.3.1 ESS Device and Security Group Setup

For our ESS test-case, we will assume the environment to be moderately secured. Some procedures are followed, but important security measures are inconsistently applied, leaving certain vulnerabilities unaddressed. This includes separating OPC UA devices into security groups by PURDUE level (rather than by process) and patching all CVEs with a CVSS score of High or above (as shown in Table 13). The recommended 1 day minimum and 1-year maximum cryptoperiod are used here as well.

	Exploitability	***Confidentiality***	***Status***
SIMATIC Process Historian 2022			
CVE-2021-3449	2.2	NONE	**UNPATCHED**
CVE-2021-40142	3.9	NONE	PATCHED
SIMATIC IPC1047E			
CVE-2023-23588	1.0	HIGH	**UNPATCHED**
CVE-2023-51438	3.9	HIGH	PATCHED
SIMATIC S7-1200			
CVE-2017-2680	2.8	NONE	**UNPATCHED**
CVE-2017-2681	2.8	NONE	**UNPATCHED**
CVE-2017-12741	3.9	NONE	PATCHED
CVE-2018-13815	3.9	NONE	PATCHED
CVE-2019-13945	0.9	HIGH	**UNPATCHED**
CVE-2020-28400	3.9	NONE	PATCHED
Magelis STO5			
CVE-2016-8367	3.9	NONE	PATCHED

CVE-2016-8374	3.9	NONE	**UNPATCHED**
PowerAgent SC BMS			
CVE-2017-6039	3.9	NONE	**UNPATCHED**
Honeywell XNX			
ABB PCS 100			
SENTRON PAC4200			
CVE-2020-13987	3.9	NONE	PATCHED
CVE-2020-1737	3.9	LOW	PATCHED
Contactor			
Temperature Sensor			
Resistor			
Gas Sensor			
PCS Actuator			

Table 13: Patched CVEs for Test-Case 3 (ESS)

In an ideal scenario, each ESS function would be contained in their own security group. Here, security groups are instead separated between the higher (2-3) and lower (1-2) layers. All Layer 1 devices which communicate upward with the HMI are put in a single group and the Layer 3 devices which communicate downward with the HMI are in another.

Group 1		**Group 2**
Contactor	BMS	Historian
Temperature Sensor	Gas Detector	Workstation
Resistor	PCS Controller	HMI *(+ESMS,BMS)*
Gas Sensor	Smart Meter	OPC Server
PCS (actuator)	HMI *(+ESMS,BMS)*	

	OPC Server	

Table 14: ESS security groups - Italicized devices are Layer 0

Group 1 is a singular process level group which encapsulates battery management, gas detection, and power conversion. Group 2 is identical to Group 3 of the water treatment example - existing to facilitate secure communication between higher-level devices while utilizing the HMI as an intermediary to manage and control lower-level devices.

We will test both security groups for various procedural policies and traffic intensities.

7.3.3.2 Security Group 1

As previously stated, in this Test Case, we assume an environment with a moderate security posture where CVEs with High CVSS score are patched, leaving low and moderate vulnerabilities unpatched (Table 13). Using this method, the software probability of compromise for Group 1 ($P_{VE\text{-}software\text{-}succ}$) is 0.41 (41%).

The purpose of Group 1 is to perform all lower-level processes, and the data communicated between devices in this security group would be related to each individual process. A DS attack would solely impact the confidentiality of said processes, but like the Water Treatment Facility example, can potentially lead to a more serious attack in the future.

- Since an adversary would gain access to all process-level data and control logic, they would obtain a deep understanding of how the environment operates without even needing access the supervisor-level devices. The absence of Environmental Control and Fire Suppression Systems also means that this information is instead communicated constantly between each device in the process. As such, a '**High**' Operational ($FI_{SG}(1)$)) and Safety ($FI_{SG}(2)$) Functional Impact is set.
- There is also moderate concern for Financial ($FI_{SG}(3)$), which is set to '**Medium**'. Since this security group includes various functions, unauthorized access could potentially expose process details, creating a risk of targeted equipment damage across operations—a notable consideration for a smaller industrial company.
- There is some concern for Privacy and Legislative ($FI_{SG}(4)$) since this would be considered a moderate breach. As a result, it is set to '**Medium**".

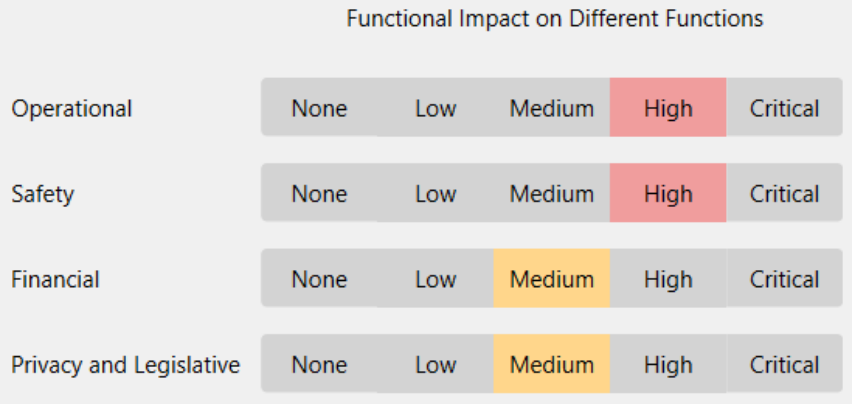


Figure 56: Functional Impact (FI_{SG}) of Group 1 Data Compromise (ESS)

Combining FI_{SG} (Figure 56) and $FO_{criticality}$ (Figure 55), the overall Functional Impact is calculated to be 0.54.

The results for each sub-scenario can be found in Table 15. The table contains all permutations of (1 - $P_{VE\text{-}procedure\text{-}succ}$) and IR_{SG}:

1 - $P_{VE\text{-}procedure\text{-}succ}$ *(estimated strength of procedural security in the system)*	IR_{SG}	Cryptoperiod
None	Very Low (0.1)	1 month 15 days
	Low (0.3)	28 days 8 hours
	Moderate (0.75)	17 days 17 hours
	High (0.9)	15 days 23 hours
	Very High (1)	15 days
Low	Very Low (0.1)	2 months 2 days
	Low (0.3)	1 month 11 days
	Moderate (0.75)	27 days 18 hours

	High (0.9)	25 days 10 hours
	Very High (1)	24 days 2 hours
Medium	Very Low (0.1)	3 months
	Low (0.3)	2 months 5 days
	Moderate (0.75)	1 month 18 days
	High (0.9)	1 month 14 days
	Very High (1)	1 month 13 days
High	Very Low (0.1)	4 months 18 days
	Low (0.3)	3 months 21 days
	Moderate (0.75)	2 months 29 days
	High (0.9)	2 months 25 days
	Very High (1)	2 months 23 days

Table 15: Cryptoperiod for Group 1 (ESS) with variable $P_{VE\text{-}procedure\text{-}succ}$ and IR_{SG}
$T_{CP\text{-}min}$ = 1 day, $T_{CP\text{-}max}$ = 1 year, $P_{VE\text{-}software\text{-}succ}$ = 0.41, FI_{SG} = 0.54

From Table 15 we observe the following:

- The shortest cryptoperiod is 15 days
- The longest cryptoperiod is 4 months 18 days

In this test-case, a relatively moderate probability of software compromise ($P_{VE\text{-}software\text{-}succ}$ = 0.41) results in a modest influence of $P_{VE\text{-}procedure\text{-}succ}$ on the cryptoperiod calculation, creating an approximate three-month difference between the 'None' and 'High' security levels. With a moderate Functional Impact (FI_{SG} = 0.54), IR_{SG} exerts a more significant effect, approximately halving the cryptoperiod between the 'Very Low' and 'Very High' categories.

Overall, this is a somewhat secure environment with a modest impact to the organisation if an attack were to succeed. As such, a relatively strong cryptoperiod is recommended, with leeway given if there is low traffic intensity and/or proper procedural policies are applied.

7.3.3.3 Security Group 2

Security Group 2 contains supervisory level devices. Using the same pattern of applying patches to CVEs as the previous test (CVSS score ≥ High), the probability of software compromise for this security group is 0.61 (61%).

The Functional Impact of the supervisory group is the same as set for Group 1. The scope of an attack on both severity groups are similar – while an attack on higher-level devices (Group 2) can grant an adversary access to system configurations and operators, an attack on lower-level devices (Group 1) contains the entire range of control logic information for each individual process. The same Functional Impact value (FI_{SG} = 0.54) is used.

The results for each sub-scenario can be found in Table 16. The table contains all permutations of $P_{VE\text{-}procedure\text{-}succ}$ and IR_{SG}:

1 - $P_{VE\text{-}procedure\text{-}succ}$ *(estimated strength of procedural security in the system)*	IR_{SG}	Cryptoperiod
None	Very Low (0.1)	1 month 15 days
	Low (0.3)	28 days 8 hours
	Moderate (0.75)	17 days 17 hours
	High (0.9)	15 days 23 hours
	Very High (1)	15 days
Low	Very Low (0.1)	1 month 25 days
	Low (0.3)	1 month 6 days
	Moderate (0.75)	23 days 16 hours
	High (0.9)	21 days 13 hours
	Very High (1)	20 days 9 hours

Medium	Very Low (0.1)	2 months 10 days
	Low (0.3)	1 month 19 days
	Moderate (0.75)	1 month 4 days
	High (0.9)	1 month 1 day
	Very High (1)	29 days 12 hours
High	Very Low (0.1)	3 months 3 days
	Low (0.3)	2 months 9 days
	Moderate (0.75)	1 month 21 days
	High (0.9)	1 month 17 days
	Very High (1)	1 month 15 days

Table 16: Cryptoperiod for Group 2 (ESS) with variable $P_{VE\text{-}procedure\text{-}succ}$ and IR_{SG}
$T_{CP\text{-}min}$ = 1 day, $T_{CP\text{-}max}$ = 1 year, $P_{VE\text{-}software\text{-}succ}$ = 0.61, FI_{SG} = 0.54

From Table 16 we observe the following:

- The shortest cryptoperiod is 15 days
- The longest cryptoperiod is 3 months and 3 days

Comparing the input values from Group 1 and 2, it can be noted that FI_{SG} remains the same (0.54 for both groups). This means any pattern we see that emerges from this test-case comes from the change in $P_{VE\text{-}software\text{-}succ}$, which increased 20% between security groups. (Namely, while the patching criteria is the same, Group 2 has more severe unpatched CVEs).

Like what was observed for Security Group 1 and 2 in Section 7.2.3, the change in cryptoperiod length for Groups 1 and 2 of this section is more apparent as $P_{VE\text{-}procedure\text{-}succ}$ moves from “None” to “High”.

1 - $P_{VE\text{-}procedure\text{-}succ}$ *(estimated strength of procedural security in the system)*	**$P_{SG\text{-}KC\text{-}succ}$ - Group 1** *$P_{VE\text{-}software\text{-}succ}$ = 0.41*	**$P_{SG\text{-}KC\text{-}succ}$ - Group 2** *$P_{VE\text{-}software\text{-}succ}$ = 0.61*	**Change in $P_{SG\text{-}KC\text{-}succ}$**
None	1	1	0
Low	0.853	0.903	0.050

Medium	0.676	0.786	0.110
High	0.469	0.649	0.180

Table 17: Comparison of $P_{SG\text{-}KC\text{-}succ}$ between Security Group 1 and Group 2 (ESS)

Referring to Table 17, the change in $P_{SG\text{-}KC\text{-}succ}$ as $P_{VE\text{-}procedure\text{-}succ} \rightarrow$ High is much more pronounced in this case, reflecting the larger discrepancy between $P_{VE\text{-}software\text{-}succ}$. This in turn results in a larger change in cryptoperiod length, with Group 2 seeing an approximately a third longer cryptoperiod when $P_{VE\text{-}procedure\text{-}succ}$ = Medium compared to Group 1.

7.4 Conclusion

Reflecting on the calculated results, ARC-C demonstrates its effectiveness in optimizing cryptoperiods by balancing security needs with operational requirements. Since no numerical data is available for comparison (as ARC-C is the first tool of its kind), we relied on the NIST standards to broadly guide our decisions. The tool's adaptive approach allows it to adjust cryptoperiod recommendations based on both unique environments and security postures, thereby enhancing the system’s overall resilience against DS attacks.

8 Conclusion

In this Master's thesis, we have explored the multifaceted challenges of securing Industrial Control Systems (ICS) against the emerging risk of Data Siphoning attacks. One key aspect of this research was the role of cryptoperiods in mitigating the impact of such attacks. While existing security standards recognize the importance of cryptoperiods in limiting the exposure of encrypted data, they lack a concrete framework for calculating their optimal duration. To address this, we developed a comprehensive framework and accompanying software tool, ARC-C, designed to calculate risk-based cryptoperiods.

The main research contributions presented in the thesis are as follows:

- development of a detailed attack tree representation/model of DS attacks on an OPC UA security group
- critical evaluation of existing security standards, identifying shortcomings related to cryptoperiod management
- development of an explicit analytical framework for Automatic Risk-based Calculation of optimal Cryptoperiods (ARC-C) in any given real-world OPC UA system
- creation of a user-friendly software tool that fully implements the ARC-C framework.

The use case experimentation conducted in this research involved two distinct ICS environments and multiple security groups of varying sizes and compositions. These experiments validated the effectiveness of the ARC-C framework in accurately calculating optimal cryptoperiods tailored to different risk profiles and demonstrated its practicality and versatility.

The existing ARC-C framework was designed with a focus on OPC UA security groups, which represent one of the most complex application scenarios, as a single encryption key is shared among multiple communicating parties. However, the framework is flexible and can be adapted to simpler application scenarios, such as those involving encryption keys shared exclusively between a single sender and receiver.

In the future, we envision two possible extensions to our research:

1. Repurposing the existing ARC-C framework and tool so they can be utilised to calculate optimal cryptoperiods in IT systems – where (e.g.) impacts and minimal allowed cryptoperiod lengths are different than those in ICS/OT systems.

2. Extending the existing risk-based framework to deal with attacks other than DS.

To conclude, ARC-C provides a strong foundation for optimal cryptoperiod calculation. While the current research has successfully demonstrated its effectiveness and adaptability, there remains significant potential for future advancements. By extending its applicability, ARC-C can evolve into a comprehensive tool for enhancing security across an even broader range of systems and threats.

Bibliography

[1] NIST SP 800-82r3, “Guide to Industrial Control Systems (ICS) Security,” September 2023. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-82r3.pdf.

[2] Trend Micro, “What is an Industrial Control System (ICS)?,” [Online]. Available: https://www.trendmicro.com/vinfo/nl/security/definition/industrial-control-system.

[3] MITRE, “MITRE ATT&CK,” [Online]. Available: https://attack.mitre.org/.

[4] Threat Advice, “APT Attacks Targeting Government Entities,” [Online]. Available: https://www.threatadvice.com/blog/apt-attacks-targeting-government-entities.

[5] F. Okeke, “Top 6 Security Risks Associated With Industrial IoT,” TechRepublic, 23 November 2022. [Online]. Available: https://www.techrepublic.com/article/top-security-risks-industrial-iot/.

[6] Pacific Northwest National Laboratory, “Attack Scenarios Relating,” January 2020. [Online]. Available: https://www.osti.gov/servlets/purl/1814569.

[7] Ustelecom, “Botnet and IoT Security Trends 2024,” 29 March 2024. [Online]. Available: https://www.ustelecom.org/research/botnets2024/.

[8] NIST SP 800-57r5, “Recommendation for Key Management,” May 2020. [Online]. Available: https://csrc.nist.gov/pubs/sp/800/57/pt1/r5/final.

[9] NIST, “National Vulnerability Database - General Information,” [Online]. Available: https://nvd.nist.gov/general.

[10] NIST, “National Vulnerability Database - Vulnerability Status,” [Online]. Available: https://nvd.nist.gov/vuln/vulnerability-status.

[11] C. Goodman, “What is the Common Vulnerability Scoring System,” Balbix, 25 October 2024. [Online]. Available: https://www.balbix.com/insights/understanding-cvss-scores/.

[12] D. Walkowski, “MITRE ATT&CK: What It Is, How it Works, Who Uses It and Why,” F5 Labs, 10 June 2021. [Online]. Available: https://www.f5.com/labs/learning-center/mitre-attack-what-it-is-how-it-works-who-uses-it-and-why.

[13] MITRE, “MITRE ATT&CK Groups,” [Online]. Available: https://attack.mitre.org/groups/.

[14] OPC Foundation, “Unified Architecture - Landingpage,” [Online]. Available: https://opcfoundation.org/about/opc-technologies/opc-ua/.

[15] OPC Foundation, “What is OPC Classic?,” [Online]. Available: https://opcfoundation.org/faq/what-is-opc-classic/.

[16] National Instruments, “Why OPC UA Matters,” 5 December 2024. [Online]. Available: https://www.ni.com/en/solutions/industrial-machinery/smart-machine-control/why-opc-ua-matters.html.

[17] “OPC UA vs OPC Classic,” [Online]. Available: https://msnmkh.gitbook.io/opc/opc-ua-vs-opc-classic.

[18] Maximize Market Research, “OPC Server Software Market- Global Analysis and Outlook 2029,” February 2023. [Online]. Available: https://www.maximizemarketresearch.com/market-report/recent-trends-in-opc-server-software-market/146398/.

[19] Automation.com, “Microsoft announces partnership with OPC Foundation,” 28 April 2016. [Online]. Available: https://www.automation.com/en-us/articles/2016-1/microsoft-announces-partnership-with-opc-foundatio.

[20] OPC Foundation, “VDMA Robotics partners with OPC Foundation to achieve standardized information integration across robots/machines to realize requirements of Industrie4.0 machine-to-machine (M2M),” 7 March 2017. [Online]. Available: https://opcfoundation.org/news/press-releases/vdma-robotics-partners -opc-foundation-achieve-standardized-information-integration-across-robotsmachines-realize-requirements-industrie4-0-machine-machine-m2m/.

[21] OPC Foundation, “VDMA OPC Machine Vision Demonstrator at the SPS 2019,” 2019. [Online]. Available: https://opcfoundation.org/markets-collaboration/machine-vision/vdma-demo/.

[22] OPC Foundation, “ Annex B (informative)Client Server vs. Publish Subscribe,” [Online]. Available: https://reference.opcfoundation.org/Core/Part14/v104/docs/B.

[23] Microsoft, “Publisher-Subscriber pattern,” [Online]. Available: https://learn.microsoft.com/en-us/azure/architecture/patterns/publisher-subscriber.

[24] OPC Foundation, “ 3.1.6 SecurityGroup,” [Online]. Available: https://reference.opcfoundation.org/Core/Part14/v104/docs/3.1.6.

[25] OPC Foundation, “ 8.4 GetSecurityKeys Method,” [Online]. Available: https://reference.opcfoundation.org/Core/Part14/v104/docs/8.4.

[26] OPC Foundation, “ 9.1.3.3 SetSecurityKeys,” [Online]. Available: https://reference.opcfoundation.org/Core/Part14/v104/docs/9.1.3.3.

[27] OPC Foundation, “5. PubSub Concepts,” [Online]. Available: https://reference.opcfoundation.org/Core/Part14/v105/docs/5.

[28] Dragos, “Improving ICS/OT Security Perimeters with Network Segmentation,” 18 May 2022. [Online]. Available: https://www.dragos.com/blog/improving-ics-ot-security-perimeters-with-network-segmentation/.

[29] zscaler, “What Is the Purdue Model for ICS Security?,” [Online]. Available: https://www.zscaler.com/resources/security-terms-glossary/what-is-purdue-model-ics-security.

[30] H. Singh Lallie, K. Debattista and J. Bal, “A review of attack graph and attack tree visual syntax in cyber security,” *Computer Science Review,* vol. 35, 2020.

[31] S. Du and H. Zhu, Attack-Defense Tree Based Security Assessment, Springer, 2013, pp. 17-22.

[32] O. Guilles, “Securing IIoT communications using OPC UA PubSub and Trusted Platform Modules,” *Elsevier journal of Systems Architecture,* vol. Volume 134, 2023.

[33] J. Stewart, Calculus: Early Transcendentals, Cengage Learning, 2015.

[34] NIST SP 800-30, “Guide for Conducting,” September 2012. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-30r1.pdf.

[35] V. Watts, Introduction to Statistics, Fanshawe, 2022.

[36] W. Mendenhall, R. J. Beaver and M. B. Beaver, Introduction to Probability and Statistics, Cengage Learning, 2009.

[37] M. M. Islam, A. Lautenbach, C. Sandberg and T. Olovsson, “A Risk Assessment Framework for Automotive Embedded Systems,” *CPSS '16,* pp. 3-14, 2016.

[38] A. Lautenbach, M. Almgren and T. Olovsson, “Proposing HEAVENS 2.0 – an automotive risk assessment model,” *CSCS '21,* pp. 1-12, 2021.

[39] A. Amro, V. Gkioulos and S. Katsikas, “Assessing Cyber Risk in Cyber-Physical Systems Using the ATT&CK Framework,” *ACM Transactions on Privacy and Security,* vol. 26, no. 2, pp. 1-33, 2023.

[40] A. Tantawy, S. Abdelwahed, A. Erradi and K. Shaban, “Model-based risk assessment for cyber physical systems security,” *Computers & security,* vol. 96, 2020.

[41] PyPi Project, “PySide6,” [Online]. Available: https://pypi.org/project/PySide6/.

[42] NIST, “CVE API,” [Online]. Available: https://nvd.nist.gov/developers/vulnerabilities.

[43] Dragos, “VOLTZITE Espionage Operations Targeting U.S. Critical Systems,” 2024.

[44] J. Clifton, “What is Purified Water,” ReAgent, 9 September 2020. [Online]. Available: https://www.chemicals.co.uk/blog/what-is-purified-water.

[45] N. Tuptuk, P. Hazzel, J. Watson and S. Hailes, “A Systematic Review of the State of Cyber-Security in Water Systems,” *Smart Urban Water Networks,* no. Special, 2021.

[46] C. Reimer, “Wastewater Treatment Plants Automation Master plan,” City of Winnipeg, September 2012. [Online]. Available: https://legacy.winnipeg.ca/finance/findata/matmgt/documents/2012/682-2012/682-2012_Appendix_F-Automation_Master_Plan.pdf.

[47] Y. Hu, A. Yang, H. Li and Y. Sun, “A survey of intrusion detection on industrial control systems,” *International Journal of Distributed Sensor Networks,* vol. 14, 2018.

[48] Siemens, “SITRANS F Electromagnetic flowmeters Operating Instructions,” March 2024. [Online]. Available: https://cache.industry.siemens.com/dl/files/684/109810684/att_1103287/v1/A5E03063678en_SITRANS_FM_MAG5100W_OI_en-US.pdf.

[49] Siemens, “Simatic S7-200 SMART System Manual,” March 2029. [Online]. Available: https://assets.new.siemens.com/siemens/assets/api/uuid:aa045b50-b9f4-4e46-a4c4-ca882c5f00ec/s7-200-smart-system-manual-en-us.pdf.

[50] Siemens, “Simatic HMI Operating Instructions,” April 2021. [Online]. Available: https://cache.industry.siemens.com/dl/files/350/90114350/att_904652/v1/HWBasicPanels2GenUS_en-US.pdf.

[51] Siemens, “S7-1200 Programmable Controller System Manual,” 2021 May. [Online]. Available: https://cache.industry.siemens.com/dl/files/241/109797241/att_1066673/v1/s71200_system_manual_en-US_en-US.pdf.

[52] Siemens, “Process Historian 2022 System Manual,” 19 October 2022. [Online]. Available: https://support.industry.siemens.com/cs/document/109814839/process-historian-2022?dti=0&lc=en-CY.

[53] Siemens, “SIMATIC IPC1047E Operating Instructions,” March 2021. [Online]. Available: https://cache.industry.siemens.com/dl/files/099/109798099/att_1073241/v1/ipc1047e_operating_instructions_enUS_en-US.pdf.

[54] Government of Canada, “Bill C-26: An Act respecting cyber security, amending the Telecommunications Act and making consequential amendments to other Acts,” 14

December 2022. [Online]. Available: https://www.justice.gc.ca/eng/csj-sjc/pl/charter-charte/c26_1.html.

[55] J. Johnson, J. R. Hoaglund, R. D. Trevizan and T. A. Nguygen, Energy Storage Handbook, DOE Office of Electricity Energy Storage Program.

[56] D. Antoniuk, “Nearly two dozen Danish energy companies hacked through firewall bug in May,” Thbe Record, 14 November 2023. [Online]. Available: https://therecord.media/danish-energy-companies-hacked-firewall-bug.

[57] W. Toll, “FBI: Cybersecurity Threats in Renewable Energy - How a Zero Trust Approach Can Safeguard Infrastructure,” Elisity, 10 July 2024. [Online]. Available: https://www.elisity.com/blog/fbi-cybersecurity-threats-in-renewable-energy-how-a-zero-trust-approach-can-safeguard-infrastructure#:~:text=The%20result%3F,our%20rapidly%20evolving%20energy%20infrastructure.

[58] R. D. Trevizan, J. Obert, V. De Angelis, T. A. Nguyen, V. S. Rao and B. R. Chalamala, “Cyberphysical Security of Grid Battery,” *IEEE Access,* vol. 10, 2022.

[59] Synopsys, “What is a Battery Management System,” [Online]. Available: https://www.synopsys.com/glossary/what-is-a-battery-management-system.html.

[60] Twaice, “Energy Storage Management System (ESMS),” [Online]. Available: https://www.twaice.com/battery-encyclopedia/energy-storage-management-system-esms.

[61] PowerAgent, “PowerAgent™ SC3 Site Controller Installation and Operation,” 28 October 2010. [Online]. Available: https://www.alpha.com/download/critical_facilities_power/batteries/poweragent/poweragent_manual.pdf.

[62] Honeywell, “XNX Technical Manual,” September 2020. [Online]. Available: https://prod-edam.honeywell.com/content/dam/honeywell-edam/sps/his/es-mx/products/gas-and-flame-detection/documents/industrial-fixed/xnx-universal-transmitter/sps-his-xnx-technical-manual-rev-16-en-09-20.pdf.

[63] ABB, “PCS100 ESS,” [Online]. Available: https://new.abb.com/power-converters-inverters/power-converters-and-inverters/pcs100-ess.

[64] Siemens, “Sentron Power Monitoring Device Manual,” May 2019. [Online]. Available: https://cache.industry.siemens.com/dl/files/595/34261595/att_951630/v1/manual_pac4200_en-US_en-US.pdf.

[65] Schneider Electric, “Magelis Small Panels HMI STO, User Manual,” 1 December 2016. [Online]. Available: https://www.se.com/ca/en/download/document/EIO0000000239/.

[66] Government of Canada, “Occupational health and safety in federally regulated workplaces,” 29 February 2024. [Online]. Available: https://www.canada.ca/en/employment-social-development/services/health-safety/workplace-safety.html.

[67] Riverback Computing, “What is PyQt?,” [Online]. Available: https://riverbankcomputing.com/.

[68] Cole-Parmer, “The Wastewater Treatment Process,” 6 August 2024. [Online]. Available: https://www.coleparmer.ca/tech-article/eight-stages-of-wastewater-treatment-process.

[69] N. Vlajic, G. Cianfarani and R. Noce, “Data Siphoning in ICSs: Attack Tree and Role of Cryptoperiods,” in *2024 IEEE International Conference on Cyber Security and Resilience*, 2024.

[70] N. Vlajic and G. Cianfarani, “Risk-Based Methodology for Optimal Cryptoperiod Calculation in ICSs Under Data Siphoning Attack,” in *RICSS '24*, 2024.

[71] G. Cianfarani, N. Vlajic and R. Noce, “Risk-Based Optimization of Cryptoperiods to Minimize Impact of Data Siphoning Attacks on ICSs,” in *IEEE ICC*, 2025.

Appendix

Appendix A: Common Vulnerability Scoring System

The base metric group for CVSS consists of two subcategories: exploitability and impact metrics [9]. Exploitability metrics assess the ease of vulnerability exploitation, taking into account attack vectors, attack requirements, complexity, privileges required, and user interaction. Impact metrics quantify the consequences of a successful exploit, utilising the CIA triad (confidentiality, integrity, and availability) to determine its risk score. The triad itself is split into vulnerable systems and subsequent systems. An example of the difference between the two is that in a virtual environment, the host operating system is the subsequent system while the virtual machine is the vulnerable system.

The threat metric assesses how the vulnerability evolves over time with the Exploit Code Maturity metrics. Exploit Code Maturity looks at the present status of exploit techniques; are exploit techniques readily available? Does functional exploit code exist? The environmental metric involves analysing the vulnerability in the context of the system it is deployed in. To do so, the base metrics are overridden based on the organisation's environment. Modified CIA metrics are also included, allowing the organisation to calculate impacts based on what they consider to be most important. Lastly, the supplemental metric provides additional context that can be used to prioritise remediation, however, does not impact the final calculated score. It is up for the user to determine if this optional metric is to be used and the weight it has in the vulnerability calculation. The supplementary metrics range from safety to provider urgency.

All metrics are calculated on a qualitative scale. Some of the more straightforward metrics such as CIA adopt a standard "none, low, high" evaluation while others are more customised to fit the need of that particular metric. Several metrics also contain a "none" evaluation which automatically assigns the score to the highest severity.

Appendix B: Exponential Scaling Extra Analysis

In Chapter 5.1, we analyse the effect of exponential scaling for a typical scenario ($T_{CP\text{-}min}$ = 1 day and $T_{CP\text{-}max}$ = 1 year). The result in this scenario was the cryptoperiod approximately halving when risk (R_{SG}) increased by 0.1. It is important to also study how the scaling works for other possible time ranges. Figure 57 illustrates this by plotting various $T_{CP\text{-}max}$ values (8 days to 1 year) and showing the rate of change when R_{SG} Δ = 0.1. $T_{CP\text{-}min}$ is set to a constant value of 7 days.

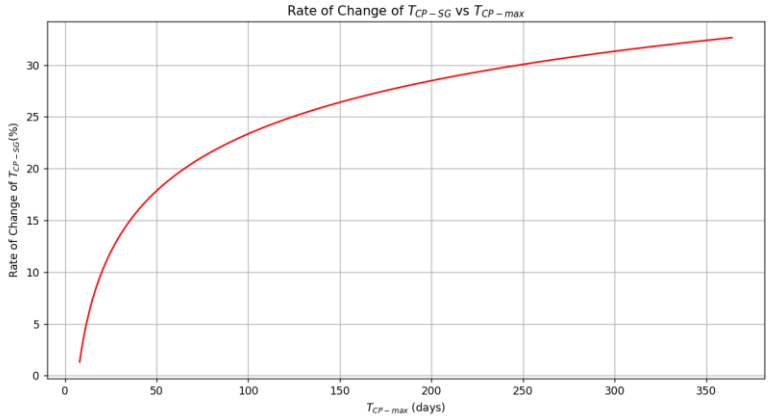


Figure 57: The rate of change as $T_{CP\text{-}max}$ increases between 8 days and 1 year ($T_{CP\text{-}min}$ = 7, R_{SG} Δ = 0.1)

When $T_{CP\text{-}max}$ is set to 8 days, the rate of change is about 2% as R_{SG} increases/decreases by 0.1. However, when the range approaches a full year, the rate of change significantly increases to 32.5%, highlighting the more pronounced effect of exponential scaling with a larger cryptoperiod range.

We can also directly compare the results for linear and exponential scaling in two different scenarios, seen in Table 18. The first example assumes a normal time range while the second drastically increases $T_{CP\text{-}min}$ to a full month.

Risk $R_{SG\text{-}KC}$	**Example 1** **Min = 1 day | Max = 1 year**		**Example 2** **Min = 1 month | Max = 1 year**	
	Linear	*Exponential*	*Linear*	*Exponential*
0	1 year	1 year	1 year	1 year
0.2	9 months 22 days	3 months 22 days	9 months 28 days	7 months
0.4	7 months 9 days	1 month 5 days	7 months 21 days	4 months 4 days

0.6	4 months 27 days	10 days 14 hours	5 months 14 days	2 months 22 days
0.8	2 months 13 days	3 days 6 hours	3 months 7 days	1 month 19 days
1	1 day	1 day	1 month	1 month

Table 18: Linear and Exponential cryptoperiod with various Risk ($R_{SG\text{-}KC}$) values

Appendix C: Qualitative Parameter Definitions

Of the parameters involved in the cryptoperiod calculation, three of them ($P_{VE\text{-}procedure\text{-}succ}$, IR_{SG} and FI_{SG}) require quantitative to qualitative mapping. This subsection will provide a general understanding of each value to more easily operate ARC-C. Note that many of these values are relative and can change based on a variety of factors.

Procedural Vulnerabilities

The following is a more detailed explanation about procedural policy values.

None (1): There are no security-related procedural policies or guidelines in place. In this scenario, the organization lacks formalized procedures to guide security practices, making it highly vulnerable to procedural attacks. A value of 1 is assigned to signify complete procedural vulnerability due to the absence of any security policies or guidelines. This represents the highest risk as a lack of procedural security provides adversaries with maximum opportunity for procedural success.

Low (0.75): Minimal security policies or guidelines. Some procedures may be in place, but they are either insufficiently detailed or inconsistently applied, leaving considerable security gaps. This value is set slightly lower than 1 to reflect that some security policies exist, but they are still minimal and largely ineffective against procedural threats. The risk remains very high, as inadequate security policies provide limited mitigation.

Medium (0.45): Security policies are established and moderately enforced. While they may cover essential procedures, there may be gaps in coverage or enforcement, reducing overall procedural security. With moderately strong policies in place, the probability of procedural success for an adversary decreases. However, a 0.45 value reflects the residual risk from possible enforcement gaps or weaknesses in the policies.

High (0.1): Robust, well-documented, and enforced security policies and guidelines exist. The organization has comprehensive procedural controls in place, which significantly reduce vulnerability to procedural attacks. Rather than assigning this category a value of 0, 0.1 is chosen to acknowledge the inherent uncertainties. Even with strong policies, there is always a small risk due to factors like human error or unforeseen vulnerabilities, hence a low but non-zero value is assigned.

Information Rate

The following is a more detailed explanation about information rate map values.

In this context, Low, Medium, and High values for both the number of publishers (N_p) and the average transmission rate per publisher (T_P) are relative to the specific environment and use case. For instance, a "High" rate in a small environment may be considered "Medium" in a larger one, depending on baseline network and security demands.

The following explanations cover the quantitative mappings:

Very Low (0.1): This value is only calculated when there is a low number of publishers and low transmission rate and presents the least risk. Fewer publishers mean fewer potential points of data leakage, and a low information rate means there is little data to exploit, even if accessed. The score of 0.1 represents a scenario where exposure is minimal, and the chance of significant Data Siphoning is very low.

Low (0.3): The risk level increases slightly due to either a higher data rate or a larger number of publishers, though each remains moderate. The configuration still limits the exposure as there are either few publishers or low data flow. The score of 0.3 reflects a controlled environment where the risk is higher than the minimum, but not substantial enough to be categorized as moderate.

Moderate (0.75): When either the number of publishers or the data rate reaches a moderate level, the risk level is elevated to "Moderate." In this range, there is a balanced but notable potential for data exposure, as the environment has a combination of increased access points or higher data flow. The 0.75 score reflects a scenario with moderate vulnerability, where Data Siphoning could be impactful but not maximally severe.

High (0.9): Used when both dimensions (publisher count and data rate) are noticeably high. This configuration indicates significant risk, as it provides substantial data flow with a considerable number of publishers. The 0.9 score represents a heightened vulnerability where the potential for data exposure is considerable, requiring strong security measures to mitigate risk.

Very High (1): Reserved for scenario in which both the number of publishers and the data rate reach maximum levels. This setup presents the highest risk, as it combines multiple access points with extensive data flow, creating an environment highly susceptible to Data Siphoning. The 1.0 score reflects maximum vulnerability, indicating that any compromise could result in significant data exposure with severe consequences.

Functional Impact

The following is a more detailed explanation about functional impact values.

General rationale for qualitative values:

None (0): This value represents no expected impact on the functional objective if SG data is compromised. It assumes that the data in question does not directly or indirectly affect this specific function.

Low (0.3): Signifies a minor impact, where a security breach would cause negligible disruption or risk. A value of 0.3 was chosen because it provides a notable but limited contribution to the overall risk calculation.

Medium (0.6): "Medium" reflects a moderate impact, where a data breach could partially disrupt the function or require significant mitigation efforts. While not immediately critical, a breach would nonetheless have noticeable consequences. The choice of 0.6 ensures that moderate impacts are weighted appropriately in the overall calculation, allowing for risk accumulation without overly skewing the final impact score.

High (1): This level suggests a substantial impact. A compromise could considerably impair the function, necessitating urgent measures. The use of 1 here represents the upper boundary of impact in typical situations, where a single category's failure could already pose significant threats but might not independently cause catastrophic failure.

Critical (2): A "Critical" rating indicates that a compromise would severely disrupt or even incapacitate the function. Per specifications described in Chapter 5.3.2, if both the Functional Impact (FI_{SG}) and criticality ($FO_{criticality}$) are marked as "Critical," the overall impact score becomes 1 immediately.

Each function—Operational, Safety, Financial, and Privacy & Legislative—has its own unique requirements and thresholds for these qualitative values, detailed below.

1. Operational Objectives (FO(1))

None: No anticipated impact on day-to-day operations if SG data is compromised, as this data is either not used in operational workflows or is redundant.
Low: A minor disruption to operations, potentially slowing down processes but not compromising the core functionality.

Medium: A compromise could affect operational performance moderately, causing delays or minor misuse but allowing overall functionality to continue.
High: A significant risk to operational function, where misuse or system manipulation could disrupt critical day-to-day tasks.
Critical: Total or near-total operational shutdown due to compromised SG data, affecting essential system functions.

2. Safety Objectives (FO(2))

None: SG data has no effect on safety, meaning a breach does not endanger people or the environment.
Low: Minimal safety implications; any impact is likely manageable and doesn't pose a threat to personnel or the public.
Medium: A moderate safety risk, with compromised data possibly leading to conditions that could harm employees or cause environmental damage.
High: Severe safety concerns, where compromised SG data could endanger lives or cause significant environmental harm.
Critical: Compromised data creates a critical safety hazard, with immediate risks to health, life, and the environment, requiring emergency intervention.

3. Financial Objectives (FO(3))

None: No anticipated financial loss or risk if SG data is exposed, as this data has no monetary implications.
Low: Minor financial impact, with potential for small losses that do not affect overall profitability or long-term revenue.
Medium: Moderate financial risk; a data breach may lead to significant repair costs or some reputational harm.
High: Major financial implications, possibly including direct revenue loss, reputational damage, and substantial repair or replacement costs.
Critical: Severe financial consequences, with compromised data leading to long-term revenue losses, extensive repair costs, and reputational damage that could impair future earnings.

4. Privacy and Legislative Objectives (FO(4))

None: No compliance or privacy concerns associated with this SG data, so a breach has no regulatory implications.

Low: Limited regulatory impact, with only minor privacy or compliance concerns if data is compromised.

Medium: Moderate impact on privacy and compliance, where a breach could attract regulatory attention and require mitigation.

High: Serious privacy or compliance risk, with the potential for regulatory penalties or significant privacy violations.

Critical: A critical compliance breach, leading to severe legal consequences, substantial fines, or reputational harm due to privacy violations or regulatory non-compliance.

Appendix D: ARC-C Tool Code Structure

In Chapter 6, we discussed the UI component of the ARC-C tool. Here, we will delve into the code itself and outline certain design decisions that were made. The code follows a simple structure and reflects the UI design. Figure 58 illustrates the structure consisting of five separate files organized in a hierarchical manner.

main.py serves as the entry point at the top level, coordinating the flow of the program. This handles the main UI window and is responsible for retrieving the formula variables obtained from the underlying processes, calculating the final cryptoperiod, and displaying the result to the user. Below it, two primary paths are depicted: one for importing devices and another for handling environmental variables. These are responsible for the respective functions discussed in Section 6.1.

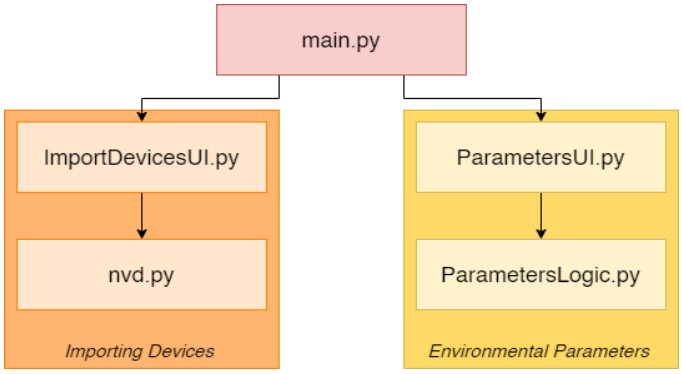


Figure 58: Hierarchy of ARC-C

Importing Devices

main.py imports two functions from ImportDevicesUI.py: SetupImportDevices and GetImportValues. SetupImportDevices contains the logic pertaining to the UI elements and how they display on the main window. A single container argument is taken as an input which is then assigned as the root window for the remaining UI elements.

```
def SetupImportDevices(container):
    frame = QFrame(container)
    frame.setFrameShape(QFrame.Box)
    frame.setLineWidth(1)

    main_layout = QVBoxLayout(frame)
…
```

Figure 59: Initial code for SetupImportDevices

GetImportValues calculates $P_{VE\text{-}software\text{-}succ}$ before returning it to the main method. However, before the calculation can be completed, the device information inputted by the user must be processed. To do so, we utilise three functions from nvd.py: searchPLCInforNVD, getExploitabilityScoreCVE, and getConfidentialityImpactCVE. These functions facilitate communication with the NVD API and populate the “Imported Devices” list. The list itself stores all the information required (i.e., device name, CVEs for associated devices, exploitability score and impact for each associated CVE) for the calculation.

```
def GetImportValues():

…

    b_d = 0.03
    c_w = 2

    overallResilience = 1
    for cpe, cves in activeDeviceInfoList:
        deviceResilience = 1

        for (cve_info, _) in cves:
            exploit = cve_info[1]
            impact = cve_info[2]

            if impact == 'NONE':
                quantizedExploit = 0.1 * (exploit / 3.9) ** c_w
            elif impact == 'LOW':
                quantizedExploit = 0.3 * (exploit / 3.9) ** c_w
            elif impact == 'HIGH':
                quantizedExploit = 1.0 * (exploit / 3.9) ** c_w

            deviceResilience *= 1 - quantizedExploit

        deviceCompromise = b_d + (1 - b_d) * (1 - deviceResilience)
        overallResilience *= 1 - deviceCompromise

    totalCompromise = 1 - overallResilience
    update_pve_alt_button(totalCompromise)

    return totalCompromise, True
```

Figure 60: Code for GetImportDevices

Environmental Parameters

To discuss how the UI elements in the environmental parameters subsection are defined, we will work bottom up, starting with ParametersLogic.py. Of the four unique formula calculations which occur in this section; time range ($T_{CP\text{-}min}$ and $T_{CP\text{-}max}$), security policy ($P_{VE\text{-}proc\text{-}succ}$), data rate ($IR_{SG}$), and functional impact (AI_{SG}) - three of them utilise segmented control. Segmented control was chosen as it presents a small number of discrete options. When one of the segments is selected, it visually changes to indicate the choice, making it clear and easy for the user to see which option is active.

While segmented control is used for three calculations, its implementation varies for each. Security policy ($P_{VE\text{-}procedure\text{-}succ}$) is the simplest, requiring a single row of selectable buttons. Information Rate (IR_{SG}) is slightly more complicated with two distinct rows. Lastly, Functional Impact (FI_{SG}) is the most involved with not only one row for each of the four Functional Categories, but it must also support two separate groups (Importance and Extent) as well as potential subcategories. Our implementation was constructed in a way where all these features are available by the way the input is initialized.

```
def CreateLayout(mainLayout, severityList, categoryList, numButtonGroups):
    buttonGroupsList = [{} for _ in range(numButtonGroups)]  # Create a list of
empty dictionaries for button groups
    activeItems = CatLayout(mainLayout, buttonGroupsList, categoryList,
severityList)
    return buttonGroupsList, activeItems
```

Figure 61: Code for CreateLayout

The CreateLayout function serves as the interface between ParametersLogic.py and ParametersUI.py. It manages the spatial arrangement of UI components, ensuring that everything from button groups to labels is positioned according to predefined logic. The first argument, mainLayout, indicates where to map the UI elements within the program (i.e. specifies the parent layout for all UI elements). The remaining three arguments specify the variables for the segmented control.

- severityList: Details different severity levels, each with a label, value, and color
- categoryList: Outlines the categories and any optional subcategories for UI representation
- numButtonGroups: Specifies the number of button groups required (e.g., FI_{SG} requires two button groups, one for importance and one for extent).

To see how to initialize a segmented control, we move up to ParametersUI.py. Creating a layout requires calling CreateGenericLayout which processes the data and handles updating UI elements to send to the original CreateLayout. It provides a generic template for creating layouts reusable across different parts of the application and generates a complete UI frame based on the provided parameters. These functions are activated through specific category functions: ImpactCategories, DataCategories and PolicyCategories. These category functions define the necessary parameters for CreateGenericLayout and thereby instantiates the layouts.

In the example in Figure 62, we initialize the data rate segmented control by defining the required values in DataCategories() and passing them to CreateGenericLayout. This process is also taken for impact and security policy.

```
def CreateGenericLayout(severityList, categoryList, numButtonGroups, updateFunc,
defaultColor, tooltips, createResultButton): ⋯
return uiFrame

def DataCategories():
    severityList = [("Low", 1, low), ("Medium", 2, medium), ("High", 3, high)]
    categoryList = ['Data Rate', 'Number of Publishers']
    tooltips = {"Data Rate": "Impact based on data rate", "Number of Publishers":
"Impact based on number of publishers"}

    return CreateGenericLayout(severityList, categoryList, 1, UpdateDataLayout,
"#90EE90", tooltips, True)
```

Figure 62: Function definition for CreateGenericLayout and full DataCategories initialization

Creating the time range UI elements is independent and requires simple initialization of the elements as well as keeping track of the min and max values.

To integrate the features from ParametersUI.py into main.py, two functions (SetupTopRight and SetupImpact) are defined. Both functions are designed to receive a specific container or frame as an argument, controlled by main.py. The two functions represent two different sections of the UI: the section that runs parallel to the import devices UI element (containing the time range, security policy, and data rate elements), and the section at the bottom which handles impact (as seen in Figure 63).

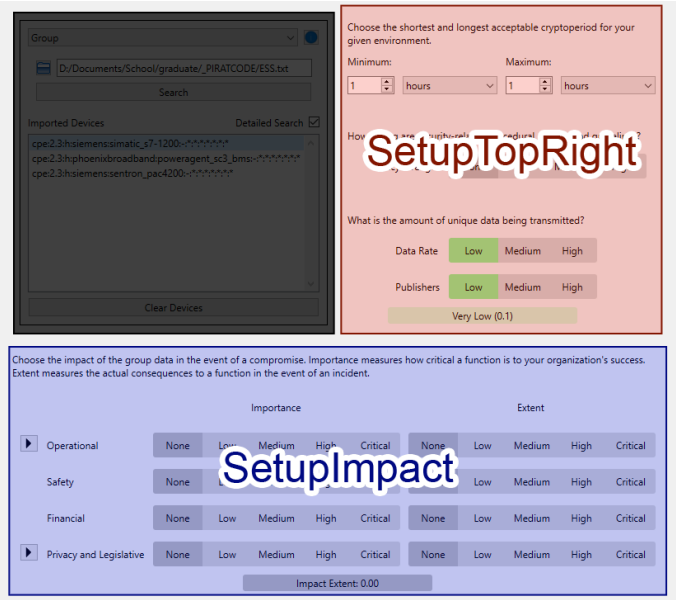


Figure 63: Functions imported to main.py and their associated UI groups

SetupImpact is relatively straightforward as it only contains a single element group, but SetupTopRight acts as an intermediary step which consolidates the individual UI elements into a single, larger element for ease of use. We delegate the time range, security policy, and data rate elements relative to a single frame and then allow main.py to position the frame in the main UI. This allows for a modular design, making it easier to manage and update specific UI sections without affecting the overall layout.

Lastly, main.py imports a “values” object from ParametersUI.py which contains the calculated segmented control values - impact extent, data rate, and security policy respectively. During initialization, each value is set to its default value from the UI.

```
class Values:
    impact = 0
    data = 0.1
    policy = 1

values = Values()
```

Figure 64: Values object defined in ParametersUI.py and imported by main.py

Main Method

Besides handling UI placement for all the different elements, the main method hosts the logic for the overall cryptoperiod calculation. Based on the formula outlined in Chapter 5, the final cryptoperiod length is calculated based on: deviceProbability ($P_{VE\text{-}software\text{-}succ}$), values.policy ($P_{VE\text{-}procedure\text{-}succ}$), values.impact ($FI_{SG}$) and values.data ($IR_{SG}$).

```
def ShowResults():

    deviceProbability, isNotEmpty = GetImportValues()
    timeRange1, timeRange2 = updateTimeDifference()

    if timeRange1 > timeRange2:
        showCryptoperiodWarning()
        return

    if not EmptyImport(isNotEmpty): # Create warning pop-up if no devices are
present
        return

    if deviceProbability == 0: #if no devices, assume worst case scenario
      deviceProbability = 1

    probability = 1 - ((1 - deviceProbability)*(1 - values.policy))

    weight = 1 - (values.impact ** (1/3))
    impact = values.impact * values.data**weight

    finalRisk = (probability * impact)

    cryptoperiod = timeRange1 * (timeRange2 / timeRange1)**(1-finalRisk)
    cryptoperiod_display = "Old: " + displayTimeDifference(cryptoperiod)

    if results_text_box:
        results_text_box.setText(f" {cryptoperiod_display}")
```

Figure 65: ShowResults() method of main.py, handling the calculation and display of the final cryptoperiod

The resultant cryptoperiod is then converted to a format which can be easily displayed and read.

Glossary

Term	Definition
Advanced Persistent Threat **(APT)**	A highly sophisticated, stealthy cyber threat typically carried out by well-resourced adversaries (i.e., nation states) to gain long-term access to a target system, often for espionage or sabotage
Adversarial Tactics, Techniques, and Common Knowledge **(ATT&CK)**	Framework which provides valuable insights into the tactics, techniques, and procedures (TTPs) used by real-world cyber adversaries during different stages of an attack
Attack Tree	A structured, hierarchical diagram used to model potential attack scenarios against a system
Common Vulnerabilities and Exposures **(CVE)**	Unique identifiers assigned to each of the known/recorded vulnerabilities
Common Vulnerability Scoring System **(CVSS)**	Framework for the quantitative and qualitative rating of a vulnerability's severity
Data Siphoning **(DS)**	The complete process an adversary undertakes to collect and exfiltrate sensitive in-transit data, including obtaining a decryption key
Industrial Control Systems **(ICS)**	A broad term for integrated hardware and software systems used to manage industrial processes
Information Technology **(IT)**	Systems focused on data processing, storage, and communication, primarily in business and enterprise environments
National Vulnerability Database **(NVD)**	Database maintained by NIST which contains a list of known CVEs
Open Platform Communication Unified Architecture **(OPC UA)**	Platform independent service-oriented communication standard for data exchange
Operational Technology **(OT)**	Systems that monitor and control physical processes and infrastructure, commonly used in industrial and critical infrastructure sectors
PubSub	A communication architecture used for scalable and decoupled data exchange
PURDUE	Reference model which devices industrial networks into levels which service specific purposes
Security Key Service **(SKS)**	Central control for OPC UA keys