TECHNIQUES FOR DEVELOPING SECURE-BY-DESIGN INDUSTRIAL CONTROL SOFTWARE

A THESIS SUBMITTED TO AUCKLAND UNIVERSITY OF TECHNOLOGY
IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Supervisor
Associate Professor Roopak Sinha
Professor Stephen G. Macdonell
Dr. Matthew M. Y. Kuo

July 2021

By
Awais Tanveer
School of Engineering, Computer and Mathematical Sciences

Abstract

Industrial Control Systems (ICS) have significantly changed how we operate manufacturing plants, transportation systems, facility management and monitoring systems. Reliance on modern ICS has significantly improved human lives but with a strong caveat that their failure can sabotage and reset our technological journey. As a result, security has become a key concern in ICS that run highly distributed software applications deployed on resource-constrained devices. The complex requirements of highly distributed and heterogeneous ICS make them challenging to design and manage. Current literature shows that most state-of-the-art approaches deal with validation and verification of security requirements in ICS, lacking meaningful traceability across the system development life-cycle of ICS. Security standards provide a consolidated set of security requirements that are rigorously validated by the practitioners and academia. ICS security standard certified products ensure that the product has met the security requirements and has been subject to rigorous testing. A common approach to requirements traceability through design and implementation can help to improve the time-to-market of the product since the vendors and certifiers aim to validate and verify the same system. However, the lack of design guidelines for certifying ICS software is detrimental for stakeholders to follow a standard process in ICS certification. IEC 61499 is an emerging standard for developing distributed ICS applications. IEC 61499 can also be used for ICS security components by automated security requirements mapping techniques to help map security standards requirements link with the ICS

design. However, the current secure-by-design approaches for ICS are not entirely aligned with security requirements engineering and security standards.

This research aims to fulfil the current gaps regarding integrating ICS security requirements engineering, secure-by-design model and security implementation approaches to provide end-to-end traceability. The significant contributions of this research include a novel "Secure Links" technique to develop secure-by-design IEC 61499 ICS applications in a uniform, consistent, maintainable, traceable and reusable manner. Another notable contribution of this research is a novel model of a security requirements repository that stores labelled property graphs for cybersecurity requirements specifications and IEC 62433-4-2 security requirements in multiple partitions while emphasising requirements structure and relationships. At the same time, this research also contributes to fulfilling security goals such as confidentiality, availability and integrity in IEC 61499 ICS applications. The integration of the contributing artefacts such as secure links and security requirements repository forms a comprehensive solution for end-to-end traceability of security standard requirements to produce certified ICS software. Experimental results show that the secure links significantly reduce design and code complexity while also improving application maintainability and requirements traceability. This research also demonstrates the construction of the requirements traceability matrix that emerges as a natural consequence of using a labelled property graph repository.

Contents

Abstract **2**

Attestation of Authorship **13**

Publications **14**

Acknowledgements **16**

1 Introduction **17**
1.1 Background . . . 17
1.2 Research Questions . . . 21
1.3 Contributions . . . 22
1.4 Research Design . . . 24
1.4.1 Problem Identification . . . 26
1.4.2 Design, Development and Evaluation . . . 28
1.5 Thesis Organisation . . . 31

2 Literature Review **32**
2.1 Introduction . . . 32
2.2 Systematic Mapping Study . . . 33
2.2.1 Introduction . . . 34
2.2.2 Background and Related Surveys . . . 37
2.2.2.1 Formal Methods . . . 37
2.2.2.2 Use of Formal Methods in Industry . . . 38
2.2.3 Systematic Mapping Study Methodology . . . 41
2.2.4 Scoping and Formulating Research Questions . . . 41
2.2.4.1 Searching . . . 43
2.2.4.2 Data Extraction . . . 47
2.2.5 Classification of Distributed Embedded Systems . . . 48
2.2.6 Formal Requirements Modeling . . . 50
2.2.6.1 Formal Requirements Modelling Techniques . . . 51
2.2.6.1.1 Logic Based Techniques . . . 51
2.2.6.1.2 Algebra Based Techniques . . . 53
2.2.6.1.3 Automata . . . 54

2.2.6.1.4 Graph Theory . . . 54
2.2.6.1.5 Petri Nets . . . 55
2.2.6.1.6 Semi-Formal Techniques . . . 56
2.2.6.1.7 Ontology . . . 57
2.2.6.2 Formal Requirements Specification Languages . . . 58
2.2.6.2.1 Architecture Description Languages . . . 58
2.2.6.2.2 Temporal Logic . . . 59
2.2.6.2.3 Communicating Sequential Processes . . . 60
2.2.6.2.4 B Method and Z Notation . . . 61
2.2.6.2.5 Domain Specific Languages . . . 62
2.2.7 Classification based on Requirement Types and RE phases . . 62
2.2.7.1 Classification Based on NFR Types . . . 63
2.2.7.1.1 Safety Requirements . . . 64
2.2.7.1.2 Reliability . . . 65
2.2.7.1.3 Security . . . 65
2.2.7.2 Classification Based on RE Phases . . . 66
2.2.8 Bibliographical Mapping . . . 68
2.2.9 Discussions . . . 72
2.2.9.1 Lack of Industrial Application of Formal Specification Languages . . . 74
2.2.9.2 Concerns on Formal Security Requirements Engineering in DES . . . 74
2.2.9.3 Emergence of Novel Formal Frameworks in IoT . . . 77
2.2.9.4 Shortfall in Sensor Networks Research . . . 78
2.2.9.5 Emphasis on Formalisation in Automotive industry 78
2.2.10 Threats to Validity . . . 79
2.2.11 Conclusion . . . 81
2.3 Survey on Security Engineering in ICS . . . 82
2.3.1 ICS Security Requirements Engineering . . . 82
2.3.2 ICS Security Standards . . . 86
2.3.2.1 Current Works in Security Standard Compliance . . . 91
2.3.3 ICS Secure Design and Implementation . . . 94
2.4 Conclusion . . . 96

3 Security Goals: Confidentiality in ICS Applications 99
3.1 Prelude . . . 99
3.2 Abstract . . . 100
3.3 Introduction . . . 100
3.4 Background and Literature Review . . . 103
3.4.1 Industrial Automation and Control Systems . . . 103
3.4.2 IEC 61499 . . . 104
3.4.3 IACS and Security . . . 104
3.4.4 Security in IEC 61499 based Applications . . . 107
3.5 Secure SIFBs for IEC 61499 Applications . . . 109

3.5.1 Secure Links for Distributed IEC 61499 based Applications . 110
3.5.2 Confidentiality Layer for Function Blocks (CL4FB) 112
3.5.3 (SI)FBs for the CL4FB 113
3.5.4 Secure Key Exchange 116
3.6 Case Study: Secure Smart-Grid Protection 118
3.7 Feasibility Analysis 122
3.8 Conclusions 126

4 Security Goals: Availability in ICS Applications 127
4.1 Prelude 127
4.2 Abstract 127
4.3 Introduction 128
4.4 Availability Attacks on IEC 61499 Applications 131
4.4.1 Attack with Malicious or Malformed Data 133
4.4.2 Application Level Flood Attack 134
4.4.3 Device Level Flood Attack 137
4.5 An SIFB based IDPS 138
4.5.0.1 Active Security Protection 142
4.6 Implementation and Results 142
4.7 Conclusions and Future Work 145

5 Secure Links: Secure-by-Design Communications 146
5.1 Prelude 146
5.2 Abstract 146
5.3 Introduction 147
5.4 Related Works 152
5.5 Secure Links for Application Design 156
5.5.1 Security Requirements Repository 156
5.5.2 Secure Links 157
5.6 Compiling Secure Links into IEC 61499-compliant Software 163
5.7 Experimental Results 165
5.8 Conclusions 175

6 Security Requirements Repository 176
6.1 Prelude 176
6.2 Abstract 177
6.3 Introduction 177
6.4 Overview of the Proposed Solution 180
6.5 Background 184
6.6 Related Works 186
6.7 Extending IEC 62443-4-2 Requirements Structure for Standard Implementations 189
6.8 Security Requirements LPGs 193
6.8.1 IEC 62443-4-2 Property Graph 193

6.8.2 CSRS Property Graph . . . 195
6.9 Security Requirements Repository . . . 196
6.9.1 Repository Architecture . . . 197
6.9.2 Repository Implementation . . . 200
6.10 Design-time Tool Integration . . . 202
6.11 Results and Discussions . . . 206
6.11.1 Requirements Traceability for Industrial Control Systems . . . 206
6.11.2 Requirements Change Management . . . 209
6.11.3 Requirement Traversal and Graph Analytics . . . 211
6.11.4 Requirements Extraction . . . 212
6.11.4.1 Manual Requirement Extraction . . . 213
6.11.4.2 Requirement Extraction using LPG . . . 214
6.11.5 Repository Reusability . . . 215
6.12 Limitations and Validity Threats . . . 216
6.12.1 Reliance on IEC 61499 . . . 217
6.12.2 Scope Limitations . . . 217
6.12.3 Scalability . . . 218
6.12.4 Error-prone Requirements Elicitation and Extraction . . . 218
6.13 Conclusion and Future works . . . 219

7 Discussions, Conclusions and Future Works 221
7.1 Introduction . . . 221
7.2 Integrated Techniques for Secure-by-Design ICS Software (ITSIS) . . . 223
7.2.1 Security Requirements Engineering . . . 227
7.2.2 IEC 61449 Application Design . . . 229
7.2.3 IEC 61499 Application Implementation . . . 230
7.2.4 Traceability via TORUS — Verification and Validation . . . 232
7.3 Results Synopsis . . . 234
7.3.1 System Complexity . . . 234
7.3.2 Scalability . . . 235
7.3.3 Maintainability of IEC 61499 Applications . . . 236
7.3.4 Requirements Traceability via TORUS . . . 237
7.4 Implications on Security Goals . . . 238
7.4.1 Confidentiality . . . 238
7.4.1.1 Abstract Function Block Interface . . . 239
7.4.1.2 Data-in-Transit Encryption . . . 240
7.4.1.3 Data-at-Rest Encryption . . . 241
7.4.1.4 Limitations . . . 242
7.4.1.4.1 Key Distribution . . . 242
7.4.1.4.2 Asymmetric Key Cryptography . . . 242
7.4.2 Availability . . . 242
7.4.2.1 Machine Learning IDPS . . . 243
7.4.2.2 Lightweight IDPS . . . 245
7.4.2.3 Limitations . . . 246

7.4.3 Integrity . . . 247
7.4.3.1 Abstract Function Block Interface . . . 249
7.4.3.2 Open-ended Secure Communication . . . 249
7.4.3.3 Lightweight Hash Functions . . . 250
7.4.3.4 Limitations and Future Works . . . 250
7.5 S-Lib: Security Function Block Library . . . 251
7.5.0.1 Usage in Secure Links . . . 252
7.5.1 S-Lib Architectural View . . . 253
7.5.2 S-Lib: An Open Source IEC 61499 Crypto Library . . . 256
7.5.2.1 Integration with Open Source IDEs and Runtimes . . . 256
7.5.2.2 Community Collaboration . . . 258
7.5.2.3 Scalability . . . 259
7.5.2.3.1 Load Scalability . . . 259
7.5.2.3.2 Functional Scalability . . . 259
7.6 ITSIS Implications . . . 260
7.6.1 ICS Security Certification . . . 260
7.6.1.1 Vendors . . . 262
7.6.1.2 Certification Lab . . . 263
7.6.1.3 Community . . . 264
7.6.2 An Open-Source Compatible Method . . . 265
7.7 Conclusions . . . 267
7.7.1 Limitations and Future Works . . . 268
7.7.1.1 IEC 61499 Bound Design and Development . . . 268
7.7.1.2 Current Limitations on Scope and Scalability . . . 270
7.7.1.3 Industrial Evaluation . . . 272

References **274**

Appendices **292**

A Systematic Mapping Study—Primary Studies **293**

List of Tables

2.1 Framework Rubric for Data Extraction and Classification 47
2.2 Semi-formal techniques in primary studies 57
2.3 Security standards relevant to ICS 90

3.1 Latency of FBs for the proposed CL4FB 124

5.1 Complexities for applications and security mechanisms 168
5.2 Pre-compilation program complexity and maintainability 170
5.3 Latency comparison between lightweight and TLS security mechanisms implementations. 174

6.1 Wind turbine security requirements 182
6.2 A sample composition of security requirements in IEC 62443-4-2 . . . 184
6.3 TORUS splices linking requirements and secure links for the wind turbine system. RID and SLID refers to requirement and secure link id respectively. SL (Secure Link) 207
6.4 Requirements traceability matrix based on TORUS splices. (PR - CSRS Product Requirement, SR - Standard Requirement, RE - Requirement Enhancement, FBN - Function Block Network, SL - Secure Link) . . 208

7.1 ITSIS consistency with other development methodologies 226
7.2 Lightweight hash functions suitable for IEC 61499 implementation . . 251

List of Figures

1.1 A general configuration of ICS 18
1.2 Research methodology and design 25

2.1 Research scope. 42
2.2 Search Execution. Processes in green boxes represent systematic mapping process by Petersen, Feldt, Mujtaba and Mattsson (2008). 45
2.3 Number of studies by DES types 48
2.4 Formal methods used for requirement modeling 52
2.5 Types of logics used 53
2.6 Formal specification languages 60
2.7 Distribution of targeted requirements by types in primary studies. . . . 63
2.8 Distribution of Non-Functional requirements 64
2.9 Classifications of works targeting safety requirements in DES 65
2.10 Classifications of works according to RE phases 67
2.11 Articles published by year 69
2.12 Articles published by venue type 70
2.13 Number of articles by research type 71
2.14 Research methods and experimental setups 72
2.15 Multiple views of a heat map showing the concentration of primary studies. 73

3.1 IEC 61499 application FB distribution concept 110
3.2 A secure FB data link with annotation 111
3.3 Proposed Security layer in an IEC 61499 distributed application 113
3.4 An internal view of IEC 61499 distributed application with CL4FB . . 114
3.5 CL4FB event and data interfaces: (a) FB for encryption services (b) FB for decryption services (c) FB for Key expansion process (d) Proposed Secure KE SIFB 115
3.6 An all-inclusive model of CL4FB with key exchange 117
3.7 An IEC 61499 system for implementing protection functions 119
3.8 Implementation of CL4FB for protection functions 120
3.9 CLSender CFB internal FB network 121
3.10 CLRecv CFB internal FB network 122
3.11 Experimental setup 123

4.1 Illustration of scenario in which the proposed SIFB has been implemented. 132
4.2 IEC 61499 implementation of case study in Figure 4.1 133
4.3 A subset of attack scenarios of IEC 61499 distributed applications . . 135
4.4 A simple network configuration containing IDPS 138
4.5 Proposed configuration of IDPS SIFB in IEC 61499 distributed applications 139
4.6 A CFB containing IDPS_SIFB and associated ALERTCHECK FB. . . 140
4.7 Application of `IDPS_CFB` with `LiftCtl` FB. 141
4.8 Experimental set up with Wago 750-8206 PFC200. 143
4.9 Snort based `IDPS_CFB` detecting attacks. 144

5.1 Industrial Mixer Control System. 149
5.2 An overview of the SLDM 151
5.3 A TORUS based CSRS repository 157
5.4 IEC 61499 implementation of IMCS including secure links. The blocks are coloured *wrt* to device mapping. 160
5.5 A send-receive security mechanism for lightweight hashing. 161
5.6 Fully IEC 61499-compliant IMCS application generated by Alg. 1 . . 166
5.7 Tracing requirements from security standards to final implementation using TORUS 167
5.8 Cumulative effect on the design complexity of IMCS and BHS after secure links are processed by algorithm 1. 169
5.9 Manual addition of security mechanisms vs the use of secure links: cumulative program difficulty (D) and cyclomatic complexity (V) comparisons. 172
5.10 Maintainability of IMCS & BHS with and without secure links. 173

6.1 Process overview of creating an LPG security requirements repository and its association with secure-by-design and traceability tools. RId and SLId in stage 3 refer to requirement and secure link id respectively, while TORUS is a requirement traceability tool. 181
6.2 An example of IEC 62443-4-2 SL-C mapping 185
6.3 IEC 62443-4-2 Extended requirements structure based on formalism described in Definition 4 192
6.4 IEC 62443-4-2 Integrity property graph produced by Neo4j 194
6.5 A CSRS graph of a wind turbine system created in Neo4j 197
6.6 Process for storing and integrating CSRS and security standard graph 198
6.7 Repository logical partitions 199
6.8 A partial resulting graph from the execution of Cypher query in Listing 4. 202
6.9 An example of TORUS and secure links working with the repository for the wind turbine requirements 203
6.10 A secure links plugin developed using 4diac IDE 205
6.11 Requirements change process using security requirements repository and design-time tools. 210

6.12 Manual requirements extraction 213
6.13 Requirements extraction based on security level using LPGs 214

7.1 An overview of ITSIS 224
7.2 ITSIS verification and validation approach. 233
7.3 An ML-based IDPS proposal for IEC 61499 applications 244
7.4 A conceptual model of IEC 61499 HIDS based on snapshotter 246
7.5 IEC 61499 keyed-hash mechanism providing data integrity (Copied from Figure 5.5) 248
7.6 IEC 61499 security library architecture 254
7.7 An overview of the SLDM (copied from Figure 5.2) 256
7.8 Current S-Lib implementation in 4diac IDE and 4diac FORTE 257
7.9 ITSIS utilisation in certification process 262

Attestation of Authorship

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the qualification of any other degree or diploma of a university or other institution of higher learning.

Signature of candidate

Publications

Publication 1 (Chapter 3), Researcher's Contribution: 90%

Tanveer, A., Sinha, R., & MacDonell, S. G. (2018, July). On Design-time Security in IEC 61499 Systems: Conceptualisation, Implementation, and Feasibility. In 2018 IEEE 16th International Conference on Industrial Informatics (INDIN) (pp. 778-785). IEEE.

Co-authors:

Roopak Sinha _____ Stephen G. MacDonell _____

Publication 2 (Chapter 4), Researcher's Contribution: 80%

Tanveer, A., Sinha, R., MacDonell, S. G., Leitao, P., & Vyatkin, V. (2019, July). Designing Actively Secure, Highly Available Industrial Automation Applications. In 2019 IEEE 17th International Conference on Industrial Informatics (INDIN) (Vol. 1, pp. 374-379). IEEE.

Co-authors:

Roopak Sinha _____ Stephen G. MacDonell _____

Paulo Leitao _____ Valeriy Vyatkin _____

Publication 3 (Chapter 5), Researcher's Contribution: 90%

Tanveer, A., Sinha, R., & Kuo, M. M. (2020). Secure Links: Secure-by-Design Communications in IEC 61499 Industrial Control Applications. IEEE Transactions on Industrial Informatics.

Co-authors:

Roopak Sinha _____ Matthew Kuo _____

Publication 4 (Chapter 6), Researcher's Contribution: 83%

Tanveer, A., Sharma, C., Sinha, R., & Kuo, M. M. (2021). Tracing Security Requirements in Industrial Control Systems using Graph Databases — Submitted in Springer Journal of Software and Systems Modeling.

Co-authors:

Chandan Sharma _____ Roopak Sinha _____ Matthew Kuo _____

Publication 5, Researcher's Contribution: 12%

Zahid, F., Tanveer, A., Kuo, M. M., & Sinha, R. (2021). A systematic mapping of semi-formal and formal methods in requirements engineering of industrial Cyber-Physical systems. Journal of Intelligent Manufacturing, 1-36.

Main author:

Farzana Zahid _____

Co-authors:

Roopak Sinha _____ Matthew Kuo _____

Acknowledgements

First and foremost, I would like to thank God Almighty for giving me the strength, knowledge, ability and opportunity to undertake this research study.

I dedicate my work to my family, especially my lovely wife, Yasmeen. A special feeling of gratitude to my loving parents, Tanveer Ahmed and Zubaida Bibi, and my siblings, Zeeshan and Madiha, whose words of encouragement made me stay motivated.

In the end, I would like to say special thanks to my supervisors Associate Professor Roopak Sinha, Professor Stephen G. Macdonell and Dr. Matthew M. Y. Kuo, who supervised and helped me throughout my Ph.D. journey with utmost professionalism.

Chapter 1

Introduction

1.1 Background

Industrial Control Systems (ICS) have significantly changed human life. The quest to mechanise has led to increasing automation in industrial processes like manufacturing plants, transportation systems, and facility management and monitoring systems. Reliance on such technologies can potentially improve human lives but with a strong caveat that their failure can sabotage and reset our technological journey. Ubiquitous computing has converted the internet into a world-sized robot. Although this robot is not smart enough to match human intelligence, it will get smarter, more intelligent, and increasingly powerful. More importantly, it will also become more unreliable.

Security is a critical system quality, but its importance is often underestimated in ICS (Yılmaz & Gönen, 2018; Zahid, Tanveer, Kuo & Sinha, 2021). Legacy ICS were often contained in a controlled environment where all the components shared the physical locations with no remote access. Modern industrial systems are not confined due to the advent of Industry 4.0 (Lasi, Fettke, Kemper, Feld & Hoffmann, 2014). ICS actively communicate with the outside world that makes them vulnerable to security threats. Although the upper layers of ICS such as Supervisory Control And Data

Acquisition (SCADA) systems are secured using network security mechanisms (Ghosh & Sampalli, 2019), devices within control and field layers such as Programmable Logic Controllers (PLC), remote terminal units, sensors, and actuators are often left susceptible to attackers (Staggs, Ferlemann & Shenoi, 2017).

Figure 1.1 shows a generic configuration of devices in SCADA, control and field layers of an ICS. Control and field layers contain resource-constrained embedded devices that control and process inputs/outputs from sensors and actuators. Such devices have minimal resources regarding computing power and memory because they are designed to provide uncomplicated and reliable operations. Due to such restrictions, cryptographic algorithms implementing security requirements are often not considered viable in such devices, making them highly vulnerable to cybersecurity attacks (Dunlap, Butts, Lopez, Rice & Mullins, 2016). Stuxnet, which specifically attacked PLCs in the Iranian nuclear infrastructure, is a significant example showing the extent of the possible damage if such devices are improperly secured. A report by (Andreeva et al., 2016) shows that more than 90% of hosts connected to Industrial control systems have vulnerabilities ready to be exploited. The author recommends using security standards and the need for security certification to provide cryptographic services in ICS.

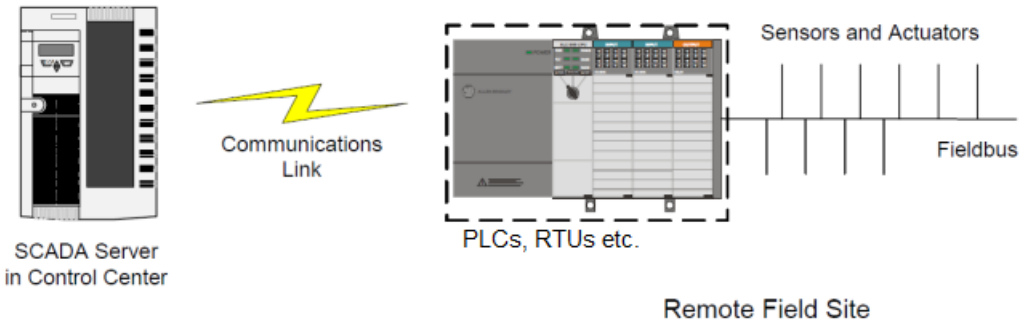

Figure 1.1: A general configuration of ICS

The complex requirements of highly distributed and heterogeneous ICS make them challenging to design and manage. Security requirements analysis and specifications is a laborious process in ICS due to the involvement of many stakeholders such as system

users, administrators, maintainers, and government regulatory organisations (Zahid et al., 2021; Mohamed, Challenger & Kardas, 2020; Green et al., 2017). Requirements are traditionally documented using software requirements specification documents and specified in natural language. (Giannakopoulou, Pressburger, Mavridou & Schumann, 2021). The misinterpretation of security requirements may result in severe consequences if such requirements are implemented unchecked in the resource-constrained devices (Pal, Hitchens, Rabehaja & Mukhopadhyay, 2020). Formal or semi-formal security requirements specification techniques can be helpful to reduce the ambiguity of natural language by using their inherent formal constructs. However, the recent research is inadequate regarding the application of formal methods for Security Requirements Engineering (SRE) of ICS as discussed in Section 2.2.9.2 of the thesis. Requirement traceability is an existing challenge for ICS (Zahid et al., 2021). A survey of current literature presented in Section 2.3.1 shows that the majority of the state-of-the-art approaches (Kavallieratos, Katsikas & Gkioulos, 2020; Varela-Vaca et al., 2020; Hansch, Schneider & Brost, 2019; Kivelä, Golder & Furmans, 2018; Zhou et al., 2020) deal with validation and verification of security requirements in ICS, also lacking meaningful traceability across the system development life-cycle of ICS.

Several SRE approaches and methods have been used in the domain of information technology that can also be applied to ICS. The prominent ones include SecureUML (Lodderstedt, Basin & Doser, 2002), Secure Tropos (Mouratidis, Argyropoulos & Shei, 2016), SQUARE (Mead & Stehney, 2005), SEPP (Fabian, Gürses, Heisel, Santen & Schmidt, 2010), MSRA (Maskani, Boutahar & El Houssaïni, 2016), CORAS (Den Braber, Hogganvik, Lund, Stølen & Vraalsen, 2007). SEPP, KAOS + anti-models, and CORAS partially support one or more security standards. However, these approaches also do not cover the full spectrum of system development life-cycle since they focus of requirements engineering phase.

Security standards provide a consolidated set of security requirements that are

validated by industry practitioners and academia. ICS specific and related security standards include NIST SP 800-82 (Stouffer, Falco & Scarfone, 2011), IEEE 6189 (AGA-12), IEC 62351 (Schlegel, Obermeier & Schneider, 2017), FIPS 140-2 (NIST, 2016), IEC 19790 (Kusumah & Andriawan, 2019), IEC62443 (62443-1-1, 2009), and ISO/IEC 15408 (ISO/IEC 15408-1:2009 Criteria, 2009). NIST SP 800-82 and IEC 62443 are specific to ICS security.

ICS security standard certified products ensure that the product has met the security requirements and has been subject to rigorous testing. Major ICS certification programs include ISASecure (*IEC 62443 Conformance Certification*, 2019) and Achilles that perform conformance of security requirements. The product certification process requires extensive requirements specification, design and testing collaboration between a vendor (product developer) and a certifier entity. Developers must provide development artefacts to certifiers who carry out detailed validation and verification of security requirements and associated artefacts of the end product. A common approach to requirements traceability through design and implementation can help to improve the time-to-market of the product since the vendors and certifiers aim to validate and verify security requirements, design and implementation of the same product. However, the lack of design guidelines for certifying ICS software is detrimental for stakeholders seeking to follow a standard process in ICS certification (Drias, Serhrouchni & Vogel, 2015). Recent studies (Matheu, Hernández-Ramos, Skarmeta & Baldini, 2020; Ehrlich, Gergeleit, Trsek & Lukas, 2020; Bicaku, Zsilak, Theiler, Tauber & Delsing, 2021) to help produce security standard compliant ICS software also focus on verification of non-functional requirements of already deployed ICS products instead of verifying functional requirements during the development phase. Another recent study (Constante, Soares, Pinto-Albuquerque, Méndez & Beckers, 2021) tries to solve this problem partially by presenting a framework to integrate Continuous Integration and Continuous Deployment (CI/CD) pipelines to security standard certification; however, it lacks the vital details of

the development life-cycle phases.

IEC 61499 (Vyatkin, 2009) is a standard for developing distributed ICS applications seeking to replace legacy ICS development standards such as IEC 61131. Work has been done to transform IEC 61131 (Ramanathan, 2014) legacy applications to comply with IEC 61499 (Peltola, Christensen, Sierla & Koskinen, 2007; Cabadini et al., 2019). However, IEC 61499 still requires new design models to comply with the distributed model provided by the standard (Lyu & Brennan, 2020). IEC 61499 can also be used to alleviate the problem of distribution within ICS security components by automated security requirements mapping techniques (Ehrlich, Wisniewski, Trsek & Jasperneite, 2018) that can help map security standards requirements to link with the ICS design. However, existing literature provides limited coverage for security design considerations for scalable and distributed ICS (Green et al., 2017). At the same time, current secure-by-design approaches for ICS, e.g. (M. T. Khan, Serpanos & Shrobe, 2017) are not entirely aligned with SRE and security standards negatively affecting security requirements to design traceability during the security certification process.

1.2 Research Questions

Research questions to proceed with this research after a detailed literature review presented in Chapter 2 are listed below:

RQ 1. What is the current state-of-the-art use of formal methods for managing non-functional requirements during the early-stage development of distributed embedded systems?

RQ 2. How can formal methods help integrate the analysis, specification and management of requirements from industrial security standards?

RQ 3. How can security requirements from the security standards be integrated with design activities using current ICS development standards?

RQ 4. What solutions can be applied to store and manage security requirements in conjunction with security standards and ICS development standards for device security certifications?

RQ 5. How do the solutions proposed in this research provide scalability to support the development of industrial-scale ICS that must conform to security standards?

1.3 Contributions

This research aims to address the current gaps regarding integrating ICS SRE, secure-by-design modelling and security implementation approaches to provide requirements traceability from requirements to verification and validation phase. The research contributes to the knowledge base by developing novel SRE and design techniques that help devices in ICS to conform to the requirements of ICS security standards. Significant contributions of this research as follows:

1. Chapter 3 introduces a novel concept of Secure links—design-time abstractions for IEC 61499 that allow designers to annotate secure communications.

2. A Confidentiality based Security Layer (CL4FB) for IEC 61499 applications and its implementation to support secure links is discussed in Chapter 3.

3. The design of an intrusion detection and prevention system based on IEC 61499 service interface function block for preventing availability attacks presented in Chapter 4.

4. A novel secure-by-design approach is proposed for developing ICS applications called the *Secure Links Development Method* (SLDM) presented in Chapter 5.

5. A novel model of a security requirements repository presented in Chapter 6. The repository stores labelled property graphs for cybersecurity requirements specifications and IEC 62433-4-2 security requirements in multiple partitions while emphasising requirements structure and relationships.

6. A formal definition of the IEC 62443-4-2 extended requirements structure discussed in Chapter 6 that helps select standard cryptographic primitives to guide the implementation of IEC 62443-4-2 requirements.

7. Chapter 6 discusses a novel process to integrate the repository with ICS design tools such as Secure Links (Tanveer, Sinha & Kuo, 2020) and TORUS (Sinha, Dowdeswell, Zhabelova & Vyatkin, 2018) from Chapter 5 in order to support end-to-end requirements traceability.

The above contributions of the novel SRE and secure-by-design techniques for ICS development lead to an integrated configuration of these techniques called Integrated Techniques for Secure-by-Design ICS Software (ITSIS) that is discussed in detail in Section 7.2 of this thesis. An implication of ITSIS is its application in the security certification of ICS components due to its ability to produce conclusive end-to-end traceable requirements. ITSIS adheres to the traditional phases of the development life-cycle, e.g.:

1. SRE by including security requirements repository.

2. A design phase adhering to secure-by-design paradigm using secure links.

3. Implementation of security methods and algorithms through IEC 61499 security function block library.

4. The use of TORUS for end-to-end traceability to help validation and verification phase.

Applying the novel secure-by-design techniques on case studies such as smart grid protection system, industrial mixer control system, baggage handling system, and a wind turbine system show an overall improvement across the ICS system development life-cycle. Secure links significantly improve the system, design complexity, scalability, and maintainability of ICS applications as shown in Section 5.7. In addition, the security requirements repository in conjunction with secure links and TORUS enables the production of intuitive requirements traceability matrix to map security standard requirements with IEC 61499 function block cryptographic implementations.

1.4 Research Design

The thesis explores the relatively uncharted territory where the confluence of SRE, security standards, and secure-by-design ICS throws new and exciting challenges. Development of secure-by-design techniques, design analysis, building prototype systems based on security requirements and their evaluation are centric activities to this study. Such methodology is essentially equivalent to the system development research methodology presented in (Nunamaker Jr, Chen & Purdin, 1990) that also adheres to the Design Science Research (DSR) Methodology. (Peffers, Tuunanen, Rothenberger & Chatterjee, 2007) show that the system development methodology provided by (Nunamaker Jr et al., 1990) covers major process elements of the DSR methodology.

Figure 1.2 shows the methodology and design of this thesis following DSR. A generic approach to DSR has three phases: problem identification, solution design and evaluation. Each phase is composed of several steps. DSR is a highly iterative process having a continuous “evaluation and redesign” loop to improve the solution. Phases can be sequential, but if required, they can be carried out in parallel. The processes may refer to each other, and the execution of the processes produces results of DSR. The procedure to carry out this research by DSR is discussed in the following sections.

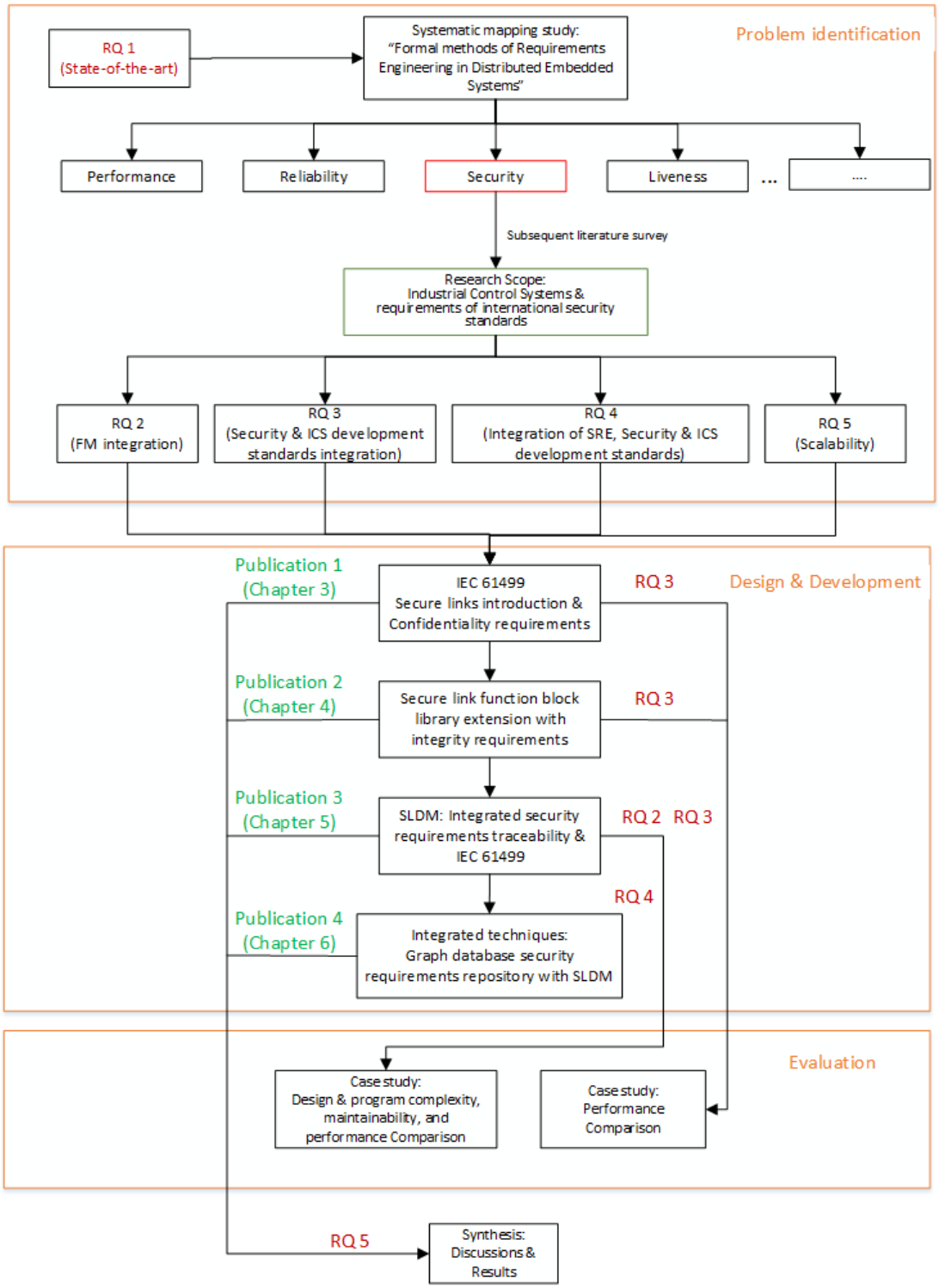

Figure 1.2: Research methodology and design

1.4.1 Problem Identification

As the first step of this phase, a Systematic Mapping Study (SMS) is conducted to explore the state of formal requirements engineering methods in Distributed Embedded Systems (DES). The rationale to choose DES as a broader scope area is the research coverage in ICS-related adjacent fields. A significant advantage of carrying out SMS is the identification of current research gaps and trends quantitatively. Moreover, the SMS was instrumental in answering the first Research Question (RQ). It has also helped with the formation and refinement of subsequent RQs.

The results of the SMS can be seen in three different meanings, i.e. formal methods, non-functional requirements and the type of DES. The most beneficial aspect of the results is identifying specific areas with a low concentration of research at the convergence points of the focused domains. Specific non-functional requirements, i.e. reliability, performance, security, liveness, have seen insufficient research, especially about cyber physical systems and ICS. With the landscape filled with research gaps, it is infeasible to focus on all areas demanding attention. A researcher must carefully select an area of interest based on their knowledge and expertise in such a scenario. It can be based on skills, familiarity with the current body of knowledge, and the selected domain's value concerning contemporary research. Consequently, selecting SRE and ICS design as an avenue for this research study is based on the gaps found in the SM study and according to the researcher's expertise and interest and the potential for impact.

A subsequent literature survey explicitly focusing on SRE based on ICS security standards and current ICS secure-by-design approaches was also carried out. Research shows that there is very little evidence of formal frameworks that can support the development of security manifested devices for ICS that can comply with the requirements of international security standards. The researcher also had similar experiences while

working on security-related projects where it was evident that traditional software development methodologies cannot sustain such endeavours' demands. The primary issue faced is the choice of requirement engineering strategies to comply with requirements from international security standards. For example, the security requirements of a particular system are challenging to map with security requirements from the standards in current SRE frameworks. Although the security goals like confidentiality, integrity and authentication still need to be specified at an abstract level, the detailed set of requirements fulfilling these goals come from the standards.

The security certification process involves vendors and third-party testers. The process is often lengthy, time-consuming and expensive due to the extensive collaboration required between the stakeholders (Lotz, 2020; Baldini et al., 2016; Fomin, Vries & Barlette, 2008). Therefore, it is imperative to have a robust requirements engineering process in place that is specifically suited to such kinds of system development processes where security requirements and their validation is of foremost concern. Such processes frameworks can help in reducing the number of iterations for requirements validation between the vendor and the external testing bodies especially in case of requirements change (Baldini et al., 2016).

ICS development standards such as IEC 61499 provide mechanisms to distribute the functionality of a software module across multiple ICS devices. There are gaps related to security analysis, development and management operations (Fruth & Nett, 2014), therefore there is a need to use more security standards and approaches in the context of distributed industrial control systems. The issues and challenges arising from the fulfilment of requirements of a security standard in distributed environments are still areas of much anticipation. Research in this direction has helped answer RQ3 and RQ4, enabling us to develop integrated techniques for SRE and security standards such as IEC 62443 and IEC 61499 ICS application development standards.

Scalability is a by-product of technology advancement that must be compulsory for

the solutions produced for the industries with explosive growth. Rapid growth in the technologies encompassing ICS demands solutions that can adapt to the ever-changing requirements and dynamics of the industry (Green et al., 2017). The solutions that cannot keep up with the advancement are set to become obsolete relatively quickly (Marali & Sudarsan, 2018), especially in the contemporary ICS domain due to rapid automation with the advent of Industry 4.0. Therefore, solutions that integrate security requirements with ICS design and implementation must have an inherent scalable design. Consequently, RQ5 focuses on achieving the scalability of solutions produced in RQ3 and RQ4, particularly for the industry adoption.

1.4.2 Design, Development and Evaluation

This research builds upon implementing minimum security requirements and subsequently adding more complex functional security requirements also introducing the distributed design, refined after each subsequent iteration. An incremental approach is followed concerning the artefact development starting with the basic security requirements and building upon these requirements after each evaluation phase. This process follows the DSR methodology that is a choice for this research. It helps explore the possible challenges and issues regarding resource constraints and demands of security standards in ICS devices.

Solutions are produced during the design and development phase, generating artefacts to advance the research systematically. Research questions derived in the problem identification phase are answered with each step of the research, as shown in Figure 1.2. Solutions are identified incrementally to achieve the overall goal of developing integrated techniques for SRE and design for ICS applications that conform to ICS security standards. Each solution adds to the body of knowledge and also to the overall goal of this research. The evaluation is done using industrial case studies and matrices

such as design complexity, program complexity, maintainability, and requirements traceability. After carefully analysing a solution, the resulting artefact is improved by extending the idea to produce an enhanced solution.

The first step in the "design and development" phase of this research introduces a secure link construct for ICS applications. An initial blueprint of a secure link is designed, developed and evaluated. A secure link is a construct to specify security requirements over IEC 61499 function blocks. These requirements include confidentiality, integrity and availability. An ICS application developed using IEC 61499 can specify security requirements over the data links of the function block. In IEC 61499, a function block network can contain multiple function blocks distributed over multiple devices that communicate with each other using data and event links. Secure links can also provide encrypted data communication between a data link of two more function blocks distributed over multiple devices. This approach is proposed as Confidentiality Layer for Function Blocks (CL4FB) in this thesis. A case study of a smart grid protection system is used to evaluate the solution against the performance requirements of the grid when confidentiality requirements are applied to the secure links. A conference publication (publication 1) helped to evaluate the secure link concept through the research community. Furthermore, the concept also helps answer the parts of RQ3 that deal with the integration of security with modern IEC 61499 ICS development standard.

The next phase of the research is to enhance and improve the secure link design to support different cryptography goals. Cryptography support for availability goal is developed and tested for ICS in this iteration since availability is the most important goal for an ICS (Knowles, Prince, Hutchison, Disso & Jones, 2015). Since secure links are initially thought to be a design enhancement to IEC 61499, implementation for all the main cryptography goals is desired to support the efficacy of secure links. An open-source Intrusion Detection and Prevention System (IDPS) was implemented and

tested for IEC 61499 using Wago PLCs. The outcome helps in the feasibility establishment of a lightweight IEC 61499 IDPS providing additional security mechanisms. It also demonstrates that resource-constrained ICS devices can use security mechanisms meant for large-scale ICS infrastructure. The concept of incrementally adding security mechanism implementations forms a function block library concept that contributes to securing communications within IEC 61499 ICS applications. The idea of the library is later consolidated in the next phase of the research. Moreover, this phase assists in answering RQ3. Publication 2 discusses the details of IDPS in IEC 61499 applications.

The subsequent iteration consolidates secure links by integrating them with requirement traceability mechanisms such as TORUS in Secure Link Development Methodology (SLDM). The methodology is a secure-by-design approach that is tested using an extensive case study of an industrial mixer control system. Secure links and a function block library are formalised that are the building blocks of SLDM. A compiler algorithm is also proposed and implemented to transform the secure link abstractions into a deployable distributed ICS application. In addition, TORUS provides the requirements traceability support at the design and implementation stage. Therefore, SLDM is an overall solution providing the secure-by-design approach to develop ICS applications in IEC 61499.

A concept of security requirements repository is tested using a wind turbine case study in the final phase of this research. Security requirements repository contains unique system-specific as well IEC 62443 security standard requirements in static and dynamic partitions. This concept forms a novel solution that can provide end-to-end traceability for the IEC 62443 security standard requirements. IEC 62443 provides a set of requirements and provides a pathway to certify an ICS device/application.

The combinations of resulting artefacts of the DSR iterations help form an integrated approach to developing secure-by-design security standard compliant ICS software. The scalability of each artefact is discussed regarding system and design complexities,

maintainability and the overall effect of the proposed integration in security certification of multiple ICS products. It helps to answer RQ4 and RQ5 related to the process of integrating security requirements with ICS development standards and the scalability of overall integration.

1.5 Thesis Organisation

The rest of the thesis is organised as follows: Chapter 2 presents a detailed systematic mapping study and an extended literature survey regarding ICS security. Chapter 3 provides an initial concept of IEC 61499 secure links abstractions and an evaluation of the confidentiality layer for function blocks. Chapter 4 extends the security requirements implementation regarding the availability of IEC 61499 based ICS devices that provides the prelude to a security library of function blocks. Chapter 5 proposes the Secure link development methodology that consolidates the concept of secure links along with requirements traceability using TORUS. Chapter 6 proposes a novel model of security requirements repository and its integration with ICS design-time tools such as secure links and TORUS. Chapter 7 discusses the implications and limitations of integrating security requirements repository, secure links, security function block library implementation and traceability using TORUS for ICS device certification. Finally, Chapter 7 also provides the conclusions and future directions.

Chapter 2

Literature Review

2.1 Introduction

This chapter presents the background and state-of-the-art literature review regarding requirements engineering methods, security standards and related design aspects in ICS. The original goal of the researcher was to find out the research gaps in the formal methods of requirements engineering in the broad domain of Distributed Embedded Systems (DES) that was the main research question at the inception of this research.

An extensive Systematic Mapping Study (SMS) presented in Section 2.2 is carried out in the DES domain to answer the research question *RQ1* listed in Section 1.2. The SMS covered 201 primary studies (Appendix A) selected through carefully designed selection criteria. The SMS finds several trends and gaps regarding the formal methods for requirements engineering in DES. Some prominent trends are the emergence of formal frameworks in IoT and the emphasis on formalising in the automotive industry. Major concerns are also discussed, including the practitioners' lack of application of formal specification languages and inadequate formal Security Requirements Engineering (SRE) in DES.

Among the research gaps listed in the SMS, the lack of formal SRE methods is

a significant threat to ICS, a sub-domain of DES. ICS are gaining attention with the emergence of the industry 4.0 phenomenon. Legacy ICS were confined to a physical location. However, the advances in network connectivity have compelled ICS to be interconnected more than ever. Such concerns on ICS motivated the researcher to narrow down the research to address the challenges in SRE of ICS. The researcher's practical experience in the industry regarding the challenges in the requirements engineering and development of security certified products also motivated and propelled to advance the body of knowledge in this field.

A further literature survey is also conducted by going through the current work in SRE incorporating ICS security standards and their impact on the overall development of security certified products. This part of the literature survey is divided into two major parts: *1)* the need for SRE in ICS and the current challenges, and *2)* the state-of-the-art research and the challenges in applying security standards with SRE methods in ICS. Such an exercise helped the researcher forming research questions *RQ2, RQ3, RQ4, and RQ5* listed in Section 1.2.

2.2 Systematic Mapping Study

Requirements of large-scale DES tend to be complex and tangled mainly due to the heterogeneity of components. Formal methods (FM) that are mathematical techniques provide a way to specify, validate and verify requirements that can remove ambiguity arising from the use of natural language. As the emphasis on automating industrial processes has become a growing trend, researchers and practitioners are keen to employ FMs in the system development process's Requirement Engineering (RE) phases. A wide variety of FMs are being used in RE and design phases of development of different DES types such as Cyber-Physical Systems (CPS), Internet of Things (IoT) and Sensor Networks, etc. As one size does not fit all, the choice of appropriate FM of RE is

essential according to the nature of an industrial DES application (Kossak & Mashkoor, 2016). Such a choice can only be made after surveying the available information in this area. Research on FM of RE in DES is growing with every passing year, yet there is no instrument that researchers can use to have a bird's eye view of the confluence of these three fields. Therefore, our motivation to conduct this study is to provide a comprehensive map of the literature regarding FM, RE and DES. In order to achieve this goal, the researcher carried out a systematic mapping process with 201 primary studies. The researcher reports the results in the context of topic independent, i.e. bibliographic mapping and topic dependent classifications. The contributions of this mapping study are the identification of research gaps and patterns from the last decade. After compiling the lists of most focused: FMs, Non-Functional Requirements (NFR) and DES, the researcher provides the quantitative summary of the combination of all entities in those lists in the form of a heat map that can help researchers to identify the quantity of research done in a particular combination of FM, NFRs and DES.

2.2.1 Introduction

Software is becoming an increasingly large and complex part of modern-day life. Challenges in developing software for large-scale systems such as intelligent transportation, distribution systems, factory assembly units and other Industrial Control Systems (ICS) have highlighted new problem areas such as resource consumption and system qualities like distribution, reliability, safety, and security. The distributed nature of these large-scale industrial systems lends an added layer of complexity in addressing these challenges. In recent years, new DES paradigms like *Cyber-Physical Systems (CPS), Internet of Things (IoT) and Wireless Sensor Networks (WSNs)* (Mosterman & Zander, 2016; Hinchey, Rash, Rouff & Gračanin, 2006) have emerged, highlighting the need for systematic development.

Large-scale industrial DES contain many independent yet highly cohesive components. These collaborating components must meet complex and heterogeneous component-level and system-level requirements. RE involves gathering, analysing, specifying, and managing requirements for a system during its lifetime. Handling a growing set of complex requirements for large DES like automated metro transportation systems is too cumbersome and challenging for current RE tools and methods. Moreover, since most requirements found in System Requirement Specification documents (Bourque, Dupuis, Abran, Moore & Tripp, 1999) are written in natural language, there is always a risk that different people may interpret these requirements differently. These issues highlight a clear need to provide scalable, systematic support for handling requirements, especially for increasingly larger and more complex DES.

FMs are mathematical tools and techniques that can be used to specify the requirements more rigorously (Campos, 2010). FMs have found extensive use in the design and testing of sizeable safety-critical systems (Berry, 2016). While FMs have found use in RE of some systems (Babin & Lustman, 2001; Martins & Gorschek, 2016), their application across DES remains inconsistent and sporadic. There is evidence that the industry practitioners have apprehensions about the use and value of FMs (Kossak & Mashkoor, 2016). High project costs due to increased demand for specialist skills, lack of familiarity amongst engineers, and a perception that formal modelling and analysis takes longer, are some contributing factors to the limited adoption of FM in industry (Bjørner & Havelund, 2014).

With CPS and IoT systems becoming mainstream, new techniques and tools have been introduced to hand requirements relating to communication and module security, performance, fault tolerance. FMs have also found use in automating parts of system design. For instance, model checking tools have found use in automatically testing formally specified requirements on system designs (Cheminod, Bertolotti, Durante, Sisto & Valenzano, 2006; N. K. Singh, Aït-Ameur, Pantel, Dieumegard & Jenn, 2016).

Recent years have seen numerous formal specification languages which have paved the way to the development of compatible model checking tools (García-Ferreira, Laorden, Santos & Bringas, 2014; Křetínskỳ, 2016; L. Liu, Kong, Ando, Yatsu & Fukuda, 2013). Moreover, several new formal frameworks embrace the rapidly improving technologies in the field of DES (Manifavas, Hatzivasilis, Fysarakis & Papaefstathiou, 2016; Derhamy, Eliasson, Delsing & Priller, 2015). The rapid introduction of new FMs over the years has posed a challenge around consolidation (Bjørner & Havelund, 2014). Criteria to select a suitable formal method for an industrial application has been prescribed in (Kossak & Mashkoor, 2016). Experience and readily available knowledge of existing FMs can aid researchers and practitioners in selecting the most appropriate solutions for their projects. Unfortunately, no current work meets this need of providing a vast landscape of FMs that can be used during the RE of DES.

An SMS (Petersen et al., 2008) of available FMs for RE in DES has been carried out, resulting in 201 selected primary studies (Appendix A). A systematic mapping study is a form of secondary study that lays out a map and provides an overview of the literature in a selected area of research (Petersen et al., 2008). The focus is on presenting the breadth of the research area by classifying related published articles and identifying the latest research trends. An initial search over multiple online databases such as Scopus, IEEE, ACM, SpringerLink and major conferences and journal publications resulted in an initial pool of studies. This list is pruned by manually reading titles, abstracts, introduction and conclusion sections of the articles. The final selection was carried on by consulting with experts in the field. A comprehensive search for other secondary studies was carried out to ensure the novelty of this study. To the best of the researcher's knowledge, there appears to be no existing review or secondary study covering the scope of this study. Some relevant literature review studies that are peripherally related to this study have been discussed in Section 2.2.2.

The results of this SMS present a quantitative map of the FMs and tools. The primary

studies are classified according to bibliography, phases of RE, types of requirements, and target application areas. The mapping study also reports important patterns and research gaps. For instance, while a sizeable chunk of existing research focuses on safety and timing requirements, most studies target the requirements specification phase of RE mainly because of the rapid rate at which new formal languages are introduced. The mapping study finds an increasing need to formulate FMs to target more requirement-types, such as security, that are becoming more prominent. A majority of studies propose new formal frameworks, indicating that existing frameworks need consolidation, stabilisation and standardisation. The mapping study also provides a comprehensive heat map that shows the quantity of research concerning various aspects of FMs, RE and DES.

2.2.2 Background and Related Surveys

This section clarifies key terminology around FMs and requirements engineering used in this mapping study.

2.2.2.1 Formal Methods

A formal method is a mathematical representation of a theory and its proof (Davis et al., 2013). FMs allow software and hardware systems to be rigorously specified, analysed and verified against functional and non-functional requirements. Formal modelling, specification and verification can facilitate the formalisation of requirements engineering-related activities. Formal verification has also proven helpful in automatically or semi-automatically finding bugs while providing comprehensive coverage (Hierons et al., 2009). Model-checking and theorem proving are some well-known FMs, and many tools for these frameworks exist. Model-checking is fully automated and can verify if a system model satisfies a given requirement. The requirement must be stated

using a formal language, such as temporal logic (Hierons et al., 2009). Model-checking suffers from the well-known state explosion problem (Baier & Katoen, 2008), where system models for real-time systems may become prohibitively large for the algorithm to work within acceptable time frames. Theorem proving is semi-automated and allows a user-guided and restricted exploration of a system's state space to characterise its properties and satisfy given requirements. Using FMs early in the system development life cycle, when system models are typically small and testing is focused on individual components, can help overcome the state explosion problem to some extent. Hence, FMs during requirements engineering, which is the first phase of the development life cycle, can potentially provide a high return on such an investment.

There are many formal tools and techniques for modelling and analysing software-intensive systems (Bjørner & Havelund, 2014; Lecomte, 2009; Weyns, Iftikhar, De La Iglesia & Ahmad, 2012). Most software products vary in their requirements; the most suitable formal method or set of methods is always application or context-specific. There is limited research in helping practitioners to choose suitable FMs for their software projects. After conducting a literature review and also drawing knowledge from their experiences in the industry, (Kossak & Mashkoor, 2016) presents the criteria and set of guidelines for choosing FMs suitable for a project: *1)* Modeling criteria *2)* Supported development phases *3)* Technical criteria *4)* Human/Social Criteria *5)* Industrial applicability

2.2.2.2 Use of Formal Methods in Industry

As the size of software-intensive systems increase, testing using conventional methods like simulation-based testing is a laborious task and even impossible at times. The rigour of FMs makes them more suitable for analysing large-scale industrial applications. FMs have been around for more than 45 years. Yet, their usage in industrial-strength software systems has been limited due to several constraints (Bjørner & Havelund,

2014), including the state-explosion issue mentioned earlier. Despite their potential, FMs have been an enigma for industry practitioners due to the high level of skill required to use them. As FMs are complex and require a reasonable amount of expertise, there is a caveat that they tend to become an overhead in developing small scale software or hardware applications because of the required implementation efforts. This overhead can become an overhead in the development of small scale software or hardware applications. Despite these hindrances, the dependence on FMs in industrial-scale software has grown over the years (Davis et al., 2013; Fitzgerald, Bicarregui, Larsen & Woodcock, 2013; Lecomte, 2009; Weyns et al., 2012). Industries like telecommunications have created customised FMs for their domains (Ardis, 1997).

A survey conducted by (Fitzgerald et al., 2013) reports FMs in industrial-scale systems. The survey focused on 98 projects, with a majority representation from transportation and finance-related projects. Most applications considered were real-time and distributed. According to the findings, more than 80 projects used formal techniques in the specification/modelling phase, indicating that practitioners use these techniques very early to ensure the correctness of requirements. Other significant phases of the project where formal techniques find use are execution, inspection and verification. Regarding time, cost, and quality, respondents mainly were neutral about the time and cost-effectiveness, but it was clear that product quality improved by applying FMs. There was a general feeling of satisfaction amongst practitioners about using formal tools and techniques in their projects. Results also tend to confirm that tool-support for formalisation plays a vital role in the industrial sector. FMs tend to be applied more in a lightweight manner to target specific subsystems and components.

A comprehensive survey of FMs' use in self-adaptive systems appears in (Weyns et al., 2012). FMs use in this domain is low, especially for ensuring that requirements are met. Embedded software systems were reported to have found the most application for FMs. Regular algebra has been used as a formal modelling language in most studies,

backing another finding that modelling languages are the favourite choice for property or requirement specification/verification. While efficiency and performance were primary concerns, security, scalability, accuracy, availability, maintainability and portability of self-adaptive systems have hardly been considered through FMs. Moreover, the primary use of these methods is for modelling and reasoning purposes.

A survey of the barriers faced by the American government and private manufacturers of large systems in adopting FMs is presented in (Davis et al., 2013). Results indicate that the use of FMs in the subject organisations has increased gradually amid some hurdles. One of the significant barriers reported is that individuals involved in the system development process have very little or no education and training in the use of formal tools and techniques, a sentiment also echoed by (Bjørner & Havelund, 2014). Other barriers such as tool support, industrial environment, engineering, certification, misconceptions, scalability, evidence of benefits and cost also hinder FMs in the industry. Several mitigation schemes have also been identified in response to survey questionnaires, with education, tool integration and evidence of benefits being the most important ones.

FMs can potentially address some issues commonly encountered during requirements engineering of DES. Software requirements are generally specified in natural language, which induces ambiguity and space for multiple interpretations. Large-scale DES often have complex and meshed requirements, and ongoing clarifications from stakeholders are often complicated and expensive to obtain. Formally specified requirements can reduce ambiguity. Moreover, formal specifications pave the way for requirements to be validated and verified formally (Easterbrook et al., 1998).

However, effective use of FMs in requirements engineering requires specialised expertise and extensive validation (Lecomte, 2009). Semi-formal languages for modelling requirements like UML (Latella, Majzik & Massink, 1999; Snook & Butler, 2006) and SysML (Linhares, de Oliveira, Farines & Vernadat, 2007) are currently in wide use in

industry. A systematic literature review on modelling languages for software product lines (Sepúlveda, Cravero & Cachero, 2016) shows the minimal adoption of formal languages for requirements modelling. There is a clear need for active co-development of such languages with industry, robust tool support, and empirical validation. Another systematic literature review of 151 primary studies about requirements engineering practices in developing safety-critical systems is presented in (Martins & Gorschek, 2016). It stresses that requirements engineering and safety engineering processes need to be better integrated as very few current approaches target both. Also, engineers tend to favour traditional and established approaches like Fault Tree Analysis (FTA) and Failure Modes and Effects Analysis (FMEA) for safety analysis, rather than adopting newer but less established formal approaches like Requirements State Machine Language (RSML/RSML^{-e}) (Jaffe, Leveson, Heimdahl & Melhart, 1990; Leveson, Heimdahl & Reese, 1999). It shows the widening gap between academic research and industry practice.

2.2.3 Systematic Mapping Study Methodology

This study follows the systematic mapping study method proposed by (Petersen et al., 2008; Petersen, Vakkalanka & Kuzniarz, 2015). An overall view of the process followed is shown in Figure 2.2, and an explanation of each step as follows.

2.2.4 Scoping and Formulating Research Questions

The scope of this research is confined by three boundaries - *Formal methods, Requirements Engineering* and *Distributed Embedded Systems*, as illustrated in Fig 2.1. DES covers a range of systems, including control systems being increasingly used in automotive, avionics/aerospace, industrial automation, telecommunication, consumer electronics, intelligent homes and, health and medical equipment (Helmerich et al.,

2005). In recent years, new paradigms like *Cyber-Physical Systems (CPS), Internet of Things (IoT)* and *Wireless Sensor Networks (WSNs)* have emerged and absorbed seamlessly into DES domain (Hinchey et al., 2006; Mosterman & Zander, 2016; Yu & Xue, 2016). The growing complexity and sizes of DES require more systematic and automated tools and processes to keep up with time-to-market pressures while also ensuring adherence to increasingly complex and larger requirements sets. Formal methods promise these much-needed features. This mapping study takes on the challenge of discovering the convergence points of the three boundaries to identify the current state-of-the-art within the chosen scope.

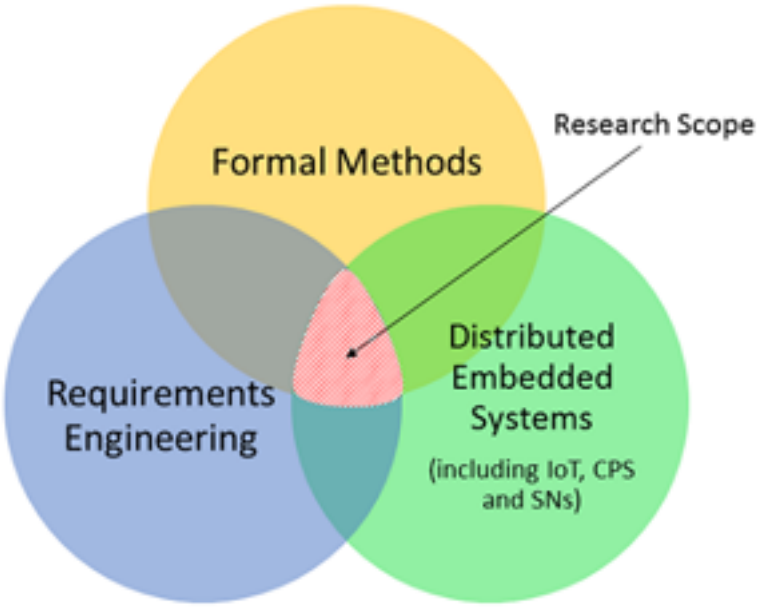

Figure 2.1: Research scope.

The identified scope leads to the following research questions:

SM-RQ1: Which formal methods are proposed or in-practice for requirements engineering and further applied in DES? Answering RQ1 provides a broad overview of the landscape and distribution of current works within the scope.

SM-RQ2: What formal methods target functional and non-functional requirements of DES?

Functional and non-functional requirements are equally important for a system. Functional requirements are related to system functions (what the system must do), while non-functional requirements relate to system qualities (how well the system should function). Hence, the functional/non-functional differentiation gives us well-defined categories for the surveyed works.

SM-RQ3: What are the latest research trends and the most investigated aspects regarding the formalisation of RE in DES? What are the research gaps and the latest challenges? This question highlights the most studied themes within the scope and the gaps within the current state-of-the-art.

SM-RQ4: What type of publications constitute the realm, and which forums have been more active in the last decade? This question calls for quantitative analysis of the surveyed works, supplementing the findings from answering SM-RQ3.

2.2.4.1 Searching

The search process leads to identifying primary studies to review and involves several steps, as described below. Figure 2.2 shows the overall systematic process, and subsequent steps are explained in the rest of this section.

Before searching, a preliminary step followed in this study is to confirm its novelty by ensuring no other existing surveys overlap with the identified scope. Following a comprehensive search using Scopus, 111 secondary studies published are found after 2010 containing references to the keywords *formal* and *requirement*. In addition to carefully reviewing these articles, a manual search is conducted following consultation with experts. The researcher searched for articles in the following venues: *IEEE Transactions on Software Engineering, Springer's Requirements Engineering Journal, Springer's Formal Methods in System Design, Transactions on Parallel and Distributed*

Systems, IEEE International Conference of Requirements Engineering, IEEE International workshop on Empirical Requirements Engineering. After going through the mentioned rigorous process, the researcher is confident that no other SMS overlaps with the scope of this SMS.

The first step in searching is *keyword identification* that converts the research questions into specific keywords that are later used to form a search string. Keyword identification followed the well-known *Population, Intervention, Comparisons and Outcomes* (Petersen et al., 2015). The researcher also manually extracted candidate keywords from known articles. Two keywords, formal and requirements, were chosen to quantify two of the three boundaries of the research scope. Concerning the much more fragmented domain of distributed embedded systems, after consulting with experts and going through known papers, a list of keywords was created which relates more closely to DES sub-domains. These keywords are *embedded, Internet of things, cyber physical* and *sensor networks*.

Scopus, IEEE, ACM and SpringerLink databases for the online search in the *database selection* step. Selecting multiple databases helps ensure a sufficient overlap to balance out each database's shortcomings (Franceschini, Maisano & Mastrogiacomo, 2016).

The keywords identified are combined to form a search string to identify candidate articles from the target databases. In addition, the following broader search string was used to minimise omissions and blind spots:

formal AND requirement AND (embedded OR "internet of things" OR "cyber physical" OR "sensor networks")*

The keyword *formal* is annotated with the wild card character (*), which means the use of relevant variations such as formal {methods, modelling, verification, specification, formalism, formalization.}

In the *searching* step, the search string was restricted to the title, abstract and

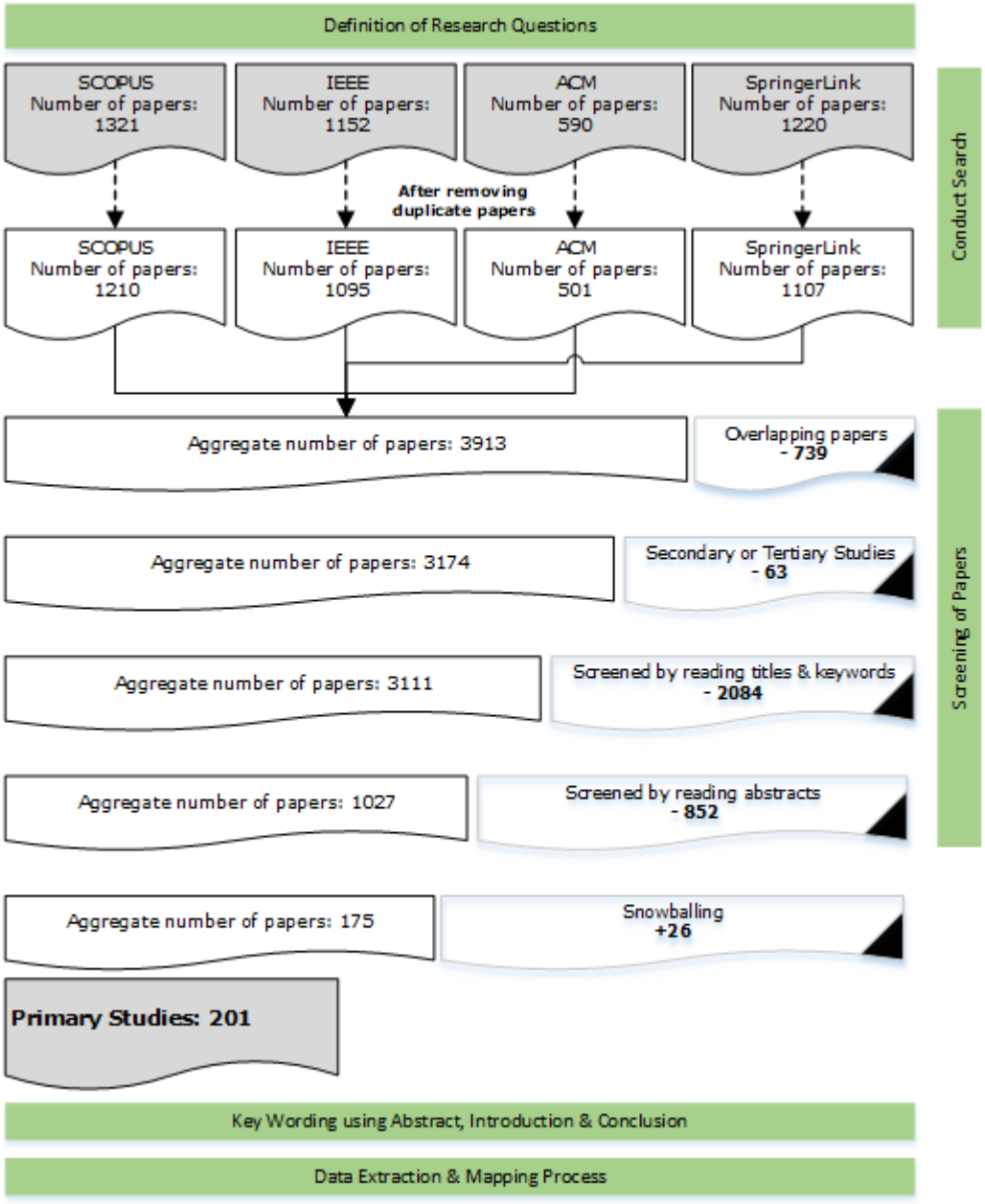

Figure 2.2: Search Execution. Processes in green boxes represent systematic mapping process by Petersen et al. (2008).

keyword sections in Scopus, “Metadata Only" in IEEE Xplore (command search), full-text search for ACM digital library and Title-only search in SpringerLink.

The *inclusion* and *exclusion* criteria are applied to chose only relevant articles for this study. The inclusion criteria are restricted to include only those studies that are from computer science and related disciplines, written in English, published in a peer-reviewed journal, conference, workshop or symposium, and appeared online only after 2006. The choice of 2006 as a cut-off year ensures the focus on the current state-of-the-art. Also, searches on the selected databases revealed that research in DES sub-domains, e.g. CPS gained momentum in or later than 2006. By creating an epoch in 2006, a balanced view of the DES domain can be obtained, including generalized literature on distributed embedded systems and newer developments such as IoT, CPS and WSNs. The exclusion criteria are used to discount some studies by excluding books and theses, non-peer-reviewed publications, secondary or tertiary studies, duplicates of another study already included in this survey, and publications for which full-text is unavailable.

Figure 2.2 shows the chronology of the *search execution* step. The latest iteration of the search was conducted in April 2021. The initial search returned 3913 articles, out of which 793 overlapping articles were removed. Another 63 secondary or primary studies were removed due to the set exclusion criteria. Afterwards, the pool of 3111 papers was first screened by reading titles and keywords, resulting in 1027 papers which were further reduced to 175 after a careful examination of their abstracts. Snowballing was applied for articles referenced directly by each of the 175 articles (forward snowballing) or those articles that directly reference any of the surveyed articles (backwards snowballing) (Jalali & Wohlin, 2012). As a result, another 26 articles were identified and included, resulting in a final tally of 201 primary studies (Appendix A) to be surveyed.

2.2.4.2 Data Extraction

The data extraction process involves identifying keywords while reading abstracts and the "introduction" and "conclusions" if the abstracts are not transparent or well-written (Petersen et al., 2015). On the other hand, Open-coding enables the reviewers to allocate a code or label to terms or phrases, not necessarily according to their literal meaning but according to the concept behind the terms (Corbin & Strauss, 2014). Thus, it underpins creating a categorical and conceptual schema of the text or data in focus. The author employed both key-wording and open coding for data extraction. Key-wording followed the three-pass method described in (Keshav, 2007) where if an article's abstract, introduction or conclusions seemed relevant, subsequently deeper investigations were used o confirm the key-wording. A rubric shown in Table 2.1 is developed to put a framework around the data extraction and classification process to address the research questions.

Table 2.1: Framework Rubric for Data Extraction and Classification

T1. Paper Title	T7. Publisher	T13. Tool, Language, Framework
T2. Authors	T8. Year	T14. Types of Requirements Addressed
T3. University/ Research Group/ Organization	T9. Keywords or Concepts (Open coding)	T15. Requirements Engineering Phase
T4. Country	T10. Research Type	T16. Specification Language
T5. Publication Venue	T11. Research Method	T17. Type of Distributed Embedded System
T6. Publication Type (Journal/ Conference/ Workshop/ Symposium)	T12. Type of Formal Method	T18. Case Study

The synthesised data is analysed using text-mining techniques to retrieve quality information, patterns and trends from the data set. Recent research (Felizardo, Nakagawa, Feitosa, Minghim & Maldonado, 2010) has also focused on developing text mining tools principally for systematic mappings. The data extraction rubric contains inherent features for implicit classification and categorisation. For tuple T9 of the rubric, a

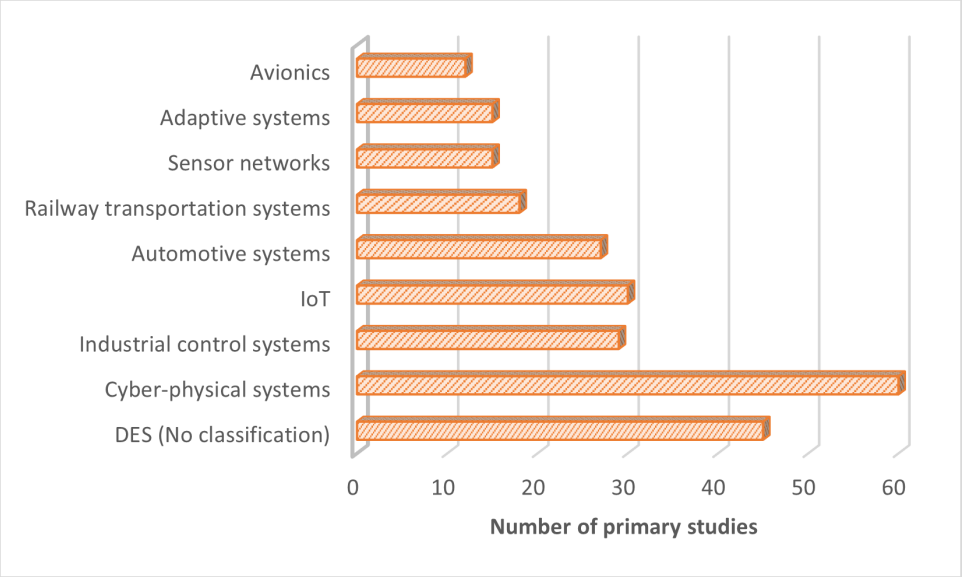

Figure 2.3: Number of studies by DES types

text-mining extension of RapidMiner tool (Hofmann & Klinkenberg, 2016) was applied on open-coded keywords and phrases to identify patterns, trends and facets within the research area. The outcomes of this analysis are discussed in detail in the following sections.

2.2.5 Classification of Distributed Embedded Systems

This section reports the distribution of the surveyed works based on the DES sub-domain they target. While many works explicitly target one of the identified sub-domain, including CPS, IoT, and WSNs, some are more generic or explicitly mentioned. For works in the latter category, the researcher carefully read through case study sections, if available, to extract this classification as per item T17 of Table 2.1. Figure 2.3 shows the distribution of the selected primary studies according to this classification.

The number of unclassified studies that cannot be categorised into any of the identified sub-domains is 45. Another 60 studies related to CPS explicitly capture the cyber (software) that typically runs on embedded computers to control physical processes.

CPS have found extensive use in safety-critical applications like avionics, transportation systems, ICS, where errors can cause catastrophic losses. As CPS becomes mature, practitioners are moving towards systematic design and formalisation of development processes. A good part of the 60 studies provides a formalisation of enabling solutions for CPS. Some of this research has focused exclusively on several other technology domains like communication, energy, manufacturing. that are part of CPS. Modern automotive systems are perhaps the most used CPS in everyday life. Embedded computers have taken control of several features such as braking, transmission or real-time engine management, which used to be mechanical or electronic. A modern luxury car typically contains more lines of code than a modern-day aircraft. This increase in size and complexity of CPS also relates to the complexity of requirements to be handled, adding more importance to handling RE phases more systematically. Out of 60 primary studies within CPS, 27 touched upon the subject of formalisation of RE processes for automotive systems.

IoT systems are closely related to CPS but differ due to an IoT system's exclusive dependence on internet connectivity for orchestrating smart objects. IoT systems contain networks of devices, some of which can be CPS too. Billions of devices are expected to connect into the form proposed by the IoT paradigm (Gubbi, Buyya, Marusic & Palaniswami, 2013). IoT based formal solutions are part of 30 of the surveyed studies. A pertinent pattern that emerges within the surveyed works is that 10 out of 30 propose new frameworks (**S[9], S[23], S[29], S[46], S[67], S[68], S[75], S[119], S[158]**, and **S[192]**). The mapping study further analyses this trend of novel frameworks in IoT in Section 2.2.9.

Transportation systems have also moved towards automation in recent years. All 18 studies in this category provide solutions that relate to the highly regulated railway's systems. Modern railway networks rely heavily on CPS for scheduling, switching and crossing operations. Studies **S[90]**, **S[92]**, and **[S179]** provide formal modelling

methods to verify the safety and liveness properties of railway crossing systems. **S[13]** proposes a novel idea of a Rail Internet of Things (RIoT). They design an architectural platform that includes components of railways systems and data communication interfaces, emphasising security management. Other important case studies on railways includes European Railway Traffic Management System (ERTMS) (**S[27]**) and Route-based tramway control systems (**S[60]**).

Automation of daily tasks has increased the reliance on sensor technology. Sensors are an integral part of the DES domain having footprints in CPS, IoT, Avionics, Automotive, and Robotics systems. Studies (**S[26], S[37], S[43], S[69], S[87], S[95], S[96], S[115], S[131], S[166]** and **S[185]**) have exclusively discussed formal RE solutions in sensor networks. These works generally target non-functional requirements such as timing, safety, liveness and dependability. Sensor nodes tend to be tiny devices with limited battery and power capability, especially in wireless sensor networks; communication security is an important requirement to be addressed. Analysis indicates that only one study discusses security while no study looks at performance and power requirements in sensor networks.

2.2.6 Formal Requirements Modeling

This section refers to the following research question:

SM-RQ1: ***Which formal methods are proposed or in-practice for requirements engineering and further applied in DES?***

The number and complexity of requirements in DES make it imperative to organise and manage them systematically. Typically, requirements and later architectural details are captured as views, where each view describes either some functional aspects or specific qualities of the system. As part of creating views, models are created to describe the targeted functionalities or qualities. Models can encompass individual requirements

as well as large sets of requirements.

Although easy to understand, requirements described in (or modelled using) natural language are considered to be ambiguous due to the tendency for varying interpretations. With large-scale DES, ambiguity can become amplified due to a possibly large number of interdependent requirements. Informal requirements models make it hard to trace, manage and validate requirements. Formal modelling can alleviate some of these problems. An essential advantage of a requirements model built using a formal modelling language is the ability to automatically analyse multiple requirements models or verify the satisfaction of a requirement on a system design or implementation using a model checking tool.

Formal modelling has been the most studied concept in the surveyed primary studies. More than 40% of the 201 studies have used the term *formal model* to describe a requirements model. In the other 60% of the studies, the majority have used a modelling methodology to model or specify requirements, indicating that formal requirement models are used although not specified explicitly.

2.2.6.1 Formal Requirements Modelling Techniques

Figure 2.4 shows the various formal modelling categories encountered in this SMS.

2.2.6.1.1 Logic Based Techniques

A method to formally express an argument is to specify it in terms of a formal logical expression. Similarly, requirements may be expressed as sets of arguments that can be formalised using formal logic. A testament to the importance of expressing or modelling requirements in a logical form is a pertinent pattern found in the selected studies. As Figure 2.5 shows, a total of 38 primary studies use temporal logic, rewriting logic, other first-order logics, or combinations of these logics.

The use of temporal logic to model requirements as logic formulas can be interpreted

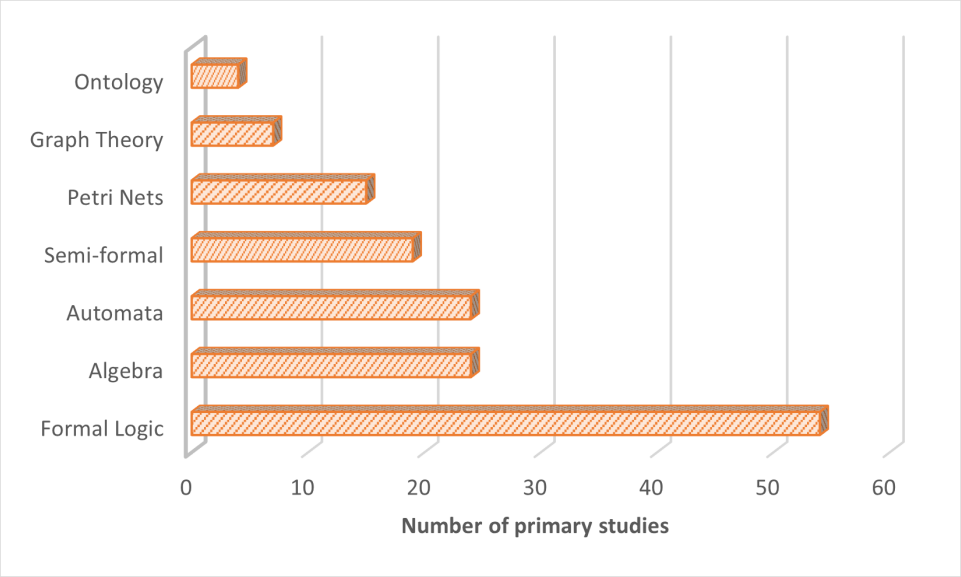

Figure 2.4: Formal methods used for requirement modeling

over time in 38 of 54 studies. Temporal logic-based requirements interlace time and events using temporal operators built over other logic systems, such as first-order logic. Such requirements are pervasive in complex distributed systems (Fisher, 2011). It is also observed that temporal logic finds most use within Cyber-Physical, IoT and real-time systems.

First-order logic is also applied in other studies, such as in **S[10]** where the logic is used as a part of ANSI C Specification language to annotate C source code for verification purposes. Higher-order logic is used for WSN, smart grids and aircrafts systems in **S[136], S[166]**, and **S[189]** respectively. Other works, such as **S[47], S[56], S[59]** and **S[93]**, use rewriting logic to model system requirements. Rewriting logic can be used to express concurrent processing as well as logical deduction (Meseguer, 2012). **S[47]** proposes reusable and frequently occurring *formal patterns* for distributed systems using rewriting logic as a semantic framework. A novel approach using the Xtune formal framework has been discussed in **S[56]** for modelling and specification of time/quality of service (QoS) properties of mobile embedded systems using concurrent rewriting logic. **S[59]** and **S[93]** use real-time Maude specification language based

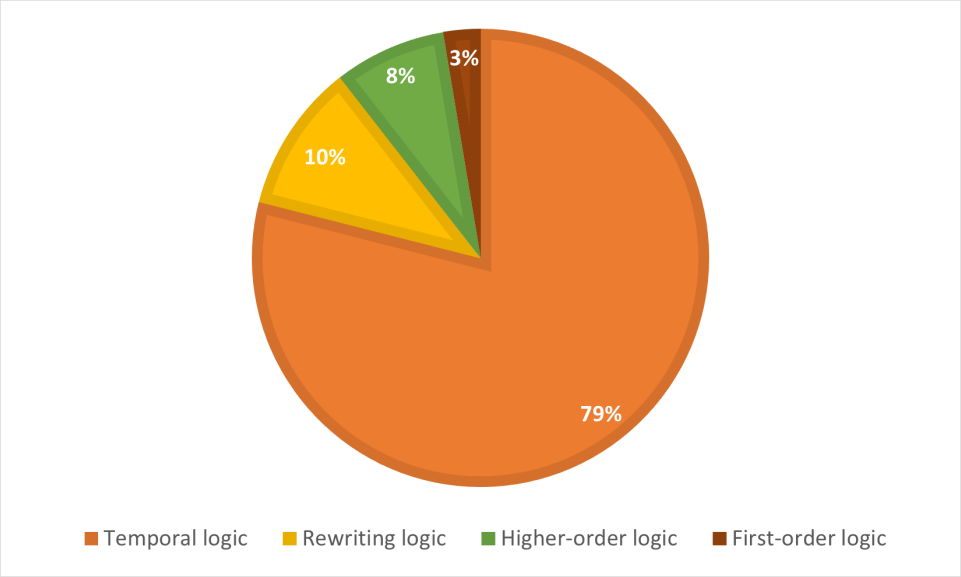

Figure 2.5: Types of logics used

on rewriting logic for QoS and timing properties of distributed real-time embedded systems.

2.2.6.1.2 Algebra Based Techniques

Algebraic techniques have been used to formally model requirements in 24 studies. Evidence suggests that most algebraic methods have been used sporadically with a few exceptions, such as Process Algebra. Process algebra provides formal description or specification of concurrent or parallel processes (Aceto, Ingólfsdóttir, Larsen & Srba, 2007). For requirements modelling, it has been used in conjunction with other formal techniques such as temporal logic or automata theory. Communicating sequential processes is a variant of process algebra that has been used in **S[49], S[76]**, and **S[152]** to specify temporal requirements of distributed real-time systems. **S[87]** proposes a high-level specification language called Active sensor processes derived from process algebra to specify and verify sensor networks formally. Other useful but sparsely used descendants of process algebra include π-Calculus **S[13], S[9]**, fluent-Calculas **S[169]** LOTOS for specifying abstract behaviour of multimedia stream (**S[78]**), PEPA

as a modelling language for specifying concurrent protocol such as Alternating Bit Protocol (ABS) (**S[78]**), E-Lotos standard based GRL domain specification language for modelling and Globally Asynchronous, Locally Synchronous (GALS) systems (**S[2]**).

2.2.6.1.3 Automata

FMs based on automata theory are used by 24 primary studies. Modern-day computer science has benefited from foundational work done in the theory of automata in the mid 20th century. Its close relationship with formal grammars makes it favourable to formally describe a model of a system or requirement as a state machine (Hopcroft, Motwani & Ullman, 2001). Large distributed systems are composed of large requirements sets. Automata-based methods enable us to model the relationships between system or component states. In 16 studies (**S[1], S[11], S[19], S[89], S[90], S[91], S[94], S[114], S[120], S[127], S[155], S[160], S[163], S[165], S[175]** and **S[181]**) timed automata are used to model the behaviour of real-time systems over time (Alur & Dill, 1994). Timed automata-based specifications are used to formally specify controllers, wireless communication protocols, asynchronous distributed systems for model checking and verification purposes using automated tools. Evidence suggests that timed automata scores favourable points for requirements verification due to tool support, such as the UPPAAL model checker. Cellular automaton (Sarkar, 2000) is used to model CPS in three studies i.e. **S[50], S[52], S[19]** and **S[53]**. Other related frameworks such as message passing automata and constrained automata have been used in one study each.

2.2.6.1.4 Graph Theory

Graph theory is another formal technique adopted for system modelling being related closely to automata and mathematical logic. For instance, graphs can be defined as automata (Sakarovitch, 2009). Also, the use of graph theory for expressing relationships between components of automata is argued in (McIntosh, 2009). The use of graph

theory in four primary studies also includes other formal modelling techniques. Studies **S[5]** and **[**S[131] use graph theory to generate with Vienna Development Method Specification Language. In **S[30]**, graphs are used in conjunction with Event-B modelling language to layout a formalisation process of wireless networked systems. **S[66]** discusses algebraic formalisation of the Ravenscar Computation Model (Burns, Dobbing & Vardanega, 2004) based on graph theory to prove safe termination in compliant systems. Finally, study **S[89]** uses Object-Based Graph Grammar for formal specification and verification of real-time systems. An extension of this grammar for modelling time constraints and a subsequent transformation of such constraints into timed automata is proposed.

2.2.6.1.5 Petri Nets

A Petri Net (PN) is a bipartite graph and a formal mathematical modelling tool. Its graphical representation enables visualisation of state changes in systems containing parallel processing during run-time. Consequently, they are used to model event-driven distributed computer systems (J. Wang, 2007). PNs have been used for requirements modelling. Seven studies (**S[4], S[57], S[72], S[73], S[81], S[92]** and **S[180]**) use some form of PNs mainly in real-time event driven systems including: satellite-based and terrestrial train control system, vehicular ad-hoc networks (VANET), Industrial manufacturing control systems, brake subsystems and pulse oximeters. In **S[4]**, authors present an approach that synthesises the different scenarios depicted as UML sequence diagrams into a hierarchical Colored Petri Net model. The apparent advantage of this approach is that it allows early-stage model checking and state-space analysis. Similarly, **S[57]** uses coloured PNs for formal simulation and verification of safety communication protocol in the European Train Control System to satisfy safety requirements. Studies **S[171** and **S[185]** also use coloured PNs for sensor networks. Both **S[4]** and **S[57]** use ASK-CTL model checker for verification of operational requirements. Similar

to the **S[4]**, **S[72]** also uses UML MARTE sequence diagrams to describe system requirements and maps them into Timed coloured PNs with Inhibitor Arcs for formal analysis purposes. The study has implemented the proposed approach in a VANET-based case study to validate the solution. **S[73]** works with SysML and MARTE activity diagrams and transforms them into timed PNs to formally validate the requirements in early SDLC as SysML lacks formal semantics. The study presents a case study of a pulse-oximeter that determines a patient's blood oxygen levels. Study **S[92]** uses PNs with temporal logic formulas to formally model and verify a generalised railway crossing platform by modelling a system controller using coloured PNs. At the same time, requirements are specified using temporal logic formulas.

2.2.6.1.6 Semi-Formal Techniques

An emerging pattern that can be identified from studies **S[4], S[57]** and **S[73]** is the use of semi-formal techniques in tandem with formal techniques such as PNs. The transformation of semi-formal models to formal PN models has certain advantages, as it reduces the time taken to develop complete formal models. UML and SysML have widely adopted modelling techniques better understood by system designers and developers than FMs due to their close relatedness to industry design standards. Moreover, it is not always possible to formally describe all parts of a system and its requirements, especially in large distributed systems. For this reason, lightweight approaches have become more useful in requirements and system modelling at the higher or system level. At the same time, more formal techniques are being used more selectively at the component level (Fitzgerald et al., 2013). Hence, semi-formal methodologies provide a balance between rigour and effort in modelling, paving the way to use traditional methods and FMs. For requirements modelling, 19 primary studies (**S[21], S[22], S[23], S[25], S[31], S[36], S[43], S[82], S[100], S[105], S[110], S[114], S[120], S[124], S[157], S[181], S[183], S[195]** and **S[201]**) use a framework

Table 2.2: Semi-formal techniques in primary studies

Primary Study	Adopted semi-formal techniques
S[21]	Development of semi-formal Requirement Specific Language (RSL) (RESA)
S[22]	Semi-formalization of requirements specification, development of semi-formal RSL
S[23]	Requirements description with SysML, UML/MARTE profile for component view
S[25]	Automatic generation of safety mechanism from semi-formally specified safety requirement using RSL
S[31]	Transformation of requirements in natural language to semi-formal notation
S[36]	Semi-formalism with the use of UML MARTE and SysML
S[43]	Use of EmotionML and SysML to specify cognition and affectivity requirements
S[82]	Formal extension of UML with timing annotations (OMEGA kernel language)
S[100]	Semi-formalization using SysML-Sec environment
S[105]	ASIL tailoring process for requirements analysis phase using SysML
S[110]	Semi-formalization of SoC requirements using SysML
S[114]	Extention UML and SysML for contract-based theory
S[120]	Semi-formalism with the use of UML MARTE
S[124]	Use of SysML-based CESAR requirements meta-model (RMM)
S[157]	Use of JSON as requirements modelling
S[181]	UML extension with timed-automata model chekcing
S[183]	Semi-formal Simulink specifications for CPS
S[195]	SysML security requirements specification for CPS
S[201]	UML/MARTE transformation into Event-B specifications

with semi-formal techniques as the front-end and provide support to transform these models automatically into formal models. Table 2.2 summarises the use of semi-formal techniques in the selected primary studies.

2.2.6.1.7 Ontology

An ontology is a conceptualisation of entities, their attributes and their relationships. Entity Relationship Diagrams (ERD) or UML class diagrams are forms of ontologies. Ontologies can be highly informal, semi-formal or rigorously formal (Uschold, Gruninger et al., 1996). Domain-specific formal ontologies are used for requirement modelling by four studies. Study **S[3]** converts requirements specified in natural language into more structured forms using formalised ontologies for further analysis. The general idea is to extend an initial requirements ontology during subsequent phases of the V-process model (Forsberg & Mooz, 1991). **S[21]** presents an ontology-based

semi-formal requirements specification language for automotive systems using an automotive domain ontology. Study **S[67]** presents a framework of several FMs such as formal ontologies, ambient calculus, ambient logic, real-time temporal logic and a design-by-contract technique to specify formal contracts. These methods are then applied at different stages of the development of wireless and pervasive health care applications. For the structural modelling process, this study uses a formal ontology to specify all entities involved in the environment, which is then converted into a variant of ambient calculus.

2.2.6.2 Formal Requirements Specification Languages

This section presents a map of formal specification languages. Requirements specification languages (RSLs) provide a structured approach to writing requirements, often through templates and patterns. RSLs can be informal, such as guided natural language, or can be highly formal. This survey reveals the commonly encountered formal RSLs in the surveyed works.

2.2.6.2.1 Architecture Description Languages

Requirements (especially non-functional) and a system's (software) architecture are interdependent. Requirements such as performance, throughput, security, scalability are often modelled in and addressed within the high-level architecture (Bourque et al., 1999). Earlier architectural descriptions carried in the manner of box-and-line diagrams were hugely inconsistent and provided no support for formal analysis. During the last two decades, researchers have overcome these limitations through formal architectural descriptions using Architecture Description Languages (ADLs) (Garlan, 2000).

ADLs can help document architectural views that define components, their relationships and primary requirements. The mapping shows that ADLs have been used in 15 studies. Initially developed for avionics systems, Architecture Analysis and

Design Language (Feiler, Gluch & Hudak, 2006) has also found use in Transport and Cyber-Physical systems consisting of real-time embedded systems. Seven such instances (**S[6], S[7], S[17], S[49], S[50], S[52], S[53]** and **S[179]**) of AADL use were found. Although many ADLs have been proposed over the years, none were able to consolidate their standing in the industry (Pandey, 2010). On the other hand, UML has now been considered as the de facto standard to document architectures. Table 2.2 refers to the use of UML amongst the surveyed primary studies.

2.2.6.2.2 Temporal Logic

There are several formal RSLs derived from temporal logic. A total of 24 studies (**S[8], S[26], S[31], S[34], S[41], S[44], S[45], S[54], S[55], S[70], S[77], S[101], S[116], S[121], S[126], S[129], S[137], S[140], S[145], S[152], S[167], S[180], S[188]** and **S[194]**) use Linear Temporal Logic (LTL). LTL allows reasoning over sequences of system or component states called computational paths (Huth & Ryan, 2004). In **S[8]**, authors argue that LTL falls short in adequately describing cyber-physical systems constraints. Therefore, a new language FORM-L used in **S[121], S[143]**, and **153** is introduced to describe temporal constraints that closely relates to LTL. In **S[77]**, LTL is used to specify the requirements of the system while adaptation-based LTL (Zhang & Cheng, 2006) helps specify requirements for the adaptive parts of the system. **S[55]** extends LTL into reliability annotated LTL with special operators to specify reliability requirements in safety-critical embedded systems.

Computation Tree Logic (CTL) presents a branching notion of time such that one of several possible successor states may be reached from any state. The superset of these logics is CTL*, which provides much higher expressiveness by allowing any sequencing of temporal and path quantifiers. A total of 10 primary studies (**S[4], S[19], S[31], S[41], S[55], S[57], S[59], S[94], S[102]** and **S[104]**) have used CTL or its variants for requirements specifications, especially for transport and avionics systems. ASK-CTL is

a formal method to express state-space models. Studies **S[4]** and **S[57]** have used ASK-CTL mainly for simulation and model checking purposes, while Timed-CTL (TCTL) has been used in studies **S[19]** and **S[31]**. Other variants of CTL such as Probabilistic CTL (PCTL) and CTL* are used in one study each.

Figure 2.6: Formal specification languages

2.2.6.2.3 Communicating Sequential Processes

Contemporary real-time distributed systems are highly concurrent due to the critical nature of operations performed by them. Communicating Sequential Processes (CSP) is a formal language based on a calculus that deals with the communication of concurrent

processes (Roscoe, 1998). CSP has found its application in seven (**S[6], S[35], S[49], S[65], S[76], S[87]** and **S[96]**) primary studies. Amongst these, four studies have used Timed CSP (TCSP), which allows timing properties to be specified for real-time systems. Like TCSP, a pattern can be seen to extend a language or modelling technique, e.g. timed automaton, timed PNs or TCTL, to fit in the timing aspects of real-time distributed systems.

2.2.6.2.4 B Method and Z Notation

B method and Z notation are related formal, model-based RSLs derived from set theory. Z is an established formal technique in academia, taught in universities for some decades (Yap & Holcombe, 1997). A comparison of both approaches in terms of object orientation, concurrency, tool support, and industrial applications has been reported in (Kaur, Gulati & Singh, 2012). The study finds that the main differences between Z and B are the level of abstraction, with Z being able to provide a higher level of abstraction than B. On the other hand, the B method can generate C code directly, while Z notation does not provide this function. Moreover, Z can help design object-oriented system designs, a capability that the B method lacks. Studies (**S[35], S[65], S[83], S[85], S[87]** and **S[147]**) apply Z notation as a formal RSL. Amongst these, study **S[65]** uses Object-Z (Smith, 2012) which is an extension of Z notation that includes object-oriented features. Studies **S[65]** and **S[87]** use Timed Communication Object-Z to integrate TCSP and Object-Z to get the best of both worlds - it blends the concurrency-rich features of Timed CSP with the object orientation of Object-Z.

B method has been used in studies (**S[27], S[30], S[38], S[97], S[118], S[148], S[174], S[187], S[197], S[198]** and **S[201]**). Event-B is derived from B method and is used for modelling purposes. Studies in **S[30], S[97]** and **S[118]** before 2017. A trend has been seen to prefer Event-B over B method in recent years. A significant amount of studies (**S[125], S[133], S[148], S[158], S[172], S[174], S[179], S[185], S[190],**

S[192], S[193], S[197], S[198], and **S[201]**) can seen be applying Event-B after 2017. It is important to note that although B and Event-B are related languages, they are applied in different contexts. The B method is mainly used for formal specification and validation of requirements, while Event-B is a system-wide modelling tool.

2.2.6.2.5 Domain Specific Languages

Domain-Specific Languages (DSLs) have been introduced to accommodate domain-specific requirements formally. Although DSLs have limitations in wider applicability, they provide for repeatable use within their domains. Moreover, they are intuitive for domain experts who may not have the necessary knowledge to use general specification languages. This study found at least 10 studies (**S[8], S[21], S[22], S[25], S[62], S[66], S[71], S[87], S[144]** and **S[174]**) which propose or extend a DSL. **S[8]** presents FORM-L mentioned earlier in this section. ReSA (**S[21]**) is an ontology-based requirement specification language built for the automotive domain. It uses requirements modelled in EAST-ADL, which is an extended ADL for automotive real-time embedded systems. Also, in study **S[22]**, a new DSL using Eclipse Xtext open-source framework has been defined, and a proof-of-concept implementation for automotive systems is presented. **S[65]** designs a DSL for the formalisation of industrial standards with a case study to implement their approach in European standard EN 1591. Figure 2.6. shows the overall distribution of formal specification languages amongst primary studies.

2.2.7 Classification based on Requirement Types and RE phases

This section refers to the following research question:

SM-RQ 2: What formal methods target functional and non-functional requirements of DES?

According to the Software Engineering Body of Knowledge (SWEBOK) (Bourque

et al., 1999), a software requirement can be either functional or non-functional. Functional Requirements (FR) are the features that a software product will incorporate, while Non-Functional Requirements (NFR) deal with the qualitative aspects of the software. This section categorises the surveyed works based on their focus on FRs or NFRs. Figure 2.7 provides this categorisation. 63% of the works target NFRs, 23% target FRs while 14% are unclassified. For the others, a breakdown based on NFRs type is shown in Figure 2.8. 27% of the primary studies targeting NFRs do not provide any further classification.

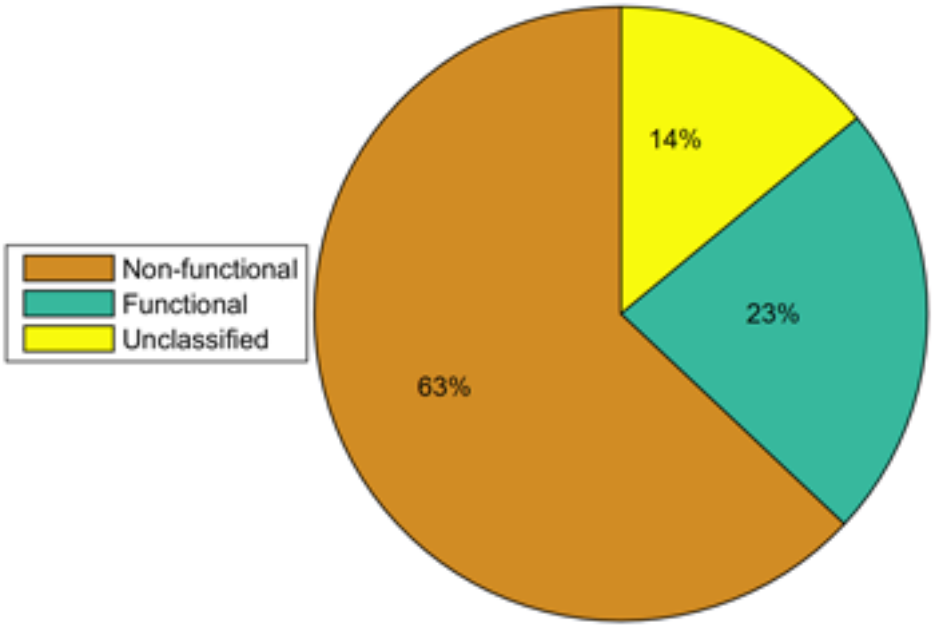

Figure 2.7: Distribution of targeted requirements by types in primary studies.

2.2.7.1 Classification Based on NFR Types

Figure 2.8 shows that a majority of the effort in the surveyed works targets NFRs and that most of this effort has been focused on safety and timing requirements. Timing properties and safety requirements are tightly coupled together in DES as the safe operation of a system are often time-critical and vice versa. Temporal logic forms the basis of most of these works. Thirty primary studies use temporal logic solely

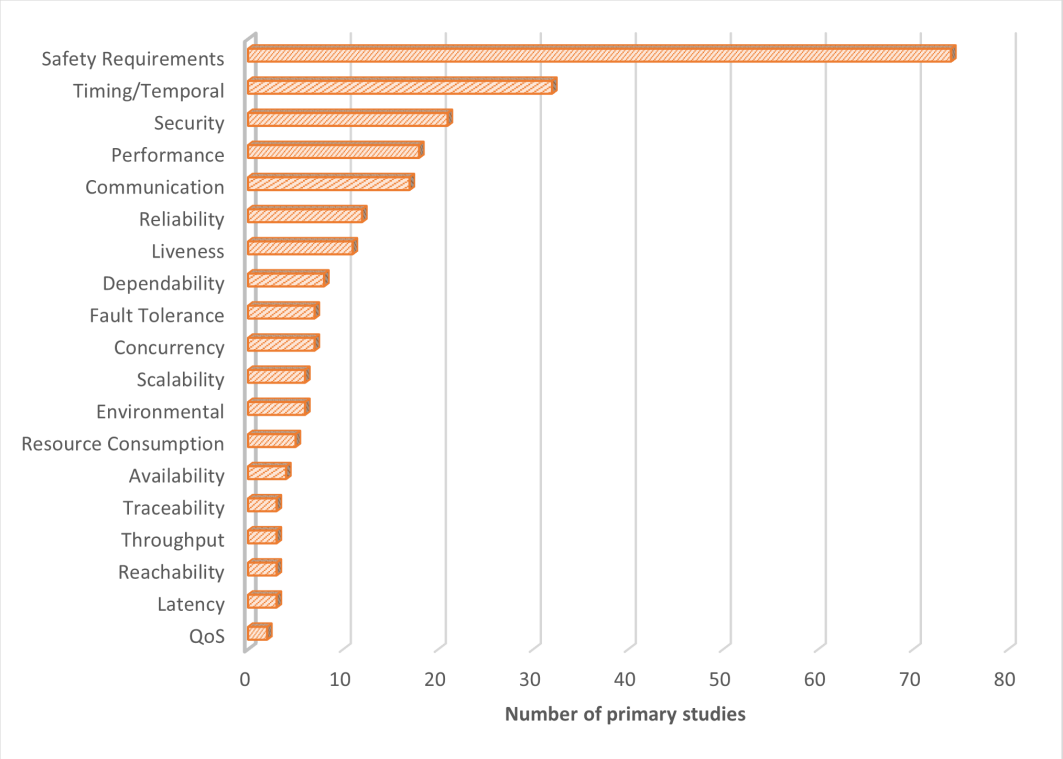

Figure 2.8: Distribution of Non-Functional requirements

for specifying timing requirements. This study now discusses commonly encountered requirement types.

2.2.7.1.1 Safety Requirements

A majority of DES finds application in the cyber-physical domain where they control physical processes. From industrial automation to transportation systems to avionics, embedded controllers command and control many physical components. The safety of such systems is critical, and safety requirements have been emphasised in 74 primary studies. Figure 2.9 shows the distribution of safety requirements according to the DES sub-domains. Unsurprisingly, cyber-physical systems, in general, have been the focus of most studies. Automotive, railways and avionics appear next in this list, followed by IoT, industrial and traffic control systems.

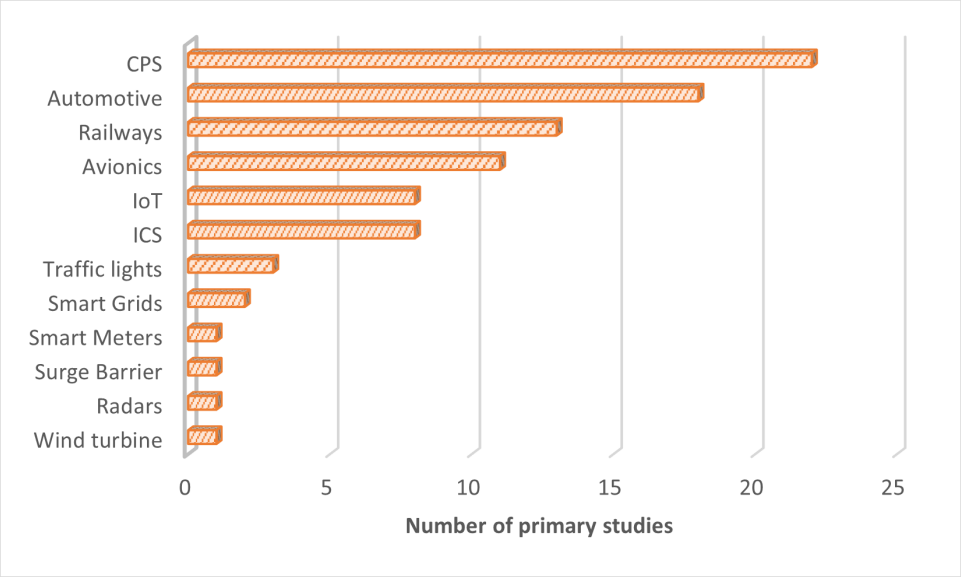

Figure 2.9: Classifications of works targeting safety requirements in DES

2.2.7.1.2 Reliability

Reliability refers to the ability of a system to remain functional for an extended period under specified operating conditions. The reliable operation of DES, especially in the industrial domain, is a critical requirement in most cases. However, given the cyber-physical nature of many DES, the operating environment is often too large and complicated to capture, reliability requirements challenging to verify. Formal support for specifying and verifying reliability requirements across various DES but mainly in Automotive and transportation systems is found in 12 primary studies.

2.2.7.1.3 Security

Security is another critical aspect of DES, gaining increasing importance in DES as they form and interact with IoT and Industrial IoT systems. DES rely heavily on internal and external communication between components and systems, and each communication link is at the risk of being attacked. There is a wide belief that security should be an integral part of the requirements and design process rather than an add-on. Although security measures taken in distributed systems have increased with time, attackers

are always finding new ways to breach the systems forcing researchers in this area to evolve continuously (Andreeva et al., 2016). The researcher considers that it is rather unexpected to find only 21 studies providing solutions to security requirements especially considering the pool size (201) of primary studies. In fact, only 10 studies (**S[125], S[127], S[128], S[150], S[161], S[164], S[170], S[173], S[195]**, and **S[197]**) focus on the security of DES in last four years that is a significant concern.

2.2.7.2 Classification Based on RE Phases

Software Engineering Body of Knowledge (SWEBoK) (Bourque et al., 1999) is used to classify the primary studies concerning phases in requirements engineering. The following phases are chronologically ordered: *Feasibility Study, Requirements Elicitation and Analysis, Requirements Specification, Requirements Validation*. Each phase of the process takes inputs and produces specific outputs to be consumed in the next phase. These outputs are essential artefacts that are helpful in later parts of the software development life-cycle. Figure 2.10 categorises the surveyed works based on the RE phases they target.

Requirements elicitation involves requirements discovery by consulting with stakeholders. From a formal RE point of view, the elicitation process is not as imperative as proceeding RE stages as stakeholders often describe requirements in natural language, e.g. in (**S[1]**) and **S[31]**. Overall, eight studies have addressed the requirements elicitation phase as part of their solution, including the *structured object-oriented security requirements framework* framework in **S[84]**.

A conceptual model for the requirements of a real-world problem can be developed for further analysis during the requirements analysis phase. These models can contain holistic requirements or chunks of them, comprising real-world components of a system and their relationships in the respective domain. Formal analysis is also a part of requirements analysis that affects requirements specification and model validation

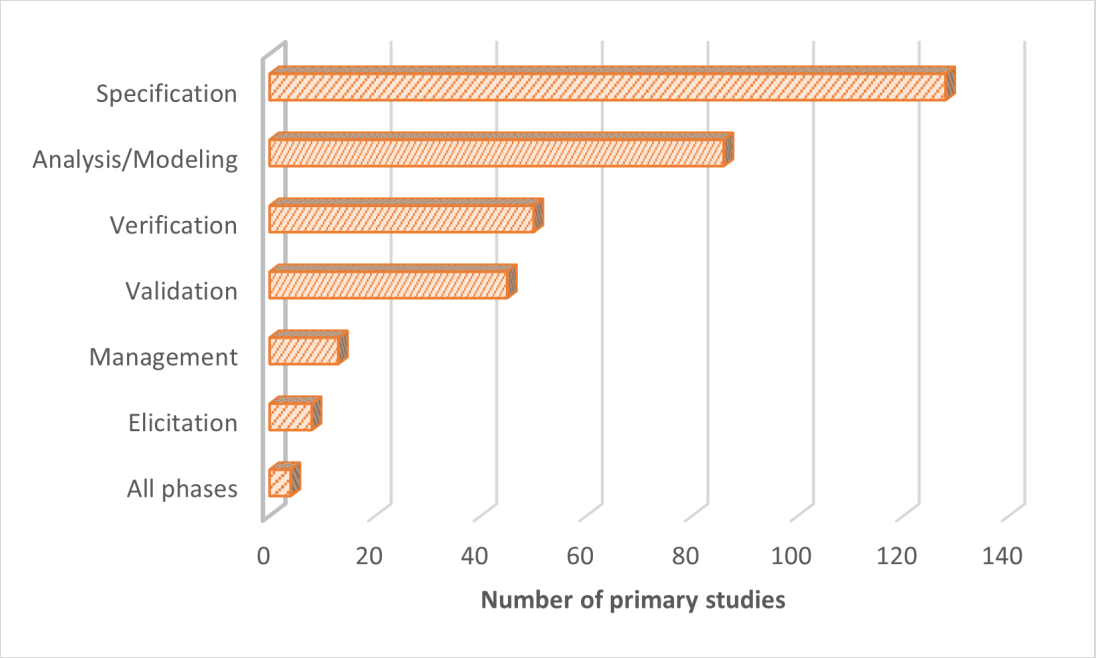

Figure 2.10: Classifications of works according to RE phases

phases (Bourque et al., 1999). The main focus in the analysis phase is put upon formal modelling that has been applied in 86 primary studies. A detailed account of formal modelling methods is discussed in Section 2.2.6.

During requirements specification, requirements are usually specified in natural language and sometimes in semi-formal forms. In safety-critical DES, where it is imperative to eliminate ambiguities in the specifications, selecting appropriate formal modelling tools or specification languages depends on the nature of the system and environment. This SMS finds that 128 primary studies span over the specification phase, a result that is not unprecedented due to the following reasons: *1)* Formal RE process is mainly dependent on the ability to specify or even modelling of requirements in a formal way. There is no additional step defined in RE that is only specific to induce formality *2)* Focus on formal modelling and specification tools can cause an unintentional bias with researchers that can affect the research methodology adoption. Section 2.2.10 discusses such factors in detail.

The ultimate reason for formal requirement specification is the eventual validation and verification process. The scope also goes beyond into implementation phase resulting in trusted use of requirements. Requirements validation deals with checking requirements according to validity, consistency, completeness, realism, and verifiability. The idea of validating requirements before design and implementation can be useful in the fact that the later an error is detected in the software development life-cycle, the more the time and the cost it will incur (Sommerville, 2011) to repair. Formal specifications also enable the validation process to be automated hence more reliable due to tools that help in automated validation. Verification and validation process on formal models or specifications has been applied in 50 and 45 primary studies, respectively. Most of these studies have taken advantage of automated tool support in the validation and verification process. Model checkers are amongst the tools that are extensively used, especially in the verification process. Although it resides at the boundary of the scope of this study, it is still imperative to be able to judge the correctness of the system according to the requirements. A model checker can verify the finite state systems concerning the provided specifications. UPPAAL (Behrmann et al., 2006) and SPIN (Holzmann, 1997) model checkers have found the most use according to this review. The other model checkers that come up regularly in the solutions and discussions are NuSMV (Cimatti, Clarke, Giunchiglia & Roveri, 1999), ASK-CTL (Van der Aalst & Stahl, 2011), ProB (Leuschel & Butler, 2003), DiVinE (Barnat et al., 2013) and Kronos (Bozga et al., 1998).

2.2.8 Bibliographical Mapping

This section answers the following research question:

SM-RQ 4: What type of publications constitute the realm, and which forums have been more active in the last decade?

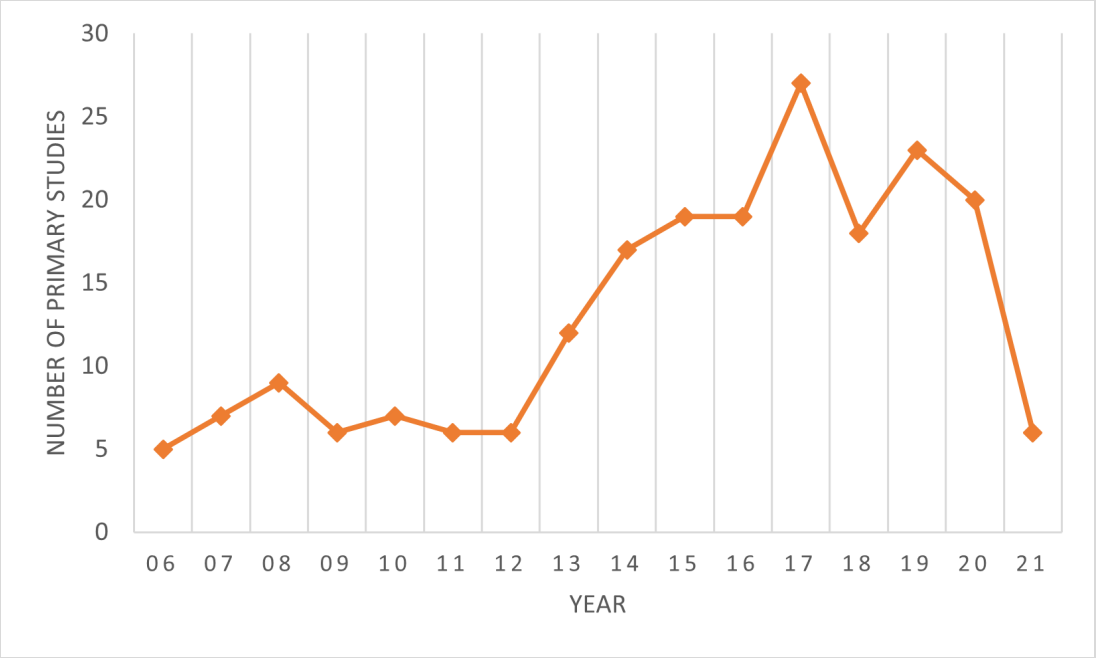

Figure 2.11: Articles published by year

Figure 2.11 presents the distribution of primary studies by publication year. The early years, from 2006 to 2012, saw very few studies published within our scope. From 2012, there has been a steady rise in numbers, with 2017 being the summit point. The figures for 2021 are lower because this survey covers only those articles that appeared online by the cut-off date of April 2021. With accelerating research and industrial interests in IoT (Gubbi et al., 2013) and CPS (Evans & Annunziata, 2012), it is expected to see even more research articles being published in this area in the future.

Concerning the qualitative value of primary studies, Figure 2.12 refers to the number of articles according to publication types such as conferences, journals, and workshops or symposiums. The author has constrained the scope to only peer-reviewed papers published in reputed venues. About half of the studies (95) were published in conferences showing that most works presented works with a relatively lower level of maturity. On the other hand, 19 journal articles corroborate consolidated and admissible research ideas and solutions. As there is no definite global standard to evaluate the quality of a

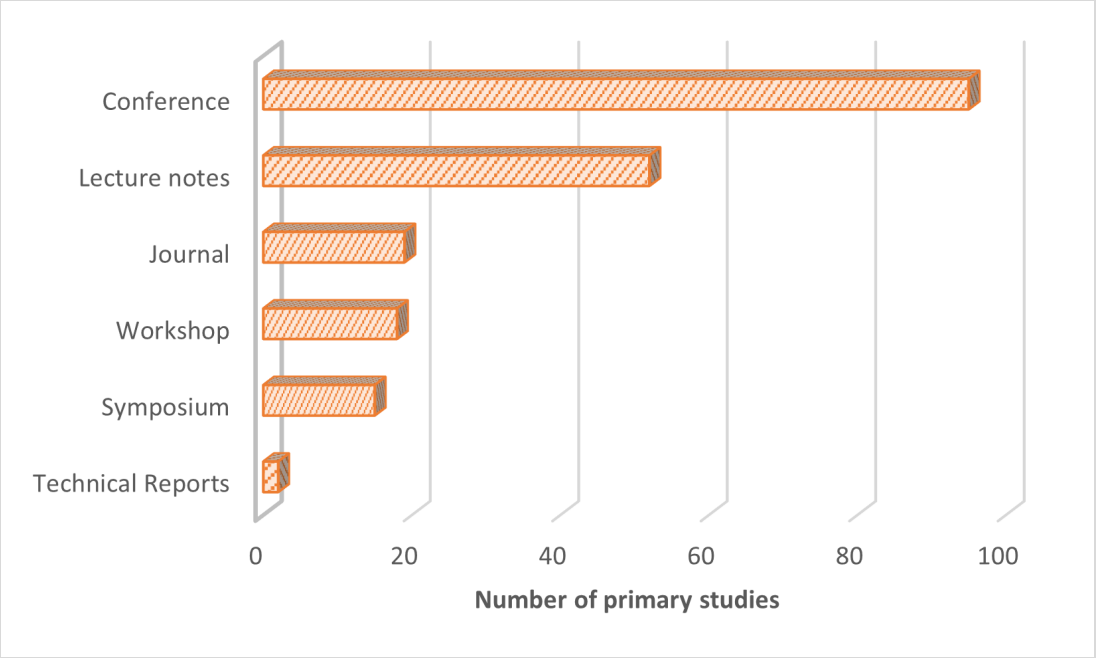

Figure 2.12: Articles published by venue type

publishing venue, assessing them is very subjective.

The research type classification scheme in this SMS is derived from (Wieringa, Maiden, Mead & Rolland, 2006). It divides requirements engineering papers into six categories and lays down criteria to sort articles into these categories. It is the relevant classification scheme for this study as the research scope also envelopes the requirements engineering domain. Research type classification of primary studies is depicted in Figure 2.13 in the form of a Venn diagram.

Figure 2.14 for research methods and the experimental setups carried out in the primary studies. A distinction worth mentioning is between validation and evaluation research. Papers classified as evaluation research tend to assess established techniques used as an implementation strategy in the industry. In contrast, validation research endorses new solutions that have not secured their place amongst practitioners. The use of different research methods and experimental setups (Petersen et al., 2015; Wieringa et al., 2006) shown in Figure 2.14 in the studies helped us in deciding the type of the

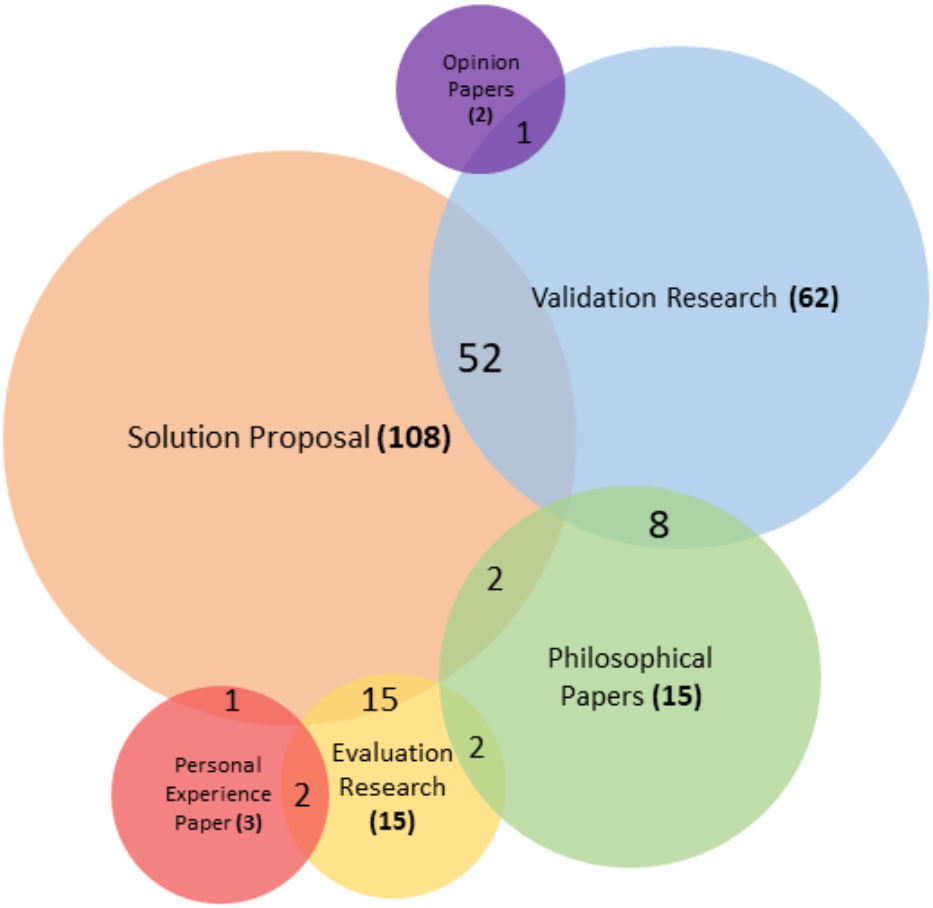

Figure 2.13: Number of articles by research type

research.

Solution proposal papers contain novel ideas or techniques that are not fully validated. Figure 2.13 shows that 38 studies presented new solutions with minimal experimental evaluation. Including overlaps, there were 178 solution proposals in total, which is an overwhelming number in the context of this study. A considerable number (52) of articles were characterised as solution proposals as well as validation research, which is quite acceptable as a solution proposal study can present proper validation (Wieringa et al., 2006). New conceptual and theoretical frameworks were presented in 27 studies classified as philosophical papers. Ten of these studies either validate or evaluate these frameworks, thus leaving 17 studies in the category describing new

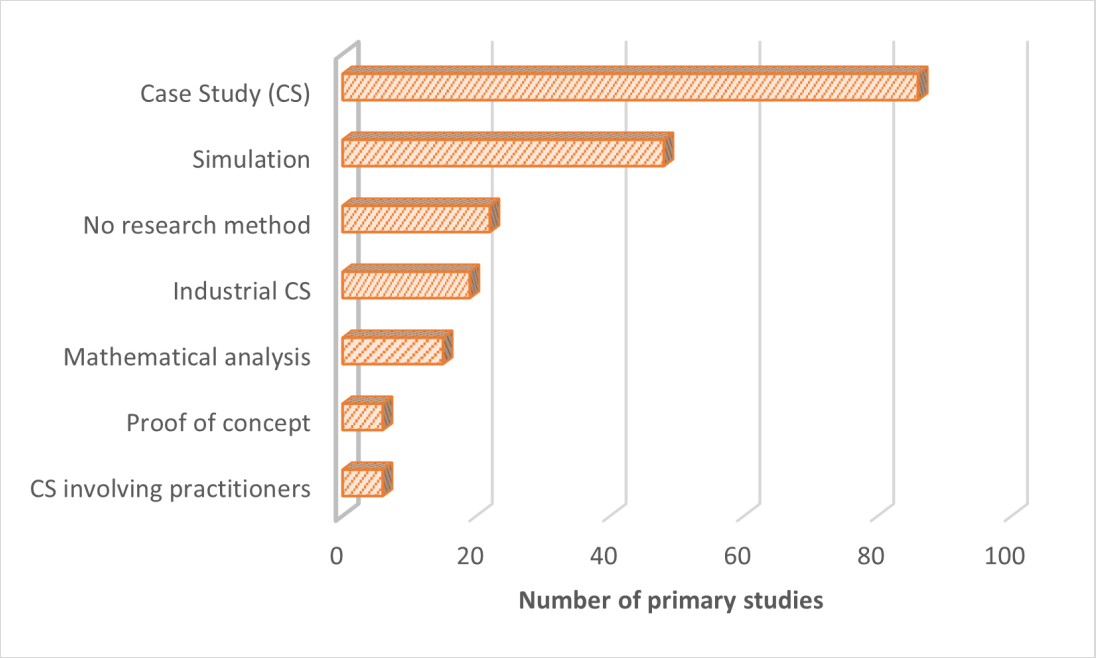

Figure 2.14: Research methods and experimental setups

concepts. There were 62 articles confined to validation research only, although the total number of articles classified in this category is 123.

2.2.9 Discussions

This section answers the following research question:

SM-RQ 3: What are the latest research trends and the most investigated aspects regarding the formalisation of RE in DES? What are the research gaps and the latest challenges?

Formal methods are used for validation and verification purposes throughout the software development life-cycle of large-scale DES. Their use for the RE stage can be invaluable, as early analysis and rigour can significantly help reduce ambiguity, inconsistencies and incompleteness of requirements. In addition to providing a contextualised synopsis of the findings, this section synthesises the data from the results presented earlier to identify research trends and assessed gaps according to the research questions

raised in Section 2.2.4.

Figure 2.15 presents the heat map showing the clustering of articles containing a combination of sub-domains of FM, DES and NFRs. On the x-axis, the sub-fields of DES are represented. Similarly, on the y and z-axes, sub-fields of NFRs and FMs are represented, respectively. There is no specific order in the arrangement of axes labels other than the found articles related to that sub-field. The heat map depicts the overall patterns and gaps in the scope discussed in this section. The usefulness of heat map is that it shows the sufficiencies and voids of the research area in a single view, with the caveat that this map is open to multiple interpretations depending upon the researcher's point of view and biases, as discussed later in Section 2.2.10. The fact that the maximum number of studies found in a particular combination (e.g. CPS, safety requirements and formal logic) is only six suggests that although the area may have a maximum number of studies, it still needs more attention. The author expects a constructive discussion on this subject matter in future research. Apart from that, some of the pertinent research patterns that deemed important are discussed next.

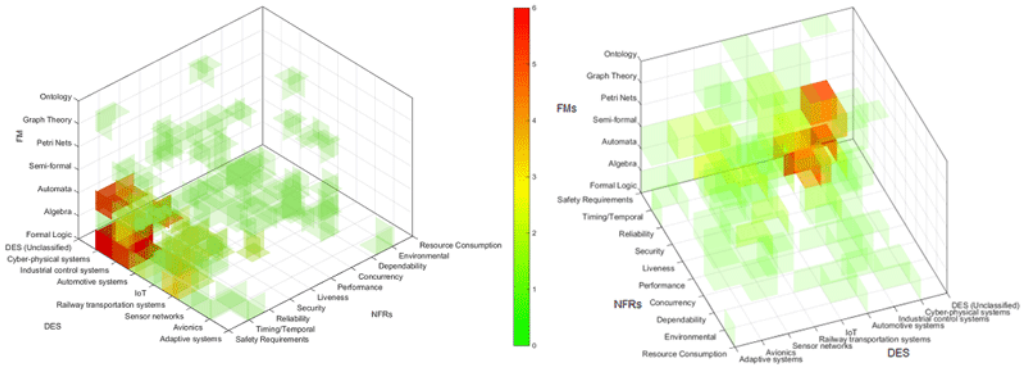

Figure 2.15: Multiple views of a heat map showing the concentration of primary studies.

2.2.9.1 Lack of Industrial Application of Formal Specification Languages

Several studies have focused on the requirements specification of DES. A requirements specification is a technical artefact intended to help designers and developers for verification purposes. Similarly, it can be helpful at later stages of SDLC, such as to make testers aware of certain conditions under which a system should pass a test (Hierons et al., 2009). Formal requirements specification and analysis/modelling phases have overlapped with vast concentrations of studies which give a promising indication that solutions have spanned for most essential phases of the RE spectrum. It is observed that the emergence of new specification languages and extension of legacy languages have made this area gather fragmented efforts, which now needs consolidation. Similar concerns are discussed in (Martins & Gorschek, 2016; Sepúlveda et al., 2016). The study postulates that emerging formal languages need to show empirical evaluation evidence, which is only possible when the industry practitioners apply them. The absence of such evaluation denies the opportunity for new solutions to mature, and hence they can fade away into oblivion quite quickly. Lack of education of FMs amongst software engineering students is also one of the obstacles to their adoption. Organisations have to induct FM specialists that increase the overall project cost. Inadvertently, they become a tool only to be employed in large organisations working on large-scale industrial projects with enough resources needed by others (Bjørner & Havelund, 2014). Reasonable criteria to measure the reach of a new language in academia and industry can be the citation count. We find that the six studies that present the new languages have a total citation count of 6.

2.2.9.2 Concerns on Formal Security Requirements Engineering in DES

The lack of security requirements engineering is a noticeable area of concern because of a deficient number of related studies in the overall sample size. DES, especially CPS,

IoT and ICS, are composed of heterogeneous components distributed across networks. These components constantly need to be in communication for I/O operations through commonly agreed interfaces.

A significant gap concerning security applications in the DES domain is the lack of studies providing modern solutions for ICS. For instance, the IEC-61499 (Vyatkin & of America, 2007) standard for distributed industrial automation systems provides function blocks as basic components that can be distributed across multiple Programmable Logic Controllers (PLC) within the same application. This standard has been used in four studies (**S[58], S[81], S[132]** and **S[160]**) for modelling and specification of ICS. In a distributed IEC-61499 application, function blocks communicating over a communication network (usually, TCP/IP stack) regularly need to send I/O parameters. An attacker can disrupt services by taking advantage of communication layer vulnerabilities. Moreover, PLC is a low-level device often devoid of inherent security mechanisms. Considering the low number of studies regarding ICS, there is still a need for formal security frameworks for IEC-61499 based applications providing a full range of cryptographic services. Further guidelines for security management in ICS are provided in (Stouffer et al., 2011).

The importance of security in ICS has been highlighted in the literature. Stuxnet, which specifically attacked PLCs in the Iranian nuclear infrastructure, is a significant example showing the extent of the damage possible if such devices are improperly secured (Langner, 2011). In an opinion, (Wagner, 2016) cites a report by (Andreeva et al., 2016) that shows that more than 90% of hosts connected to Industrial control systems have vulnerabilities ready to be exploited. The author advocates the use of security standards and the need for security certification to provide cryptographic services in ICS. The FIPS 140-2 (NIST, 2016) standard that defines minimum security requirements to validate a cryptographic device can also be considered for assistance in ICS devices.

IoT has been the most consolidated area for the formalisation of security requirements compared to other DES technologies. One of the rationales behind this phenomenon can be the abundant research work in internet security, yet; security requirements are not the primary focal point for most studies. Security in IoT is focused in 10 primary studies. The study **S[13]** is unique in proposing a new architectural platform for a novel Rail Internet of Things (RIoT) with its security management as a significant area of concern. The RIoT system consists of trains, tracks, stations, passengers and a control centre for management. Communication security between components or "smart things" is highly desirable due to their limited computing capabilities. The potential attacks and vulnerabilities on RIoT devices, such as side-channel attacks, privacy abuse, DOS (Denial of Service), denial of sleep, wireless signal jamming and presence of malicious devices in the network, are discussed along with their mitigation solutions. Discussions on security concerns of IoT systems are trivial and insignificant in other studies in which the terms "IoT" and "security" are encountered. Moreover, the SMS does not find novel security framework or innovations for Industrial IoT in recent years.

Another significant pattern is the scarcity of selected primary studies that deal with security and CPS. The concern stems from the ever-increasing use of CPS in safety-critical ventures. The studies **S[49], S[68], S[127], S[154], S[182]** and **S[195]** discuss CPS security. The study **S[68]** proposes a solution that primarily deals with the formalisation of security requirements for smart meters in the context of smart grids. Such grids are considered a combination of physical components (such as smart meters, generators, distributors, solar panels, storage) networked with cyber systems through actuation and sensing technologies. Cybersecurity is a huge challenge in a smart grid environment where reliance on public communication networks such as telecommunication networks is likely to put sensitive data prone to cyber-attacks. The attacks that come in the form of unauthorised access, DOS, and the network latency during peak hours may cause the collapse of the whole smart grid infrastructure affecting

millions of people (Yu & Xue, 2016). Therefore, this SMS suggests an increased focus on formal solutions for smart grids is required.

2.2.9.3 Emergence of Novel Formal Frameworks in IoT

An interesting pattern of the emergence of many novel frameworks in the IoT domain was mentioned briefly in Section 2.2.5. This research trend can be attributed to the fact that IoT is still a new but rapidly growing paradigm. It is estimated that nodes in IoT may reach the trillion mark in the next decade, given that IPv6 is now set into full motion (Gubbi et al., 2013; N. K. Singh et al., 2016). There is a need for established frameworks or standards to solve diverse problems in this domain. For instance, the need to develop the new framework for optimising bandwidth when the number of sensor nodes increases in a network is discussed in (N. K. Singh et al., 2016). The other challenges that are needed to be addressed are privacy and security for communication between sensor nodes in such networks. Moreover, (Gubbi et al., 2013) provides the vision and motivation for IoT, deliberates on the same lines and discusses the viability of several new frameworks that will help the more established technologies to be integrated with the IoT. They also present a conceptual IoT framework with cloud computing at the centre.

The study **S[9]** pitches an integrated framework of FMs for interaction behaviours of IoT-based industrial equipment. It consists of two layers, i.e. specification and verification. π-Calculus is used for specification rules while verification is performed through NuSMV model checker. A translator tool (piCal2NuSMV) was developed to transform π-Calculus to NuSMV according to the definitions and rules specified in the paper. Study **S[29]** is referred for a novel testing framework for Extensible Messaging and Presence Protocol (XMPP) communication protocol extensively used in IoT devices for data exchange. Another theoretical framework for formal requirements modelling in pervasive computing systems is presented in **S[75]** and **S[158]** proposes a framework

to integrate model design with formal methods for embedded software development.

Although the overall numbers of studies for the formalisation of the IoT domain are pretty low, the percentage of new frameworks targeting the formalisation process can be considered an encouraging factor for the IoT future.

2.2.9.4 Shortfall in Sensor Networks Research

Sensor networks are one of the main branches of DES that is in the scope of this mapping study (refer to Figure 2.1). The combination of sensor networks and the internet has given rise to the IoT (D. Singh, Tripathi & Jara, 2014). Research on IoT has now engulfed the sensor networks domain. However, standalone applications using sensors have still their applicability in areas such as detection of earthquakes, nuclear reactors, agricultural industry, military applications (Ebenezer & Murty, 2015; Katyara, Shah, Zardari, Chowdhry & Kumar, 2017; Pawgasame, 2016; R. Wang et al., 2016). Such critical application areas for sensor networks should emphasise FM research because only 15 primary studies out of 201 related to this field. However, the areas of applications found in these studies are pretty distinctive, i.e. ICS, automotive, network protocols, radars, robotics and environment monitoring.

2.2.9.5 Emphasis on Formalisation in Automotive industry

The automotive industry has taken long technological strides during the last two and a half decades to such an extent that driverless cars are now the talk of the future (Schedel, 2008). Safety being the key concern in these systems, various manufacturers now seek to incorporate new safety features in their vehicles. As more and more such features involve using embedded systems, evolved risk levels are expected (Briciu & Filip, 2014). In the quest for safer modern vehicles, the automotive industry has agreed to adopt the ISO-26262 Road vehicles — functional safety (Czerny, D'Ambrosio & Debouk, 2002) standard. It deals mainly with the safety requirements of electrical and electronic

(E/E) systems in road vehicles, but it does not account for threats arising from non-E/E systems, e.g. fire, smoke, heat) in a vehicle.

A promising trend is seen amongst a significant amount (27) of selected primary studies in the automotive domain that rely on ISO-26262 for formal specification, analysis and verification of safety requirements of components such as Fuel Level Display (FDL) in trucks and Turn indicator system. Moreover, the standard also advocates the use of semi-formal specifications and analysis of such requirements. AUTomotive Open System ARchitecture (AUTOSAR) (Fürst et al., 2009) has been coupled with the ISO-26262 standard to achieve scalable implementation of safety requirements. In addition, it also deals with performance and usability properties in automotive vehicles. It provides a comprehensive open software architecture that must be implemented to make automotive software portable and transferable amongst vehicles of leading industry manufacturers.

2.2.10 Threats to Validity

Several factors can affect the accuracy of the findings of a systematic SMS. The following section describes particular limitations, assumptions and factors that may threaten the validity of this SMS.

The SMSs are intended to cover the breadth of a research area. As mentioned in the earlier section, the mandate of SMSs is to quantify the relevant entities, unlike systematic literature reviews where the main objective is to analyse the current work in the field regarding quality. The search string is devised in such a way that it could return the maximum amount of papers from online databases to cover the breadth of the area. As a result, more than 3500 studies were collected for further investigation. The first criteria to exclude a study was by title. It can be assumed that some relevant studies might have been missed considering that title of the study is a minimum amount of information.

Abstracts can also misinform a reader if they are poorly written. The standard SMS process requires a reviewer to snip by reading titles, abstracts and keywords. To mitigate issues arising from these constraints, the introduction, conclusion and in some cases, the internal sections of a study to find any missing information from titles and abstracts were also read.

The number of selected primary studies is another factor that can affect the findings as the systematic mapping process aims towards quantitative analysis. A smaller sample size can cause the loss of significant trends that may have formed the research gaps ultimately. On the other hand, a bigger sample size can make it difficult for researchers to put things into perspective, risking the resulting data becoming redundant. Trivial patterns may be taken as important ones, while the non-trivial ones may get lost due to the "choice overload" phenomenon (Iyengar & Lepper, 2000). The size of the primary studies' population also depends on the scope and research area. In this SMS, the scope of the research spans over three distinct but related domains and many areas needed to be covered. The author has tried to keep the sample size in check due to time limitations due to which it is possible to speculate over optimum pool size, considering the vast research area this study had to cover.

Researcher bias is another factor that may have affected the validity. The researcher can carry certain biases when dealing with several variables. For example, it can be caused by the strengths of a researcher in a specific aspect of the research area, or he/she may weigh one variable more than others. Multiple researchers working on the same study may elevate this problem further. Especially during the early stages of article screening, bias may affect the selection of a paper by reading abstracts, introduction and conclusion sections. Regular consultation meetings between the supervisors were carried to resist this problem. Domain experts are also consulted for validation of the study after each milestone step, e.g. the development of research questions, the formation of the search string and especially before the final selection of the primary

studies.

Another factor that must be discussed is the focus on CPS, IoT and Sensor networks within the domain of DES. The researcher tried to introduce a balance between legacy DES and CPS, IoT and Sensor Networks. It affected the formation of the search string, resulting in a study output that may cause divergence from legacy DES. It can also be assumed that the exclusion criteria of selecting studies published in/or since 2006 also affected the tilt of results towards newer paradigms since they have been the target of most of the research in the formal methods domain during this period.

Researchers' expertise is also a potential factor to the validity when considering the research scope (Figure 2.1) that is an overlap of three domain areas. Over the years, industry dependability on formal methods has increased, resulting in the introduction of a wide variety of formal methods that are also evident from the results of this study. Although the knowledge of this area is mandatory, it is inconceivable for a researcher to be an expert at most of them, considering that education on the subject is still an obstacle to the growth of formal methods (Bjørner & Havelund, 2014). The same is the case regarding the DES domain and choices of sub-domains. The immunity against this validity threat has come from the nature of the systematic mapping process, i.e. it does not focus on qualitative aspects of the selected primary studies; instead, researchers may still survey the research area without the comprehensive knowledge.

2.2.11 Conclusion

Large scale DES are rapidly growing in size and numbers and defining new boundaries using advanced communication technologies and novel heterogeneous components. The complexity and decentralised nature of modern-day DES demands a formal way to manage the requirements engineering process to reduce the ambiguity and confusion related to requirement specification. Research activity around this problem area has

surged with the advent of new paradigms such as IoT and CPS. Numerous formal methods have been proposed during the last few years, but using correct formal methods in a given scenario is still a problem for practitioners.

In this section, an SMS is conducted to use formal tools and techniques in the requirements engineering process of DES. Based on 201 selected primary studies, the detailed bibliographical mapping and topic-dependent classifications in the research scope are presented. In addition, a detailed discussion on significant patterns and research gaps was carried out in the Section 2.2.9, also with the help of an elaborative heat map (Figure 2.15). The primary concerns were found related to the lack of industrial applications of formal specification languages and limited research on formal methods for ICS and industrial IoT. Apart from security, the emergence of novel frameworks in IoT is a promising trend.

2.3 Survey on Security Engineering in ICS

A significant research gap in the systematic mapping study is the lack of formal security requirements engineering in ICS. This section aims to survey the current state of the literature regarding SRE and its integration with the design and implementation of ICS.

2.3.1 ICS Security Requirements Engineering

This section reviews the current state of work in the domain of Security Requirements Engineering (SRE) in ICS and related fields. The need for SRE and the current challenges are discussed below.

An integrated safety and security requirements elicitation method for CPS called "SafeSec Tropos"" is proposed in (Kavallieratos et al., 2020). This method is based on SRE-based Secure Tropos (Mouratidis et al., 2016). It contains five stages: *scope*, *view*, *description*, *validation*, and *security and safety co-analysis*. The work focuses on

early integration of safety and security requirements that are inter-related properties of ICS. Although the method is a viable solution to ensure the validity of requirements for CPS within ICS, it is focused on the analysis of requirements traceability back to the objectives. As a result, the SafeSec Tropos method lacks in forward traceability of requirements to the design and implementation stage.

A security requirements specification method for CPS that can also be applied in ICS is proposed in (Varela-Vaca et al., 2020). The method uses a feature model catalogue for CPS to validate and diagnose device configurations regarding security requirements. It represents the security requirements in JSON, also categorising the requirements regarding security features such as *constraints*, *security level conditions*. The requirements verification is automated using feature models contained in a catalogue. The proposed approach is restricted to verification of already implemented requirements. Moreover, it is unclear how the correct verification of cryptographic primitives can be achieved using this method. Limited details are provided to ensure confidentiality in high-level security requirements verification is listed in the case study, e.g. the measures are taken to ensure correct implementation of an encryption algorithm.

A process to derive security requirements based on the impact analysis is proposed for industrial IoT in (Hansch, Schneider & Brost, 2019). First, the process identifies the assets and their security needs in a system using domain-specific language. It uses the assessment to derive and categorise (based on security goals) the security requirements using graphs. The process also proposes detection and monitoring of requirements fulfilment using a catalogue. The monitored logs can analyse the requirements based on a specific set of rules to raise violation alerts. The challenge with this approach is its treatment of devices in industrial systems as a block box. While the process can validate a device's compliance when added to the network, it lacks the verification of correct requirement implementation during the development.

In (Hansch, Schneider, Fischer & Böttinger, 2019), an integrated architecture for

open platforms communications in industrial IoT is presented. The solution uses OPC-UA compatible security requirements specification to communicate and compare requirements in diversified technological devices in ICS. The architecture model divides the requirements into *technical*, *functional*, and *functional* requirements types.Using the unified model, the devices in ICS can communicate their security requirements and capabilities using OPC-UA client-server architecture. However, the possible implications of automatically communicating security requirements are not clear. For instance, OPC-UA is a resource-intensive protocol that might not be suited to resource-constrained devices in ICS (Mahnke, Leitner & Damm, 2009). Moreover, the proposed model is focused on requirements communication rather than SRE.

The lack of security requirements integration with the design and implementation of ICS applications is evident in few studies. An approach to validating security requirements after the deployment of a device in ICS is reported in (Faily, Iacob, Ali & Ki-Aries, 2020). A comparison between STRIDE and eSTRIDE to derive security requirements in ICS on a risk and threat analysis is presented in (Tuma et al., 2021). A similar approach to specify security requirements based on threat modelling for industrial IoT is discussed in (Jabangwe & Nguyen-Duc, 2020). In (Kivelä et al., 2018), a risk reduction assessment and reduction process based on security requirements from the security standards is proposed. Similarly, (Zhou et al., 2020) also presents an overall risk-based unified architecture for ICS security requirements. Another early-stage security requirements specification model that does not extend the requirement's use to the ICS design and development phase is presented in (Vistbakka & Troubitsyna, 2021).

The application of formal security requirements in CPS is also another area of concern in this research. (Fuchs, Gürgens & Rudolph, 2010) proposes a solution that primarily deals with the formalisation of security requirements for smart meters in the context of smart grids (SG). SGs are considered a combination of physical components (such as smart meters, generators, distributors, solar panels, storage)

that are networked with cyber systems through actuation and sensing technologies. Cybersecurity is a considerable challenge in a smart grid environment where reliance on public communication networks such as telecommunication networks can leave sensitive data prone to cyber-attacks. The attacks may come in the form of unauthorised access, DOS, and network latency during peak hours, which may cause the collapse of the whole smart grid infrastructure affecting millions of people (Yu & Xue, 2016). The security Modelling Framework used in (Fuchs et al., 2010) presents formal definitions of security requirements based on the European Union's Measuring Instruments Directive 2004/22/EC (MID).

(Preston, Duck, Finnegan & Munoz, 2019) suggests the benefits of integrating cyber-security requirements from the security standards into the design of power management systems. It also suggests the collaboration between vendors and users to link the user's requirements with security standards early on in the project to support the cybersecurity security strategy in power management systems.

(Fernández, Kantarcioglu & Thuraisingham, 2016) provide a formal secure logic-based framework to collect and manage a large amount of data streaming from IoT devices. The framework is a category-based layered meta-model. The first layer is governing data collection, and the second and third layers are mainly for storing and sharing data. Rewriting logic is used for policy specification and analysis. (Koundinya, Sharvani & Rao, 2016) also provides a framework for calibrated security measures for IoT-based ICS with the focus on smart grids.

A Conceptual framework for SRE is presented in (Fabian et al., 2010). This framework classifies SRE methods mainly in four segments, i.e. stakeholder views, system requirements, specification & domain knowledge and threat analysis. It then compares the proposed framework with other SRE approaches by mapping the varied terminologies of other SRE methods to their framework's segments. Compared methodologies include Multilateral security requirements analysis, SRE methodology, UML-based approaches

such as secureUML and UMLsec, Goal-oriented approaches, problem frame-based approaches, risk analysis-based methods and CC-based approaches. Further details of comparison can be examined in the referred study.

ICS can also be categorised as Cyber-Physical Systems (CPS), as they contain software components running on computational nodes (PLCs) controlling physical processes. A systematic mapping study by (P. H. Nguyen, Ali & Yue, 2017) on Model-Based Security Engineering for CPS (MBSE4CPS) shows that of all CPS-related studies published in the last 15 years, most are academic case studies, resulting in a significant gap between industry and academia. There is little emphasis on explicitly addressing security concerns like confidentiality, integrity, and availability. They argue that it could be because of missing the issue of integration between security and CPS. Being a new field, the main focus of MBSE4CPS research is upon requirements and domain analysis. As CPS are highly distributed and heterogeneous, the introduction of security requirements may introduce some uncertainties into the system behaviour. (P. H. Nguyen et al., 2017) argue that they have found little evidence for analysing and managing such uncertainty in CPS. Therefore, the research community should spend more effort in resolving such issues. They also debate that the research is focused more on security analysis based on threats, attacks or vulnerabilities. Overall, they report a severe lack of security engineering solutions for the development of CPS.

2.3.2 ICS Security Standards

The notion of security is contextual to a system's operating environment and its stakeholders. Security requirements vary with each system type, and there is no single definition of absolute security. Large requirements sets are often difficult to manage and prioritise in industrial systems, leading to missing requirements. Security standards put forward sets of best practices that should be followed to achieve the maximum possible

security. The advantages of following a standard ensure that the system implements a rigorous, well researched and published set of security requirements. Security standards provide the basis for system-wide assurance, risk assessment and mitigation through policies and frameworks. For instance, (Ani, Watson, Green, Craggs & Nurse, 2020) recommends the use of security standards to evaluate the reliability of an ICS security testbed.

(Srinivas, Das & Kumar, 2019) expresses the importance of cybersecurity standards in improving the efficiency of key processes and facilitate systems integration and interoperability. However in case of ICS, (Rogowski, 2018) suggests that the main focus of the ICS research is the application of safety standards and that the security has been neglected.

A state-of-the-art evaluation of the security standard landscape in industrial IoT is presented in (Dhirani, Armstrong & Newe, 2021). It reviews the convergence of security standards across IT and ICS domains and states the current cybersecurity challenges in industrial IoT cybersecurity. The authors find that improper understating and implementing commercial security standards that are not suited to ICS lead to vulnerabilities in communication security. It is because hybrid standards do not specify environment-specific guidelines that can lead to incompatible security implementation. Therefore, ICS must use related standards providing security specifications suited to current innovations in the domain.

A 2016 ICS vulnerability report by (Andreeva et al., 2016) shows a grim picture regarding ICS security. Their principal findings show that the number of exploitable vulnerabilities in ICS components are growing with time. Most of the vulnerabilities (49%) found are of a high-risk level, and 5% of the vulnerabilities found in 2015 were yet to be patched at the time of publishing this report. ICS components available through the internet use insecure protocols such as Http, Telnet, SNMP, etc. Citing (Andreeva et al., 2016), (Wagner, 2016) raises concerns about the state of hardware

security and the need for appropriate measures to be taken by vendors. The author highlights the role of cryptography in securing hardware devices, but the ever-changing landscape of ubiquitous computing demands a more comprehensive approach. It includes robust architectures and frameworks with security requirements integrated "by-design". The author proposes fundamental guidelines for businesses to protect their hardware modules. One of the significant recommendations is to have a system of security certification that must ensure compliance with minimum security and legal requirements before deploying the hardware or software components.

In addition, a comprehensive survey on cybersecurity management in ICS has been undertaken (Knowles et al., 2015). It provides an extensive list of information security standards and guidelines. It also outlines security standards, guidelines and best practices that are specific to ICS. In this study, standards are categorised regarding cross-industry publications, e.g. oil and gas, chemical and energy. Further analysis of control system publications is carried out by an exhaustive criterion that includes: risk management, asset identification and classification, threat assessment, vulnerability assessment, risk level evaluation, the recommendation of countermeasures, change management, etc.

Reliance on security standards should become obligatory in the opinion of the experts. For example, (Litherland, Orr & Piggin, 2016) raises the alarm related to security in nuclear plants. The authors argue that even though the international atomic energy agency establishes the importance of nuclear safety and nuclear security, there is no agreed standard set of policies and procedures amongst the licensees. There is also a misconception that safety will automatically result in adequate security. Therefore, there is a need to invest in resources to enhance assessment, management and assurance of security requirements. To implement adequate security measures by the nuclear licensees, the authors offer guidelines and a set of good practices in light of international standards such as NIST SP 800-82, IEC-62859, IEC-62645.

ISO/IEC-15408, also known as Common Criteria (CC), is a worldwide recognised framework for evaluating information technology security products. Although it does not guarantee absolute security, it assures that the system is evaluated rigorously. The importance of CC certification gets even more highlighted due to specific legal bindings, i.e. National Information Assurance Acquisition Policy, NSTISSP No. 11 (USA) requires only certified CC security products to deploy in US governmental infrastructure. It also affects modules within ICS.

Process Control Security Requirements Forum (PCSRF): a group of professionals formed in 2001 whose main objective was to specify security requirements for the process control system. They produced a system of various Protection Profiles (PP) for ICS. A system protection profile (Control, Process and Requirements, Security and Bond, Digital, 2006) produced for CC mentions the security requirements for field devices such as PLCs and RTUs for Supervisory Control and Data Acquisition (SCADA) systems in medium robustness environments. This PP requires that if Target Of Evaluation (TOE) (Field devices) implements cryptographic functionality, it must comply with FIPS 140-2 requirements.

In (Parvez, Ali, Ahmed & Farhan, 2015), authors present a framework for implementation of the AGA-12 standard for SCADA systems in the oil and gas industry. Their implementation flow is composed of risk assessment, description of security goals, compliance of hardware, compliance of software, cryptographic algorithm compliance, implementation and post-implementation audits. They have also compared AGA-12 with other security standards against technical specifications and coverage areas.

(Beckers, Faßbender, Heisel & Schmidt, 2012) maps the CF above to ISO-27001 standard (Calder, 2013). They show the relation of CF terms with ISO-27001 terms and relate some sections of ISO-27001 with SRE methods. The authors present similar work to ours but within different scope. It provides the extension of SRE methods to be compliant with ISO-27001, a security standard for Information security management

systems. The research also presents a practical application of their work that integrates the SRE methods to the ISO-27001 implementation process. Carrying forward the research mentioned above in (Beckers, 2015), the author has contributed to relations from the previous study. In addition, this study discusses the ISO-27001 documentation requirement for compliance and how SRE methods can confirm the production of such documents. Both pieces of research are specific to the ISO-27001 security standard. Therefore, there is a need to explore frameworks that can integrate SRE methods with ICS security standards.

Security certifications based on standards are the way to move forward to develop inherently secure modules. Security standards have been used in the industry for a long time. Industrial control systems and especially automation systems, are at an early stage of evolution. Hence standards are also in a re-alignment process for the needs of ICS. Several standards are now targeting ICS and related control system architectures. Table 2.3 shows the list of some of the relevant established security standards for ICS.

Table 2.3: Security standards relevant to ICS

Security Standard	**Title**
NIST SP 800-82	Guide to ICS Security (Stouffer et al., 2011).
IEC-62351	Information Security for Power System Control Operations (Schlegel et al., 2017).
IEEE 6189 (AGA-12)	AGA - Cryptographic Protection of SCADA Communications
FIPS 140-2	Security Requirements for Cryptographic Modules (NIST, 2016).
IEC-19790	Security techniques – Security requirements for cryptographic modules (Kusumah & Andriawan, 2019).
IEC-62443	Industrial Network and System Security (62443-1-1, 2009).
ISO/IEC-15408	Common Criteria for Information Technology Security Evaluation (Mellado, Fernández-Medina & Piattini, 2007).

2.3.2.1 Current Works in Security Standard Compliance

The abstract and extensive nature of requirements in security standards challenges identifying and extracting specific requirements for a particular system. In a standard certification process, multiple stakeholders are involved in checking the compliance of security requirements for a product. The stakeholders include vendors and testers who work together to validate and verify a product's compliance with a particular security standard. These stakeholders follow independent processes regarding validation and verification where coordination and understanding of each other's views can be challenging. The researcher discusses some of the current works regarding security standard compliance.

A novel approach of integrating IEC 62443-4-1 security standard into ICS Continuous Integration/Continuous Delivery (CI/CD) pipelines is presented in (Constante et al., 2021). The approach also describes how to integrate the security requirements with the security tools in an automated way. Each stage of the security standard compliant pipeline, such as *plan, code, build, test, release, deploy* is compatible with IEC 62443-4-1 practices such as *security requirements, secure design, secure implementation and security verification and validation testing (also including third-party verification)*. The approach is promising regarding building standard-compliant ICS products; however, it is limited to mapping CI/CD activities to security standards. It is an umbrella approach that does not discuss an individual activity's details, e.g. integration of security requirements with the secure design.

A state-of-the-art framework for Monitoring And Standard Compliance Verification (MSCV) is presented in (Bicaku et al., 2021). The framework acts as a measurement tool for automated and continuous security compliance for the System of Systems. The framework is divided into three distinct phases. The first phase is the identification of requirements, relevant security standards and measurable metrics. The second

phase involves the monitoring of the measurable metrics identified in phase one. The third phase is the delivery of results containing standard compliance classification and visualisation of results. The framework proposes integrating the Eclipse Arrowhead framework to verify the real-time compliance of a device at the time of its connectivity to the cloud. The degree of compliance of a device based on seven foundation requirements of ICS specific IEC 62443 standard is shown in the results. The overall framework is promising for ensuring standard compliance; however, it is limited to the onboarding stage, i.e. a device can be accepted or rejected from the cloud-based on its compliance. It does not provide a process to develop standard complying ICS devices or applications from scratch.

A formal framework of compliance verification for cloud-connected SCADA is presented in (Kulik, Tran-Jørgensen & Boudjadar, 2019). The study uses IEC 62443-3-3 ICS security standard as a target compliance standard. The system behaviour along with the security requirements of IEC 62443-3-3 are modelled using LTS. The formal verification of the requirements regarding the standard is enabled using the TLA+ language. The framework can verify the qualitative properties such as security and liveness requirements of the system; however, it does not provide the ability to verify the functional requirements. Moreover, it is also limited to the deployment stage to verify a device as a black box. It lacks the functional requirement verification that is necessary for security certified devices.

An IEC 62443 compliant threat analysis model is discussed in (Fockel et al., 2019). The authors present a relationship between threats and standard's requirements based on security levels. The study proposed a tool-based threat analysis approach integrated into the application development process. This approach is a multi-stage process that involves system architecture specification, threat analysis and countermeasure requirements. The study proposes to specify countermeasure requirements as a general security requirement to integrate threat analysis into development; however, it lacks

further details and implications. For instance, the method to link a countermeasure requirement to its mitigation implementation for traceability purposes, is not explained.

In (Ehrlich et al., 2020), a *Sec4ICS* tool to model security properties and architecture using IEC 624443 standard structure is presented. The tool analyses the model produced by TOSCA modelling language to obtain the application network topology and the security level of the components, and the required security level regarding IEC 62443 foundation requirements. It can report whether the component is compliant with the required security level from the standard. The study proposes inducing security into design and implementation; however, it is not clear how it can be achieved using their approach, which is restricted to evaluation based on IEC 62443 security levels.

Challenges to the automated mapping of security requirements in ICS are discussed in (Ehrlich et al., 2018). The paper points to the lack of security standard conformance for the legacy devices for communications. It is because legacy devices were not designed to match the increasing demands of communication in the current industrial landscape. Security modelling metrics are required in industrial automation to describe the application requirements and capabilities.

(Baldini et al., 2016) highlights the current challenges in security certification of IoT products. The authors have mainly discussed Common Criteria and its issues when certifying products. Issues such as certification time and costs, re-certification for a new version of the product, incompatibility of user requirements with certification requirements and requirements traceability etc., have been highlighted. This paper introduces new security certification methods specifically designed for IoT products. This process is based on formal model-based testing. It introduces a new concept of labelling, which identifies the level of security assurance of the product.

A comprehensive survey of cybersecurity certification for IoT is presented in (Matheu et al., 2020). It proposes a certification framework to alleviate the challenges such as standardisation, harmonisation with semantic specification and, the need

for the support for certificate life-cycle for an IoT system. However, the framework focuses mainly on risk assessment and security testing.

2.3.3 ICS Secure Design and Implementation

A secure-by-design methodology—ARMET for behaviour-based ICS is proposed in (M. T. Khan et al., 2017). The methodology uses a deductive synthesis of the executable specification to apply security properties in ICS applications. The authors also propose monitoring and verification techniques of executable specifications. Although the methodology helps secure-by-design ICS application development and its verification, it focuses on non-function properties of the system, such as security and safety. Lack of emphasis on function requirements leads to a black-box system with the obscure implementation of security.

A state-of-the-art literature review on the current IEC 61499 ICS application development standard is carried out in (Lyu & Brennan, 2020). The authors outline three major issues. These include *1)* Industrial concerns, *2)* Technical issues, and *3)* Societal aspects. A significant issue is the redesign of IEC 61131 legacy ICS applications with modern IEC 61499 standard. The study argues that the research is still required to enforce new design paradigms to ICS architecture modelling.

IEC 61499 can be used to alleviate the problem of distribution within ICS components by automated security requirements mapping techniques (Ehrlich et al., 2018). Distributed systems, especially Cyber-Physical Systems (CPS), IoT and ICS, are composed of heterogeneous components distributed across the system. These components constantly need to be in communication for I/O operations through commonly agreed interfaces. For example, the IEC 61499 Standard (Vyatkin & of America, 2007) for developing distributed industrial automation systems provides function blocks as reusable software blocks that can be distributed across multiple PLCs within the same

application. (Hirsch, Missal & Hanisch, 2008) and (Nicholas, Bhatti & Roop, 2012) have used IEC 61499 for modelling and specification of ICS. In a distributed IEC 61499 application, function blocks communicating over a communication network (usually TCP/IP stack) regularly need to send I/O parameters. An attacker can disrupt services by taking advantage of communication layer vulnerabilities. Moreover, PLC is a low-level device often devoid of inherent security mechanisms. (Homay & de Sousa, 2016) proposes a solution by adding a Message Authentication Code (MAC) layer for assurance of message integrity between IEC 61499 compatible devices but other important aspects of security such as confidentiality and authentication are still desired. Furthermore, there is still a need for formal security frameworks for IEC 61499 based applications providing a full range of cryptographic services.

ICS, when seen in such a larger picture, is CPS using aspects of the IoT paradigm (Fernández et al., 2016). Therefore, it can be assumed that solutions produced for the IoT issues can be more relevant to ICS and be adapted after minor adjustments. An interesting pattern has been observed related to the concentration of novel frameworks for IoT systems. It is estimated that nodes (or the interconnected "things") in the IoT will reach the one trillion mark in the next decade, given that IPv6 is now fully available (Gubbi et al., 2013; D. Singh et al., 2014). It also serves as evidence that there is a need for established frameworks or standards to solve diverse problems in this domain. For example, (D. Singh et al., 2014) discusses the need to develop a new framework for optimising bandwidth when the number of sensor nodes increases in a network. The other challenges that need to be addressed are privacy and security for the communication between sensor nodes in such networks. Moreover, (Gubbi et al., 2013) deliberates on the same lines and discuss the viability of several new frameworks that will help more established technologies to be integrated with the IoT.

A study regarding industrial IoT (Eiza et al., 2015) propose an architectural platform for a novel Rail Internet of Things (RIoT) with security management as a significant area

of concern. The RIoT system consists of trains, tracks, stations, passengers and a control centre for the management. Communication security between components or ”smart things” is highly desirable due to their limited computing capabilities. The potential attacks and vulnerabilities on RIoT devices such as side-channel attacks, privacy abuse, DOS (Denial of Service), denial of sleep, wireless signal jamming and the presence of malicious devices in the network are discussed along with their mitigation solutions. However, this methodology also focuses on non-functional security properties for RIoT such as run-time vulnerabilities and latency in resource-constrained devices.

(Deng, Ren, Yuan, Chen & Hua, 2015) presents an integrated framework of FM for interaction behaviours of IoT-based industrial equipment. It consists of two layers, specification and verification. π-Calculus is used for specification rules while verification is performed through the NuSMV model checker. (Che & Maag, 2014) refers to a novel testing framework for Extensible Messaging and Presence Protocol communication protocol extensively used in IoT devices for data exchange. Another theoretical framework for formal requirements modelling in pervasive computing systems is presented in (Ishikawa, Suleiman, Yamamoto & Honiden, 2009).

2.4 Conclusion

After an extensive SMS and the subsequent literature survey in this chapter, a lack of formal frameworks for security requirements, especially those from security standards specific to ICS, can be seen. The summary of limitations and challenges are listed before:

1. The SMS indicates that there is little research work done on the SRE of ICS. For instance, only ten studies in the last four years provide such solutions, whereas only two (Lahbib, Wakrime, Laouiti, Toumi & Martin, 2020; Hailesellasie & Hasan, 2018) are mainly related to ICS.

2. SRE approaches for ICS are mainly concerned with verification and validation, risk analysis and a general lack of traceability approaches can be seen. Few of such patterns can be seen in (Kavallieratos et al., 2020; Varela-Vaca et al., 2020; Hansch, Schneider & Brost, 2019; Kivelä et al., 2018; Zhou et al., 2020). Chapter 6 of this thesis solves this problem by proposing a security requirements repository connected to design-time tools to provide end-to-traceablity.

3. Similarly, current secure-by-design approaches for ICS are not entirely aligned with SRE and security standards. In addition, there is a general lack of focus on functional requirements and their integration with the design. Recent literature also suggests that modern IEC 61499 ICS development methods still require new design models (Lyu & Brennan, 2020). Secure links proposed in Chapter 5 aim to bridge this gap by providing a secure-by-design model for IEC 61499 distributed ICS applications.

4. The literature survey on the state of security standard implementation in ICS provides some promising approaches to help in security standard compliance in ICS. However, it can be observed that the majority of approaches focus on the verification of security requirements in the end products (Matheu et al., 2020; Ehrlich et al., 2020; Bicaku et al., 2021). None of the studies provides a comprehensive solution to verify the standard compliance during the development stages required in the security certification process. The only recent research that discusses the integration of security standards with ICS product development life-cycle stages is (Constante et al., 2021). However, it also lacks the internal details of security requirements, design and implementation stage. The lack of work on these aspects can also be attributed to the fact that the security standards for ICS are very recent, and their integration into the ICS development process is in very early stages. Such a gap can be filled by the integration of security

requirements repository, secure links, and TORUS as discussed in Chapter 7, Section 7.2.

Chapter 3

Security Goals: Confidentiality in ICS Applications

3.1 Prelude

The following chapter is published as a conference paper in ***2018 IEEE 16th International Conference on Industrial Informatics (INDIN)*** under the title of ***On Design-time Security in IEC 61499 Systems: Conceptualisation, Implementation, and Feasibility***.

The chapter introduces security links in IEC 61499 applications and a Confidentiality based Security Layer (CL4FB) for IEC 61499 applications and its implementation to support secure links. The work results from the first iteration of DSR methodology in which the researcher explored the methods to achieve security goals in the IEC 61499 ICS application. The first iteration regarding security goals is to investigate the feasibility of providing the design-time concept of confidentiality in ICS applications.

3.2 Abstract

Cyber-attacks on Industrial Automation and Control Systems (IACS) are rising in numbers and sophistication. Embedded controller devices such as Programmable Logic Controllers (PLCs), which are central to controlling physical processes, must be secured against attacks on confidentiality, integrity and availability. The focus of this paper is to add design-level support for security in IACS applications, especially around inter-PLC communications. The author proposes an end-to-end solution to develop IACS applications with inherent, and parametric support for security. Built using the IEC 61499 Function Blocks standard, this solution allows us to annotate certain communications as 'secure' during design time. When the application is compiled, these annotations are transformed into a security layer that implements encrypted communication between PLCs. This paper implements a part of this security layer focused on confidentiality, called Confidentiality Layer for Function Blocks (CL4FB), which provides a range of encryption/decryption and secure key exchange functionalities. We study the impact of using CL4FB in IACS applications with real-time constraints. Through a case study focusing on protection functions in smart-grids, we show that varying levels of confidentiality can be achieved while also meeting hard real-time deadlines.

3.3 Introduction

Industrial Automation and Control Systems (IACS) have significantly changed the way we operate manufacturing plants, transportation systems, facility management and monitoring systems (GICSP, Assante & Conway, 2014). Subsequently, the advent of the Internet of Things (IoT) has brought automation out of the industrial processes to the daily human lives. Reliance on such technologies has the potential for significant

improvement of human lives but with a strong caveat that their failure can sabotage and reset our technological journey. Of the many requirements relating to system quality, security is of chief importance in IACS. Legacy IACS were often confined within factory walls which is no longer the case. Now, they actively communicate with the outside world, leaving them more vulnerable to the security threats than ever before.

Although the top layers of most IACS are secured using network security mechanisms, devices at lower levels such as Programmable Logic Controllers (PLC), Remote Terminal Units (RTU), sensors and actuators are often left susceptible to attacks. Such small embedded devices control and process inputs/outputs from sensors and actuators. As they tend to have constrained resources and are designed to provide simple and reliable operations, cryptographic requirements for security purposes, like encryption and authentication mechanisms, are often overlooked which makes them highly vulnerable to attacks (Dunlap et al., 2016). Stuxnet (Langner, 2011), which specifically attacked PLCs in nuclear infrastructures, is a major example showing the extent of the damage possible if such devices are improperly secured. A vulnerability report on industrial control systems (Andreeva et al., 2016) shows that more than 90% of hosts connected to IACS have vulnerabilities ready to be exploited. The author advocates the use of security standards and the need for security certification when it comes to providing cryptographic services in IACS.

The IEC 61499 (Vyatkin, 2009) is a standard for designing and developing IACS applications. It is an event-driven architecture that uses Function Blocks (FB) connected to each other to form a chain of events. An FB provides an interface consisting of I/O events and data ports. There is a steady flow of industry practitioners adopting IEC 61499 to develop distributed applications. Research on security considerations in IEC 61499 based distributed applications is in the very early stage. There is also a scarcity of solutions that deal with full-scale security implementation regarding confidentiality, integrity and authentication in IEC 61499 distributed applications. The solitary solution

that we were able to find is the proposed authentication layer (Homay & de Sousa, 2016) which is more conceptual and is yet to be evaluated. The solution discusses the importance of secure key exchange but falls short of providing the enabling implementation. The contemporary research contains no solution providing confidentiality and integrity services to IEC 61499 based applications, therefore exposing communication between control devices make them susceptible to attackers. Also, existing solutions to secure communication like OPC Unified Architecture or Secure Socket Layer demand more processing power in small embedded devices which is not always available.

In this research, we propose a method where IEC 61499 distributed FB data links can be annotated with security requirements at the design-time. Then, pre-configured security mechanisms, forming a security layer, are added automatically at compile time to ensure secure communications between distributed slices of the application. In this paper, we also discuss the implementation of a security layer focused only on confidentiality. In the proposed layer, called *Confidentiality Layer for Function Blocks* (CL4FB), we implement encrypted data communications using Advanced Encryption Standard (AES) (Heron, 2009). We also implement the Diffie-Hellman Key Exchange (KE) algorithm for setting up and renewing secure communication sessions. These features are encoded using IEC 61499 compatible composite and service-interface function blocks. Furthermore, a study has been conducted to explore the feasibility of the proposed CL4FB, and its impact on system functions with strict timing constraints. Through a case study of protection functions in smart-grids, the study finds that while the CL4FB does introduce latency, most real-time constraints can be met while also ensuring some level of confidentiality. The key contributions of this paper are:

1. *Secure links*, a design-time mechanism that allows designers to annotate secure communications.

2. A *confidentiality based security layer* (CL4FB) and its implementation to support

secure links.

3. A *feasibility study* showing how the security-performance trade-off can be managed at design-time.

The rest of this paper is organised as follows: Section 3.4 presents background and a review of related literature. Section 3.5 discusses the notion of secure data links for an IEC 61499 application and subsequently proposes the CL4FB model and implementation. Section 3.6 implements our solution on an IACS application for protecting smart-grid power distribution networks. Section 3.7 presents the feasibility regarding latency that is incurred by adding CL4FB to the case study. Finally, Section 3.8 provides the conclusions and future directions.

3.4 Background and Literature Review

3.4.1 Industrial Automation and Control Systems

IACS are highly distributed and heterogeneous systems. They are an integral part of safety-critical cyber infrastructures around the world (Knorr, 2013). These systems contain technologically diverse sensors, actuators, controllers, communication channels and monitoring systems which leads to high complexity of design and development. Controller devices play the role of an arbiter between other components by collecting data from sensors, processing it and subsequently driving the actuators. They also interface with Supervisory Control And Data Acquisition (SCADA) Systems for monitoring purposes (Control, Process and Requirements, Security and Bond, Digital, 2006). Controller devices include PLCs and RTUs that are embedded devices often with limited processing power (Control, Process and Requirements, Security and Bond, Digital, 2006).

3.4.2 IEC 61499

Development of IACS is a difficult task because of complexity and enormity of the scale. IEC 61499 (Vyatkin & of America, 2007) standard provides mechanisms to develop applications for industrial distributed systems. It specifies distribution and management models for compliant applications. It is an event driven architecture that uses Function Blocks (FB) connected to each other to form a chain of events. An FB provides an interface consisting of I/O events and data ports. It relies on Execution Control Chart (ECC) to implement internal logic through associated algorithms. One major advantage of IEC 61499 is the possibility of distributing the functionality of a software module across multiple controller devices. Moreover, its helps in development of distributed application by inducing quality attributes like re-usability and reconfigurability while proving an abstraction against the underlying hardware.

The standard specifies different types of FBs. A Basic Function Block (BFB) is a simple FB. A Composite Function Block (CFB) is a network of FBs but it can not be distributed across the multiple resources or devices in contrast to Sub-application type that is also a form of CFB. Service Interface Function Block (SIFB) is like a device driver that provides services to 61499 applications hiding the under-lying complexities of hardware. Behaviour of SIFB is defined through sequence diagrams.

3.4.3 IACS and Security

Increasing emphasis on automating industrial processes means that the security of such infrastructures is of utmost importance. Several attacks (German nuclear plant, Dragonfly and Stuxnet) on IACS with devastating consequences have been reported in the last decade (Chapman, Ofner & Pauksztelo, 2016). Researchers and practitioners are facing a race against time to protect and secure critical infrastructure, as the automation technology for industrial processes is taking giant strides in its scale and reach.

The notion of security is contextual to a system's operating environment and its stakeholders. Security requirements vary with each system type, and there is no single definition of absolute security. Industrial systems are often difficult to manage and prioritise which may lead to missing requirements. Security standards cater sets of best practices that should be followed to achieve the maximum possible security. Following a standard ensures that the system implements a rigorous, well researched and published set of security requirements. They provide the basis for system-wide assurance, risk assessment and mitigation through policies and frameworks.

A 2016 Industrial Control Systems (ICS) vulnerability report by Kaspersky Labs (Andreeva et al., 2016) shows a grim picture regarding ICS security. Their main findings show that the number of exploitable vulnerabilities in ICS components are growing with time. Most of the vulnerabilities (49%) found are of a high-risk level, and 5% of the vulnerabilities found in 2015 were yet to be patched at the time of publishing of this report. ICS components that are available through the internet are using insecure protocols such as Http, Telnet, SNMP, etc. Most of these components can be accessed externally and combining this phenomenon with the use of insecure protocols, has resulted in 92% of hosts deemed to be insecure and vulnerable. The report also finds out that a wide range of industries is affected by the lack of security measures taken against exploitable vulnerabilities.

Citing (Andreeva et al., 2016), (Wagner, 2016) raises concerns about the state of hardware security and the need for appropriate measures to be taken by vendors. These works highlight the role of cryptography in securing hardware devices, but the ever-changing landscape of ubiquitous computing demands a more extensive approach. It includes robust architectures and frameworks with security requirements integrated "by-design". Lack of resources (e.g. memory and processing power) in such devices also means that they have to bank on lightweight security solutions which otherwise, may lead to attacks such as denial of service where an attacker overwhelms its target.

Therefore, robust measures should be taken when considering securing such controller devices. A system Protection Profile (PP) developed by the Process Control Security Requirements Forum (PCSRF) (Control, Process and Requirements, Security and Bond, Digital, 2006) specifies the minimum security requirements for field devices such as PLCs and RTUs for SCADA systems in medium robustness environments. The PP also requires that if the Target Of Evaluation (TOE) (Field device) implements cryptographic functionality, then it must comply with FIPS 140-2 (NIST, 2016) requirements at certain levels. One of the cryptographic requirement is the provision of encryption services that are also being advocated in the British Standard for Industrial Communication Networks — Network and system security (IEC-62443-3-3:2013) (62443-1-1, 2009).

An attempt to secure PLCs at the component level is made in (Alves, Morris & Yoo, 2017). The paper discusses the implementation of AES256 using the OpenPLC runtime platform. It implements AES256 on the network layer in OpenPLC so that all data leaving the PLC is encrypted. For decryption, a separate application that runs on a SCADA device receives and decrypts packets. It also reports encryption-related latency induced but is limited to communications between PLC and SCADA devices and lacks more fine-grained support required for distributed architectures like IEC 61499.

Moreover, security solutions in most IT environments prioritise security goals in order of Confidentiality, Integrity and Availability (CIA), but in IACS environments, the system's priorities are the other way around. That is, IACS or SCADA systems put importance on availability because of the real-time nature of the operation; therefore, such kind of systems should consider these security goals in the order of AIC (Knowles et al., 2015).

3.4.4 Security in IEC 61499 based Applications

The IEC 61499 standard provides mechanisms to develop applications for industrial distributed systems. It specifies a distribution model for deploying an application on multiple devices and a management model for managing resources within a device for compliant applications. It is an event-driven architecture that uses interconnected function blocks (FBs) to form chains of events. FB have interfaces containing I/O events and data ports (Vyatkin, 2009). It relies on an Execution Control Chart (ECC) to implement internal logic through associated algorithms. IEC 61499 allows distributing the functionality of a software module across multiple controller devices. Moreover, it helps in the development of the distributed application by focussing on re-usability and reconfigurability through design-time abstractions of underlying hardware.

The distribution model of IEC 61499 specifies that a sub-application type containing multiple FBs can be deployed on one or more devices. The distribution is seamless as far as the architecture is concerned, but the distributed FBs communicate using a communication network. The standard specifies interfaces through communication FBs (that are a type of SIFBs), but the underlying properties of a communication channel are not in the scope of the standard. Securing such channels is very desirable as IACS are susceptible to a variety of cyber attacks (Dunlap et al., 2016). In a distributed IEC 61499 application, FBs communicating over a communication network regularly need to send I/O parameters. An attacker can disrupt the services by taking advantage of communication layer vulnerabilities.

OPC UA (Mahnke et al., 2009) is a resource-intensive communication protocol designed for IACS. Thus it is not always practised to deploy it for the security of small embedded devices. It also provides a security model for communication between the components of IACS. OPC UA is a resource-intensive protocol that works on a client-server model. It is not always possible to deploy such a protocol in a micro-environment

where controller components of ICS communicate with each other. In the absence of a secure channel, such devices must rely on end-to-end security mechanisms to be able to get protected against attackers.

Security of IACS and its related domain such as Cyber-physical Systems (CPS) or Internet Of Things (IoT) has been the focus of the academia and industry (Wagner, 2016; T. Lu et al., 2014; Krotofil & Gollmann, 2013). In contrast, research on security considerations in IEC 61499 based distributed applications is very early. It can be because IEC 61499 only mandates providing semantics for the design and development of distributed applications in IACS.

A proposal to achieve integrity and authenticity in the context of IEC 61499 using the Universal Message Authentication Code lightweight algorithm for message authentication through an Authentication Layer has been described in (Homay & de Sousa, 2016). It utilises the publisher/subscriber model of IEC 61499 in multi-cast mode, i.e. a publisher can communicate with multiple subscribers. It assumes that the keys have already been transmitted securely before the start of the session. It proposes the use of 4diac FORTE (*Eclipse 4diac™ - 4diac IDE and 4diac FORTE runtime*, 2021) IEC 61499 runtime for the implementation purposes. A similar approach to protect message integrity between PLCs hosting IEC-61131-3 (Ramanathan, 2014) — a predecessor of IEC 61499 — based applications is presented in (Homay, Martins, de Sousa & Kashefi, 2016).

Although message integrity and authenticity are priorities in IACS environment, lack of confidentiality in distributed controller communication can also disrupt the services provided by these devices. The confidentiality goal is achieved by encrypting the data. Encryption ensures that an unauthorised party does not get to read the private information. In the absence of the encryption, an attacker can eavesdrop on the data to alter it in ways that can lead to activities such as unauthorised monitoring and traffic analysis. It is also true in case of IEC 61499 based applications. For

example, unencrypted data exchange between two FBs deployed on multiple devices in a sensitive environment can cause a breach of private data leading to the revelation of critical parameters to the attacker. Although, an attempt to introduce confidentiality in PLCs has been made in (Alves et al., 2017) but it lacks the support for distributed applications. In this research, we will largely focus on providing cryptographic services to a distributed IEC 61499 based application that will help in achieving the goal of confidentiality.

3.5 Secure SIFBs for IEC 61499 Applications

The IEC 61499 standard specifies different types of FBs. All FBs have unique interfaces consisting of I/O events and data ports. The execution of a Basic FB (BFB) relies on an ECC which is a finite state machine that implements internal control logic using associated algorithms. Composite Function Blocks (CFB) contain networks of FBs to achieve more complex functionalities. A Service Interface Function Block (SIFB) is like a device driver that provides services to IEC 61499 applications, such as network communication while hiding the underlying complexities.

Figure 3.1 shows how parts of an IEC 61499 application can be deployed over separate PLCs, each hosted within a supported runtime. The communication SIFBs may be inserted where needed to ensure event and data flow between these runtimes. However, data transmitted by such SIFBs over a network is susceptible to attacks unless the transmission medium is secure. Heavy-weight solutions such as IPSEC are often not feasible due to the limited processing power of the PLCs (Potlapally, Ravi, Raghunathan, Lee & Jha, 2007). Also, leaving the selection of the communication medium until deployment time is not ideal, especially as the configuration of the system over PLCs can change. Currently, secure communications are treated no differently to any other communication during design time.

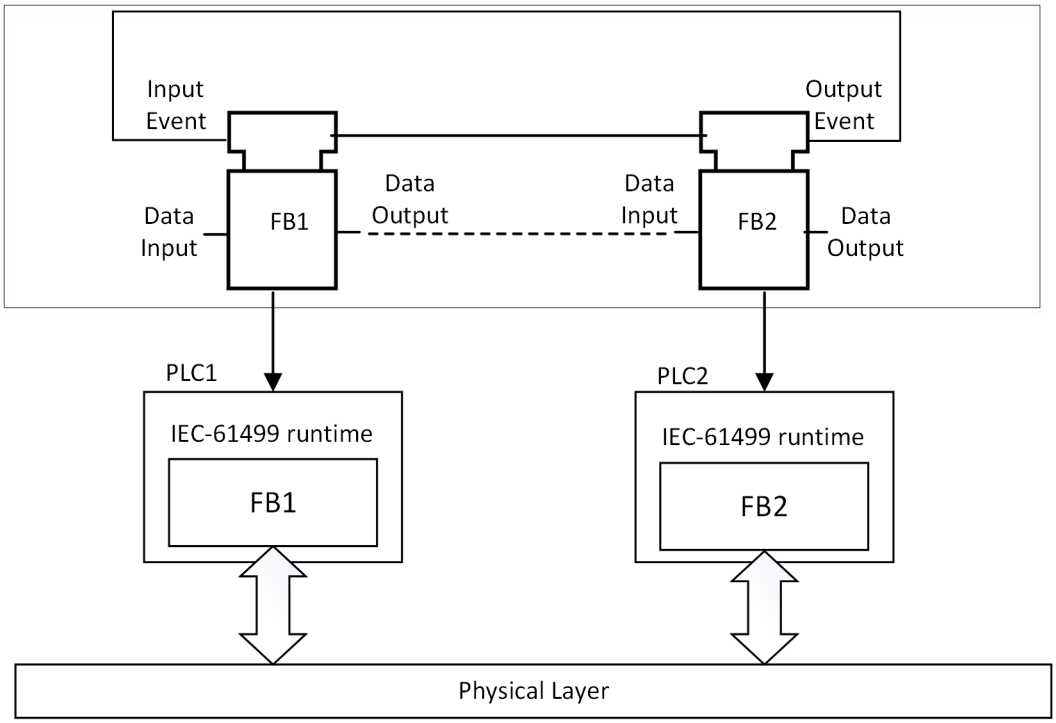

Figure 3.1: IEC 61499 application FB distribution concept

3.5.1 Secure Links for Distributed IEC 61499 based Applications

We propose a technique that allows designers to identify secure data links within an IEC 61499 application during the design time. At the compile time, such links are automatically translated into pre-configured application logic and supporting SIFBs, to make the links secure. Formally, a secure link sl is defined as:

$$sl = \langle d\_con, sec\_goal, alg, params \rangle$$

where $d\_con \in D\_Conns$ is any link between the data ports of an FB application (Vyatkin, 2009), $sec\_goal \in \{Confidentiality, Integrity, Availability\}$ identifies the type of security required, $alg \in Algs(sec\_goal)$ is an available algorithm for the type of security required, and $params$ is a set of parameters required for configuring algorithm alg. For instance, a data link requiring confidentiality can be annotated with

a specific confidentiality algorithm and related parameters such as key size, encryption mode etc. Similarly, if integrity is a required goal for a link, it can be annotated by specifying algorithms such as HMAC and corresponding parameters. Secure data links essentially translate to annotations on specific data-links in a FB application. These annotations, namely sec_goal, alg, and $params$ are used by the compiler to automatically include and configure the required security code from an existing FB library. This technique provides improved usability and abstraction by preventing the need to include and configure security blocks explicitly at design time. Figure 3.2 illustrates an example of secure IEC 61499 FB data link. `@secure` is the keyword to mark the secure data link. The first parameter `C` stands for Confidentiality that is a security goal. The second parameter `AES` is the name of the encryption algorithm. The last parameter may accommodate varying numbers of arguments required for the algorithm used in the second parameter. The event connection/link (`CNF` to `REQ`) does not carry the data but only acts as a trigger. Therefore, encrypting event links is not applicable especially in this case.

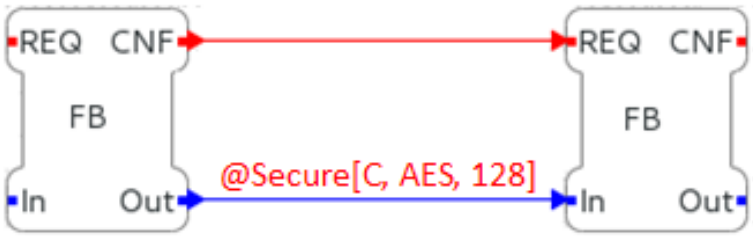

Figure 3.2: A secure FB data link with annotation

For the compiler to generate secure links, a set of enabling FBs and SIFBs must be available, in the form of a library. In the following sections, we demonstrate an FB library for proving security, focussed on confidentiality. The library can be extended in the future to provide other security elements such as integrity, availability, authenticity etc.

3.5.2 Confidentiality Layer for Function Blocks (CL4FB)

All secure data links in an application are compiled into a logic that supports secure communications. The logic, implemented using IEC-61499 FBs, ultimately forms an independent layer whose chief aim is to secure data communications between distributed parts of the application. In this paper, we propose a confidentiality-based security layer built on the IEC 61499 distribution model. The library, called Confidentiality Layer for Function Blocks (CL4FB), provides the services related to confidentiality. It can be envisioned as a secure tunnel through which data from one FB is transmitted to another FB over a non-secure medium.

An alternative way to ensure security is to secure the medium over which PLCs communicate. It can be complex, especially in the case of confined but critical industrial networks. It may include deploying security protocols, e.g., a Transport Layer Security (TLS) or IP Security (IPsec) if Ethernet-based communication is required, the set up of which may not be feasible in the PLC network topology. Constraints like power consumption and timing can limit the usage of this alternative approach. By incorporating security within IEC 61499 distributed applications, we can avoid the overhead of using a secure medium, and also make the application more self-contained and portable.

CL4FB consists of encryption and decryption logic implemented using FBs. Figure 3.3 shows the abstract configuration of FBs in a distributed application forming CL4FB. The choice of implementation is the prerogative of the designer and the security level they desire.

In this research, we assume that a block cypher is more appropriate in the current context as the amount of data to be processed is known in most cases. In the case of stream cyphers, the amount of data to be encrypted is usually unknown. When choosing a block cypher, the primary criteria to consider is the trade-off between security and performance. We choose to adopt the well-known and widely accepted AES algorithm

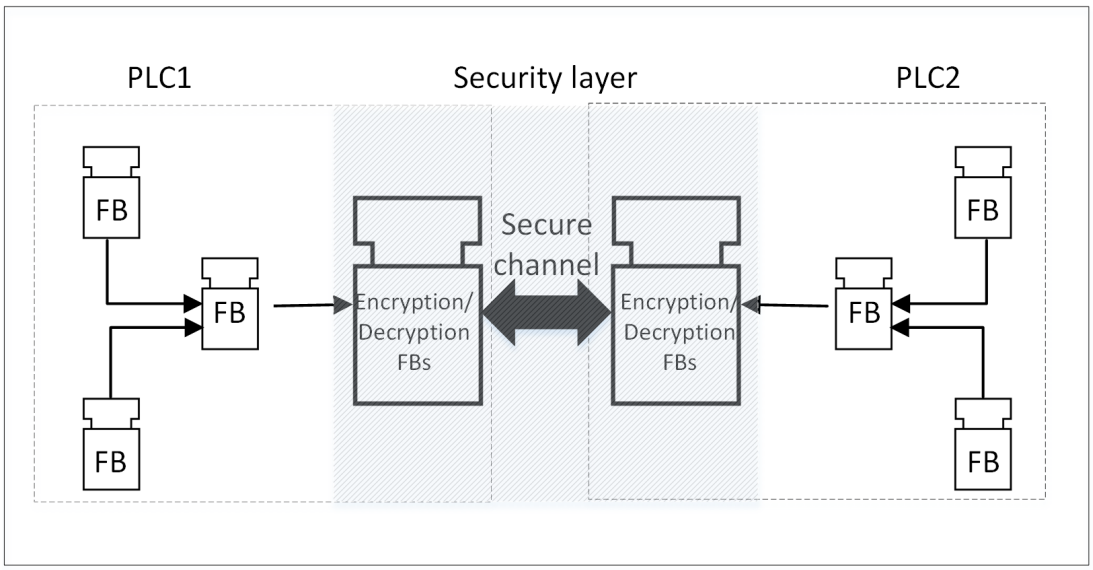

Figure 3.3: Proposed Security layer in an IEC 61499 distributed application

for secure data communications.

Key expansion is an integral part of block cyphers. It is the foremost step performed to expand a smaller key into a larger key that is used in a subsequent Feistel network (Nyberg, 1996) that is common to block cyphers. It is a one time-operation that is performed before an encryption session until the key is replaced. In the proposed CL4FB, the key expansion process can be realised as a separate FB because incorporating key expansion in each cycle of encryption induces unwanted latency in the encryption/decryption process. A designer must make sure that the key expansion is performed before the actual encryption process starts on all controller devices where the FBs of an IEC 61499 application are distributed.

3.5.3 (SI)FBs for the CL4FB

The most common method to transfer data between distributed FBs is through publisher/subscriber SIFBs (Vyatkin, 2009). Two data transmitting FBs need a pair of publisher-subscriber SIFBs to send data over the network. A publisher is used for

an FB providing encryption services to send cypher text over the network. Similarly, decryption FB uses a subscriber SIFB. Publisher/Subscriber pair may be conceived as an implicit part of the CL4FB providing confidentiality services. A more detailed view of CL4FB can be seen in Figure 3.4. In the case when a distributed application needs to send secure data in a bidirectional manner, another set of publisher/subscriber SIFBs along with encryption/decryption FBs can be used in a flipped configuration. An alternate way of aiding a simpler design for an IEC 61499 distributed application is to implement both encryption (publish/send) and decryption (subscribe/receive) algorithms in a single FB. It can be efficiently realised by adding a predicate variable to the FB interface and subsequently incorporating relevant logic in the ECC of the FB.

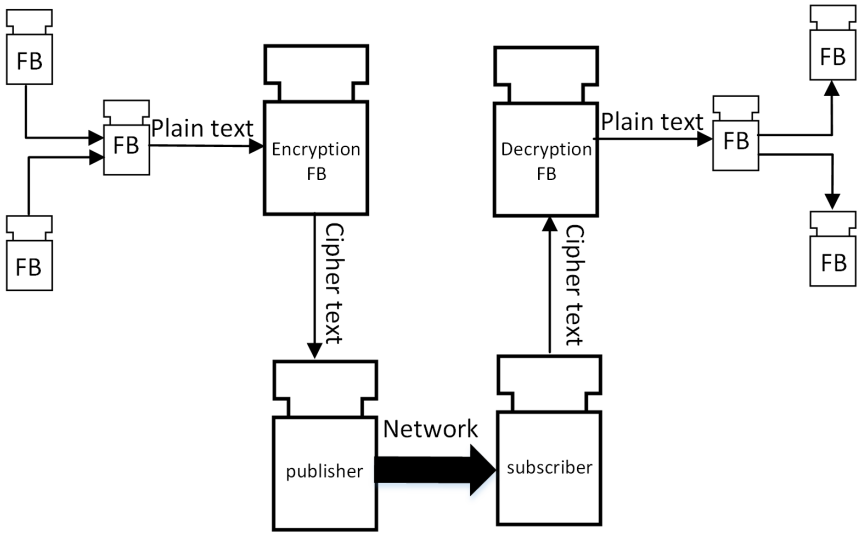

Figure 3.4: An internal view of IEC 61499 distributed application with CL4FB

Figure 3.5 shows the event and data I/O interfaces of the FBs proposed for encryption services and designed for the I/O requirements of the block cyphers. IEC 61131-3 data types are used to define I/O data. The service is started when the `REQ` event is triggered, and triggering of a `CNF` event indicates that the services have been completed. In the encryption FB, *pt* is a 16-byte array which makes a block of plain text. *ksize* denotes the size of key to be used in a block cypher. The most common lengths are 128,

192 and 256 bits. It is an important variable because the length of the key determines the number of rounds to be executed in a Feistel network block cypher. *expkey* is the expanded key that is the outcome of the key expansion process. Its size depends upon *ksize*. Finally, *ct* is the cypher text which is the output of the encryption process. All the data I/O parameters described here remain the same for decryption FB except that it takes cypher text (*ct*) as an input and decrypts it to produce the plain text (*pt*). The key expansion FB simply takes key and key size as input parameters and outputs an expanded key which is then fed into the encryption and decryption FBs. Alternatively, *KeyExp* FB can be disposed of by making key expansion an implicit step during the initialisation of encryption and decryption FBs. It can be achieved by adding an IEC 61499-provided INT event and necessary logic in the ECCs. We believe that separating the key expansion process from the main encryption/decryption process leads to a modular approach inducing clarity and flexibility that is the hallmark of the IEC 61499 standard.

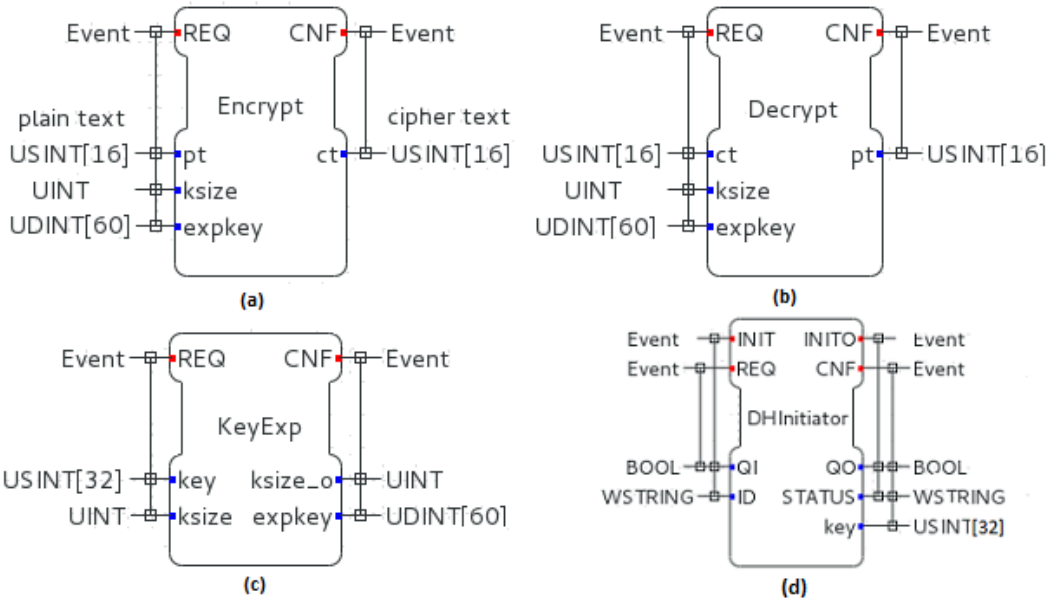

Figure 3.5: CL4FB event and data interfaces: (a) FB for encryption services (b) FB for decryption services (c) FB for Key expansion process (d) Proposed Secure KE SIFB

3.5.4 Secure Key Exchange

The security of an encryption algorithm depends upon its keys. An attacker can get hold of the key if the medium used to transfer the key is not secure. There has been a significant amount of research regarding methods of secure Key Exchange (KE) and distribution (Canetti & Krawczyk, 2001). Participating controller devices must have the same key for the encryption and decryption process if using symmetric key block cyphers to secure the communication between distributed FBs. Therefore, we also propose a supplementary secure KE process implemented in the form of an SIFB that may be slotted in as a precursor to the actual encryption process.

Cryptographic KE protocols such as Diffie-Hellman KE (Diffie & Hellman, 1976) require a Random Number Generator (RNG) on each participant device, implemented as a Pseudo-Random Number Generator or a True Random Number Generator. In either case, an FB implementing KE protocols requires accessing the entropy pool for RNG. Moreover, Diffie—Hellman KE also requires the transfer of public parameters over the network. Therefore, an SIFB is used to implement the KE protocol as it can provide underlying system services for RNG and network operations.

Figure 3.5 (d) presents proposed SIFB for secure KE in an IEC 61499 distributed application. The SIFB can act both as an initiator and a responder.

A unified SIFB for KE reduces the design complexities induced by the use of publisher/subscriber pair. Communication through publisher and subscriber SIFBs is unidirectional which dictates that, for a well-defined bidirectional communication, two publisher/subscriber pair are needed. It is an undesirable design decision for an IEC 61499 distributed application to guarantee a unique publisher/subscriber pair for each thread of communication in each direction. It can overwhelm the topology of the FBs with too many communication FBs. Even if input multiplexing/de-multiplexing is used with the publisher/subscribe respectively, the design becomes unnecessarily

complicated.

Events in KE SIFB are standard `INIT`, `INITO`, `REQ` and `CNF`. `INIT` can be used to initialise KE protocol's context structures and generate random numbers from the entropy pool. Data input *QI* is used to determine the role of the particular SIFB, i.e. initiator or responder. Input *ID* is a standard publisher/subscriber parameter that is used to assign a unique pair of IP and port number. *STATUS* shows the current status of the service at each phase of KE process. A public value computed by each SIFB is transferred to the participant peer SIFB using system's underlying network services. The public value is used to compute the symmetric shared secret key in for Diffie—Hellman KE. Finally, *Key* output parameter is the generated symmetric key that can subsequently be used by encryption/decryption FBs. A comprehensive data flow model involving (SI)FBs of CL4FB along with secure KE can be seen in Figure 3.6.

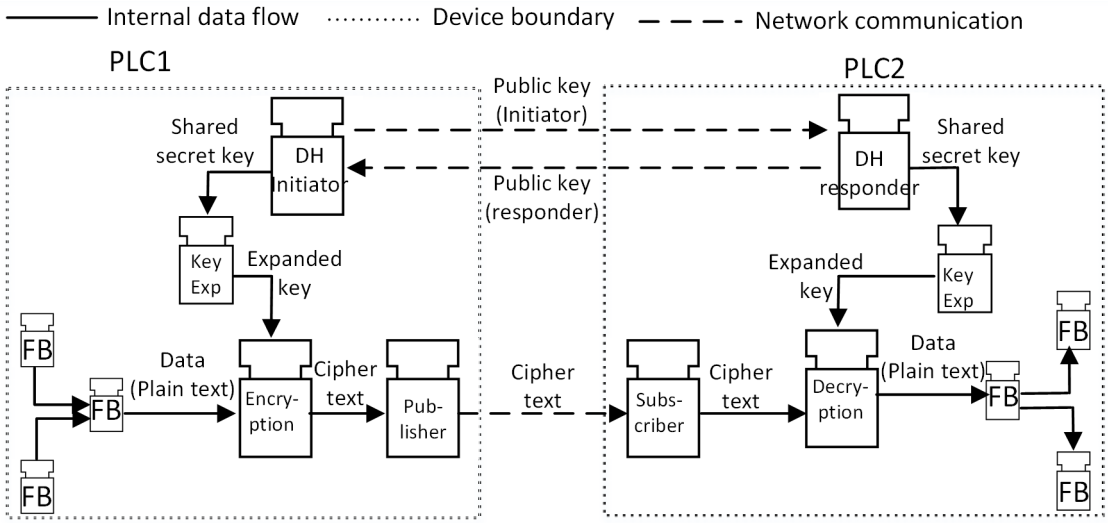

Figure 3.6: An all-inclusive model of CL4FB with key exchange

The proposed security (SI)FBs maintain secure sessions along with security policies in an IEC 61499 environment. The encryption/decryption FBs and the KE SIFB offers flexible interfaces to implement a variety of block cyphers and KE protocols. Sessions may be maintained by putting a time constraint on the usage of a particular key after which, re-keying and secure KE must be performed, and a new key is distributed

amongst participating devices. Moreover, multiple block cyphers may be implemented in the proposed encryption/decryption FB encapsulations that opens up the possibility of dynamic set of options to be used in each session. However, such mechanism is out of scope for this paper.

3.6 Case Study: Secure Smart-Grid Protection

In this section, we demonstrate the applicability and viability of CL4FB using an IEC 61499 based solution for protection and control functions in electric power distribution (Zhabelova, Yang, Vyatkin, Etherden & Christoffersson, 2016). Here, IEC 61499 is used for three protection functions, earth fault, over-current protection and differential protection. A similar concept of Intelligent Fault Management is also discussed in (Zhabelova, Patil, Yang & Vyatkin, 2013) where trip signals from protection functions are sent to circuit breakers using User Datagram Protocol connection.

The over-current protection function safeguards the system against issues like short circuits or overloaded lines. It trips the circuit breaker when the currents surmount a threshold value of 100A. It must trip the circuit within 600ms as specified in the referenced research. Similarly, a differential protection function protects the power transformer from internal malfunction when the difference between two currents in the transformer exceeds 1A, triggered within 5ms of the fault occurrence. We use these timing requirements as benchmarks to show the feasibility of CL4FB in this paper.

Electric power distribution requires high security due to its widespread utilisation and pivotal role in critical infrastructures. The possibility and nature of cyber and physical attacks on electric power delivery system have been discussed in (Council et al., 2012).

Figure 3.7 shows IEC 61499 implementation scenario of three protection functions described earlier, designed to be implemented in designated Intelligent Electronic

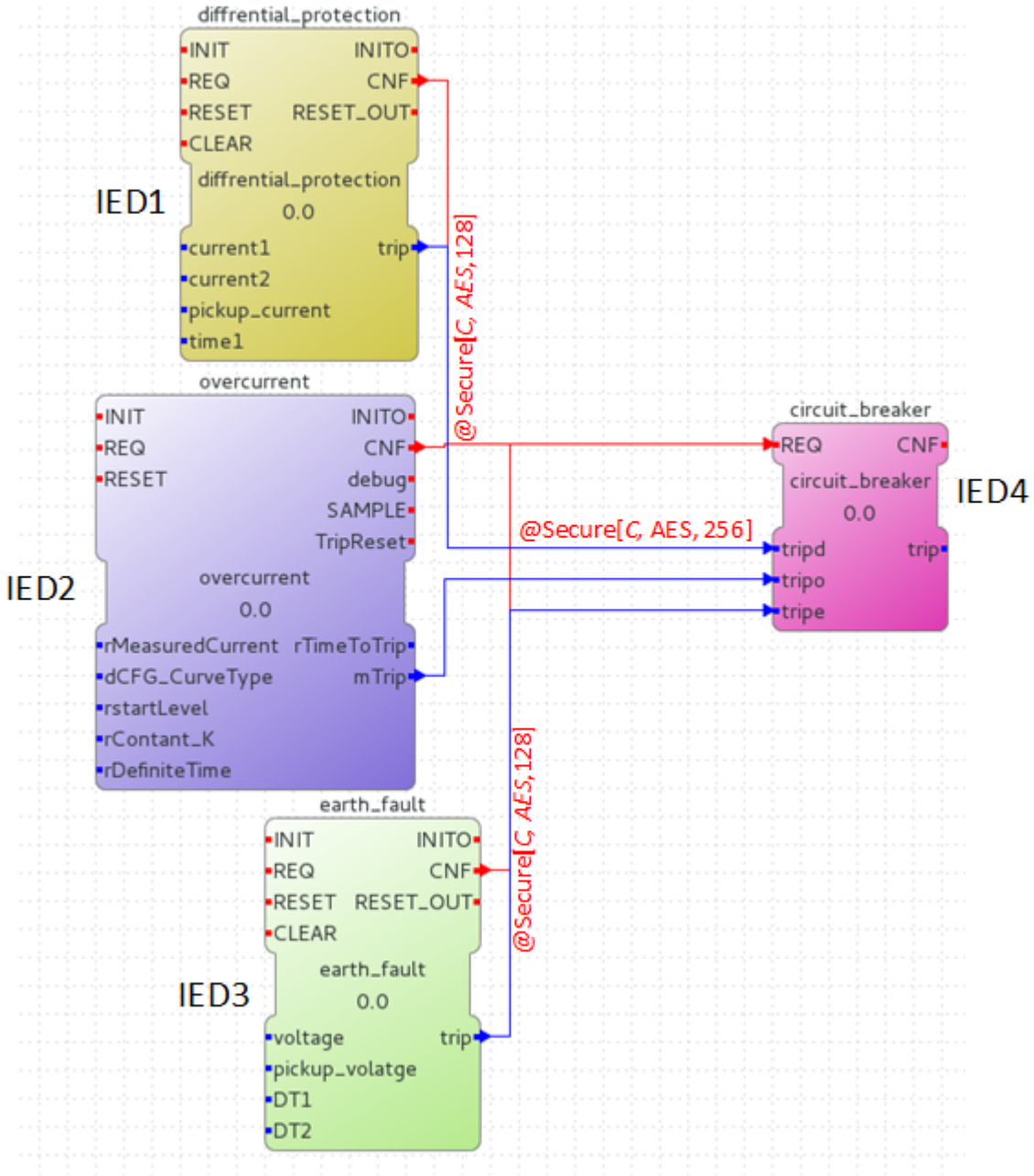

Figure 3.7: An IEC 61499 system for implementing protection functions

Devices (IEDs). The IED sends a trip signal to a circuit breaker when an anomaly occurs in current or voltage levels. Circuit break is perceived to be a separate IED as well. All the devices on the left side of the figure communicate to circuit breaker using the Ethernet. Each left-to-right link is annotated with security parameters assessed to fulfil the timing requirements. That is, AES128 with fewer rounds is appropriate for

strict timing requirement of 5ms for differential protection function while AES192 or AES256 is well-suited for the slightly relaxed timing requirement of overcurrent protection. As far as the annotations on the links are concerned, these are envisioned to be part of an automated security framework for IEC 61499 based IDE and run-times where annotations may help compilers to generate and configure FBs according to the provided security parameters. In this research, we have executed the same process manually.

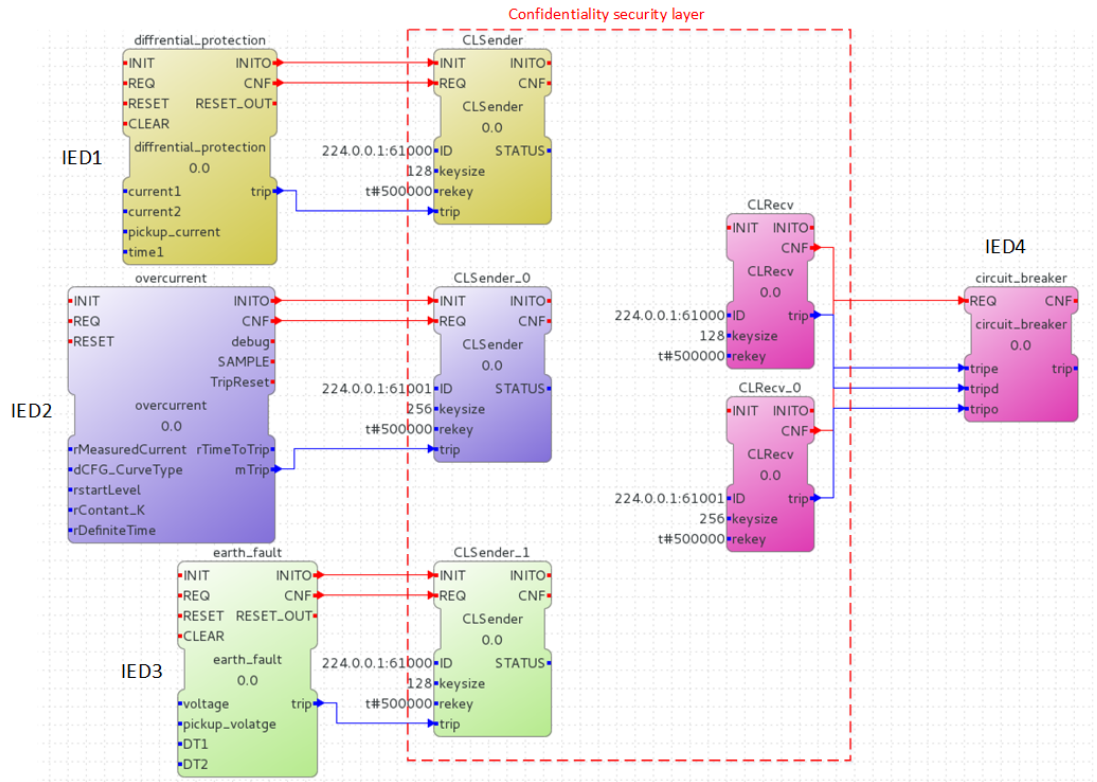

Figure 3.8: Implementation of CL4FB for protection functions

Figure 3.8 demonstrates a CL4FB implementation to secure the communication between protection FBs and the circuit breaker. The dotted outline represents the domain of the layer. It consists of two CFBs named `CLSender` and `CLRecv` responsible for sending and receiving the encrypted trip signal respectively. FBs deployed in an IED are shown in corresponding colours. In this case, three `CFSender` FBs—deployed on

designated IEDs—are required to carry the encrypted trip signal over the network from respective protection function and deliver it to the circuit breaker. *ID* in `CLSender` and `CLRecv` input interfaces is the combination of multicast IP host group and the port. Due to the timing requirements of protection functions described earlier, `CLSender` and `CLSender_1` encrypt the trip signal with a 128 bit key (via *keysize* interface), so they share the same IP:Port combination. The `CLRecv` CFB receives the trip signal and decrypts it before sending it to the circuit breaker. Similarly, `CLSender_0` and `CLRecv_0` communicate at different IP:Port combination using a 256-bit key but they can also use 128 or 196-bit keys due to the slightly relaxed requirements of overcurrent protection. The *rekey* parameter specifies the time interval after which a new key must be negotiated between communicating IEDs.

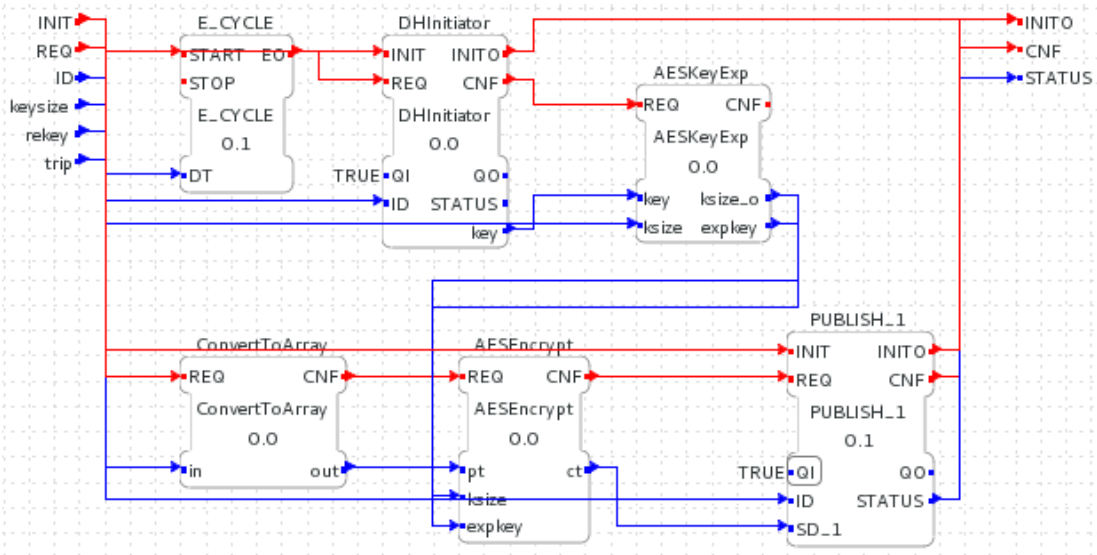

Figure 3.9: CLSender CFB internal FB network

Figure 3.9 represents internal FB network of `CLSender` CFB. `DHInitiator` SIFB and `AESKeyExp` FB are executed once per session; the length of which is determined by *rekey* input parameter associated with `E_CYCLE` FB. Expanded key from `AESKeyExp` is fed into `AESEncrypt` while `ConvertToArray` is another FB that converts the boolean value of the trip signal into an input plaintext block of

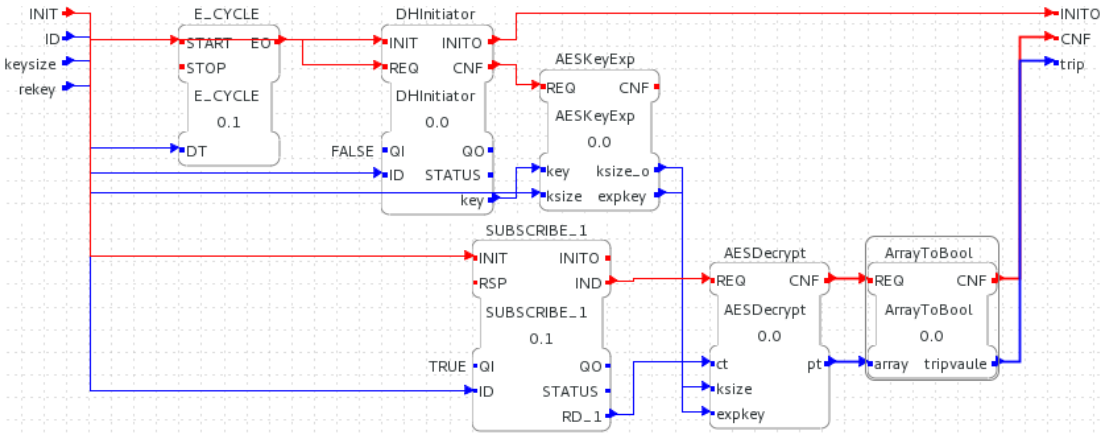

Figure 3.10: CLRecv CFB internal FB network

16 bytes required by AES. Resulting cyphertext is sent to the circuit breaker over the network using a publisher SIFB. Similarly, Figure 3.10 serves an internal FB network of `CLRecv` CFB that is deployed in circuit breaker IED. Cyphertext sent over the network is received by the subscriber SIFB and forwarded to `AESDecrypt` FB for the decryption. The resulting plaintext block is converted back to a boolean trip value.

Establishment of a secure channel in this case study is only the demonstration of the capability of the proposed CL4FB. Having said that, using larger aliases for boolean trip values may enhance the security. The next section tests the feasibility of our approach by deploying and executing the FBs in an IEC 61499 runtime environment.

3.7 Feasibility Analysis

The trade-off between a system's performance, usability and functionality at the expense of security has been a topic of interest in the research community (Elahi & Yu, 2007). The trade-off between the performance and security is well known, due to their competing needs for the processing power. In this section, we evaluate how the proposed security library affects the performance, characterised by latency, for the protection system as discussed in section 3.6.

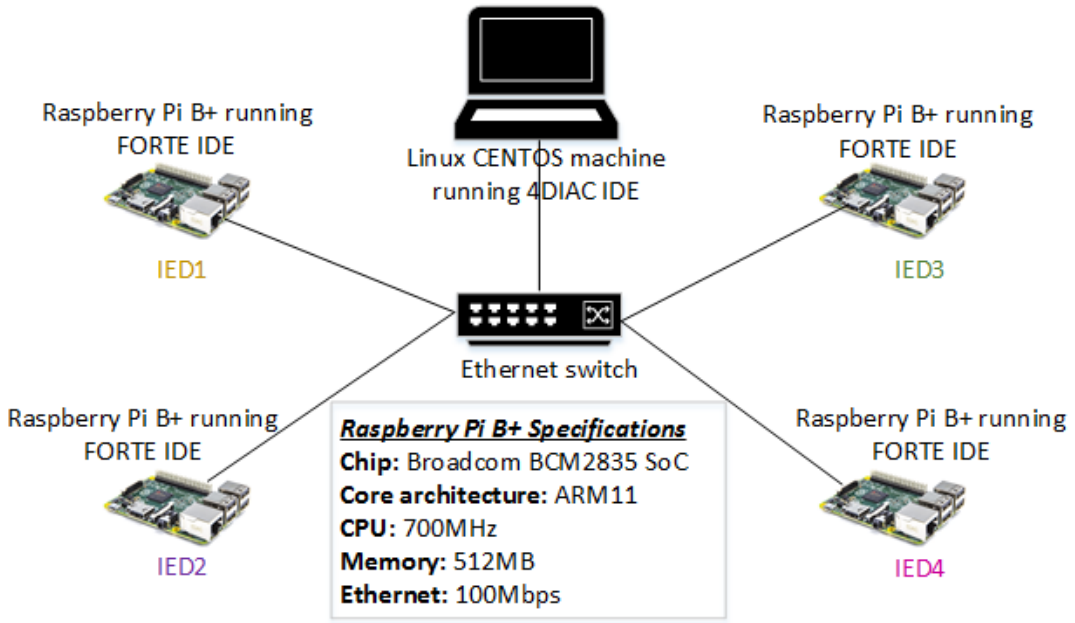

Figure 3.11: Experimental setup

To realise the CL4FB (SI)FBs, we have used the 4diac Integrated Development Environment (IDE) (version 1.8.4) that provides a platform to develop distributed applications according to IEC 61499 specifications. It also provides 4diac FORTE which is an open-source portable Runtime Environment (RTE) for executing IEC 61499 applications. 4diac FORTE (version 1.9.0.M1) was used to deploy and execute FBs on Raspberry Pi B+ boards to simulate IEDs. Figure 3.11 illustrates the experimental setup. A Linux machine (CENTOS 7.2) runs the 4diac IDE that is used to design and deploy IEC 61499 distributed applications. The deployment process consists of sending FB constructs including the information related to distributed operations, to 4diac FORTE RTE instances each running on a separate Raspberry Pi. The distributed FBs deployed on the 4diac FORTE RTE communicate with each other via Ethernet. A Network Time Protocol (NTP) server was established in a LAN environment to ensure the synchronisation of the systems clocks on each Raspberry Pi. A less than 1 ms NTP offset was achieved due to little congestion in the LAN implemented only for the NTP purposes. We deem NTP server and client configurations to be out of the scope this

paper. For latency measurement, an SIFB `TimeStampRecorder` recording UNIX timestamps in milliseconds was developed. A time stamp t_1 was obtained by placing `TimeStampRecorder` before the `AESEncrypt` FB in each of the `CLSender` CFB. Similarly, time stamp t_2 was obtained by putting `TimeStampRecorder` right after the `AESDecrypt` FB in each of the `CLRecv`. The time stamp t_1 was sent over the network to the Raspberry Pis having `CLRecv` CFBs using a separate publisher/subscriber pair. The latency (L) is then calculated by $L = t_2 - t_1$. Using this setup, we measured the latency of CL4FB to analyse their usability in IACS systems. AES block cipher is implemented in encryption/decryption FBs using Electronic Codebook Mode (ECB). In the proposed SIFB for secure KE, Diffie—Hellman KE method is implemented along with necessary network communication using multicast mode. A stub application was created to generate the plain text which is then fed into the encryption FB.

Table 3.1: Latency of FBs for the proposed CL4FB

Latency without encryption process	1-2 ms		
Configuration	**AES128**	**AES192**	**AES256**
Single controller device	2-3 ms	3 ms	3-5 ms
FBs distributed on multiple IEDs	5-6 ms	8-9 ms	10 ms

Calculating delays is an important factor to consider when design distributed applications where response time to an event is critical (W. Wang & Lu, 2013). Table 3.1 shows the latency induced by the CL4FB. The results are obtained by noting the minimum-maximum latency values over 100 cycles for each scenario. The first row of the table shows the processing time taken by encryption and decryption FBs, i.e. time taken from when encryption FB's `REQ` event is triggered to the triggering of a `CNF` event of decryption FB that outputs the decrypted plain text. Both FBs are deployed on the same machine in this instance. However, the second set of results present the latency of the encryption process when FBs are distributed on multiple devices. Resulting

values are slightly higher due to network communication. The processing time for key expansion is not included as it is a one-time process that is not required for the encryption of each block.

The results in Table 3.1 indicate that AES with 128, 192 and 256-bit key may well be suitable for overcurrent protection function because of its lenient timing requirements. On the other hand, only AES128 may satisfy the requirement of differential and earth fault protection functions in the current experimental setup. It can be seen that a 5 ms threshold value is barely achieved even by AES128 that may cause problems and damage the equipment by not tripping the circuit in the required amount of time. However, execution platforms more powerful than the deployed Raspberry Pi B+ may produce better results to satisfy the timing requirements of differential and earth fault protection functions. Therefore, a designer can consider these results as a clue to assess the feasibility of introducing CL4FB into IEC 61499 distributed applications. Subsequently, it will also help in the selection of appropriate cryptographic transform to minimise the trade-off between security and latency. For example, a light-weight encryption algorithm can greatly help to reduce the latency when security is a desired feature. Also, we believe that selecting an IEC 61499 dedicated IED can eliminate the overhead caused by the general-purpose operating system in a Raspberry Pi. The timing constraints noted in (X. Lu, Wang & Ma, 2013) regarding smart-grid based Future Renewable Electric Energy Delivery and Management (FREEDM) systems project, lie within the range of results.

Similar to the key expansion operation, secure KE is involved whenever the key needs to be updated, after a specific interval of time depending on the security policy. Latency induced by secure KE depends upon external factors such as generation of random numbers and transferring of public value over the network. It results in a variable delay for KE which is why latency values involving KE are not included.

3.8 Conclusions

Securing embedded controller devices such as PLCs or RTUs against attacks by adding security mechanisms to these resource and memory constrained devices can mean sacrificing performance. Due to this trade-off, often security-related decisions are delayed from early design and development phases to the later deployment phase. This paper proposes an approach to make security related decisions. At the heart of this approach is a confidentiality-based security layer for function blocks called CL4FB, implemented using IEC 61499 FBs. This layer acts as a library for secure communications and supports a variety of security algorithms varying in performance overheads and levels of security provided. Through a case study of a smart-grid protection system, we show that even for communications constrained by strict hard real-time deadlines, some level of security can be incorporated.

Future directions for this research include the implementation of additional confidentiality, integrity and availability related mechanisms and algorithms to the library and their benchmarking. Also, there is a need to incorporate mechanisms for authentication and integrity, the two equally important pillars of security in industrial automation systems. Finally, tool-support for automatically instantiating security blocks against secure FB links required for any IEC 61499 application is another promising research avenue that we are exploring.

Chapter 4

Security Goals: Availability in ICS Applications

4.1 Prelude

The following chapter is published as a conference paper in ***2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*** under the title of ***Designing Actively Secure, Highly Available Industrial Automation Applications***.

The work is conducted in the second iteration of DSR to advance the knowledge by exploring the feasibility of an IEC 61499-based intrusion detection and prevention system to achieve the critical security goal of availability. The result of this work helps to extend the S-Lib security function block library first presented in Chapter 5 and further discussed in Section 7.5.

4.2 Abstract

Programmable Logic Controllers (PLCs) execute critical control software that drives Industrial Automation and Control Systems (IACS). PLCs can become easy targets

for cyber-adversaries as they are resource-constrained and are usually built using legacy, less-capable security measures. Security attacks can significantly affect system availability, which is an essential requirement for IACS. We propose a method to make PLC applications more security-aware. Based on the well-known IEC 61499 function blocks standard for developing IACS software, our method allows designers to annotate critical parts of an application during design time. On deployment, these parts of the application are automatically secured using appropriate security mechanisms to detect and prevent attacks. We present a summary of availability attacks on distributed IACS applications that can be mitigated by our proposed method. Security mechanisms are achieved using IEC 61499 Service-Interface Function Blocks (SIFBs) embedding Intrusion Detection and Prevention System (IDPS), added to the application at compile time. This method is more amenable to providing active security protection from attacks on previously unknown (zero-day) vulnerabilities. We test our solution on an IEC 61499 application executing on Wago PFC200 PLCs. Experiments show that we can successfully log and prevent attacks at the application level as well as help the application to gracefully degrade into safe mode, subsequently improving availability.

4.3 Introduction

The fourth industrial revolution commonly knowns as Industry 4.0, critically relies on networked automation systems with decentralized decision making and connectivity beyond the confines of the factory. Such an exposed posture attracts cyber adversaries that may result in devastating effects if critical industrial processes get disrupted from their normal behavior. Industrial Automation and Control Systems (IACS) enable the automation and control of physical processes right from the factory floor to higher level data analytics at the enterprise level. Communication between SCADA components and the outside world forms a heterogeneous network comprising of different technologies

and protocols. The heterogeneity adds to the complexity of overall system design that may limit protection against adversaries. Consequently, establishing robust security solutions becomes a paramount requirement for SCADA systems.

PLCs are critical components of IACS. They control physical processes and provide floor data to the upper layers for processing. PLCs read data from multiple sensors, processes the data through installed software applications, and then either send messages to other PLCs or drive actuators. A secure IACS needs robust PLC-level protection from attacks on peripherals and networks. In the past, PLC-level security attacks, such as Stuxnet (Nourian & Madnick, 2015), have posed significant threats to critical infrastructure. An analysis of PLC protocols such as UMAS, S7Comm, and Optocomm-Forth on Schneider PLCs revealed several vulnerabilities like user program comprise and alteration, configuration compromise, authentication/access control violation, etc. (Martín-Liras et al., 2017). In another instance, researchers successfully carried out the replay, man-in-the-middle and stealth command modification attack on PLC devices (Ghaleb, Zhioua & Almulhem, 2018). Successful Denial of Service (DoS) attacks were carried out on real PLC devices rendering them unresponsive (Ylmaz, Ciylan, Gönen, Sindiren & Karacayılmaz, 2018).

IACS are meant to run for long durations. It is desirable that the software controlling physical processes in an IACS remains available with minimum downtime, which makes system *availability* a principal goal. Availability attacks, such as DoS-type attacks, on even a single PLC can sabotage the entire IACS (Ylmaz et al., 2018). In fact, for IACS, availability is a more critical security concern than confidentiality and integrity (Knowles et al., 2015). PLCs are vulnerable to external availability attacks as well as internal attacks (Ylmaz et al., 2018; Henrie, 2013). The resource-constrained nature of PLCs makes it harder to provide absolute protection against sophisticated external and internal attacks. Sometimes, an Intrusion Detection and Prevention System (IDPS) may be used to detect and prevent attacks against PLCs (Alves & Morris, 2018). An IDPS

continuously monitors traffic over a network and/or the internal file system or memory footprint of its host. If anomalous or unexpected patterns are detected, it may take preventative actions to secure the PLC. IDPS usually acts as a last line of defense for PLCs in a network. However, even the most sophisticated IDPS may also fail to detect an attack, especially *zero-day* vulnerabilities that are attacked for the first time (Ylmaz et al., 2018).

In this paper, we investigate the case for providing security protection at the application level of a PLC, such that attacks that circumvent an IDPS can be intercepted by the software application running on the PLC. Application-level security needs to be light-weight as PLCs are highly resource constrained. However, this additional layer of security can provide active security protection against zero-day attacks, helping to improve overall system availability. We first study the most commonly-encountered availability attacks on PLCs. For the mitigation, we use the IEC 61499 Service Interface Function Blocks (SIFB) that implements IDPS-like functionality within applications. IEC 61499 provides a highly modular application structure where large applications can be constructed through the reuse of much smaller and pre-verified function blocks from a library. We implement and evaluate a proof-of-concept distributed IEC 61499 function blocks application running over Wago PFC200 PLCs. The security functions are provided through SIFB executing Snort, a well-known IDPS (*Snort - Network Intrusion Detection and Prevention System*, n.d.). It emulates availability related attacks on the proof-of-concept application through multiple tools. Subsequently, performance analysis of Snort in terms of packet dropped over a time interval, running in IEC 61499 environment, is provided. The results show that the Snort drops more packets with the increasing intensity of attacks losing critical data for analysis purposes. At a certain intensity, the PLC device also breaks down, resulting in total denial of service.

The primary contributions of this paper are:

1. A summary of commonly-encountered availability attacks on PLC applications in IEC 61499 environment, presented in Section 4.4.

2. The design of an IDPS-based SIFB for preventing availability attacks, described in Section 4.5.

3. A sample implementation of a secure, distributed IEC61499 application and its evaluation, presented in Section 4.6.

4.4 Availability Attacks on IEC 61499 Applications

This section presents the scenarios for IEC 61499 distributed applications in which availability attacks might disrupt the operations of the system. We list the most common and easily replicable availability attack vectors, mapped to IEC 61499 IACS applications. A hypothesis is formed for each attack scenario that is tested in the subsequent section.

To map availability attacks on a real example, consider a scenario where two cylinders controlled by two cooperating PLCs as seen in Figure 4.1. In Figure 4.1 (a), both cylinders are initially at retracted state while `cylinder 2` is waiting for the box to arrive at its plate. As soon as the box arrives at `cylinder 2`, the controlling PLC detects it through a sensor and commands the cylinder to start extending. When the box reaches at the top, as seen in Figure 4.1 (b), the PLC controlling `cylinder 1` signals it to extended until it pushes the box off the ledge of `cylinder 2`. At this point, both cylinders are at the fully extended position. To get back to the original position, the PLC controlling `cylinder 1` issues a signal to the PLC controlling `cylinder 2` to retract as seen in Figure 4.1 (c).

Figure 4.2 shows the IEC 61499 implementation of case study described in Figure 4.1. The PLC that controls `cylinder 1` hosts `ThrustCtl` and `IX` Function

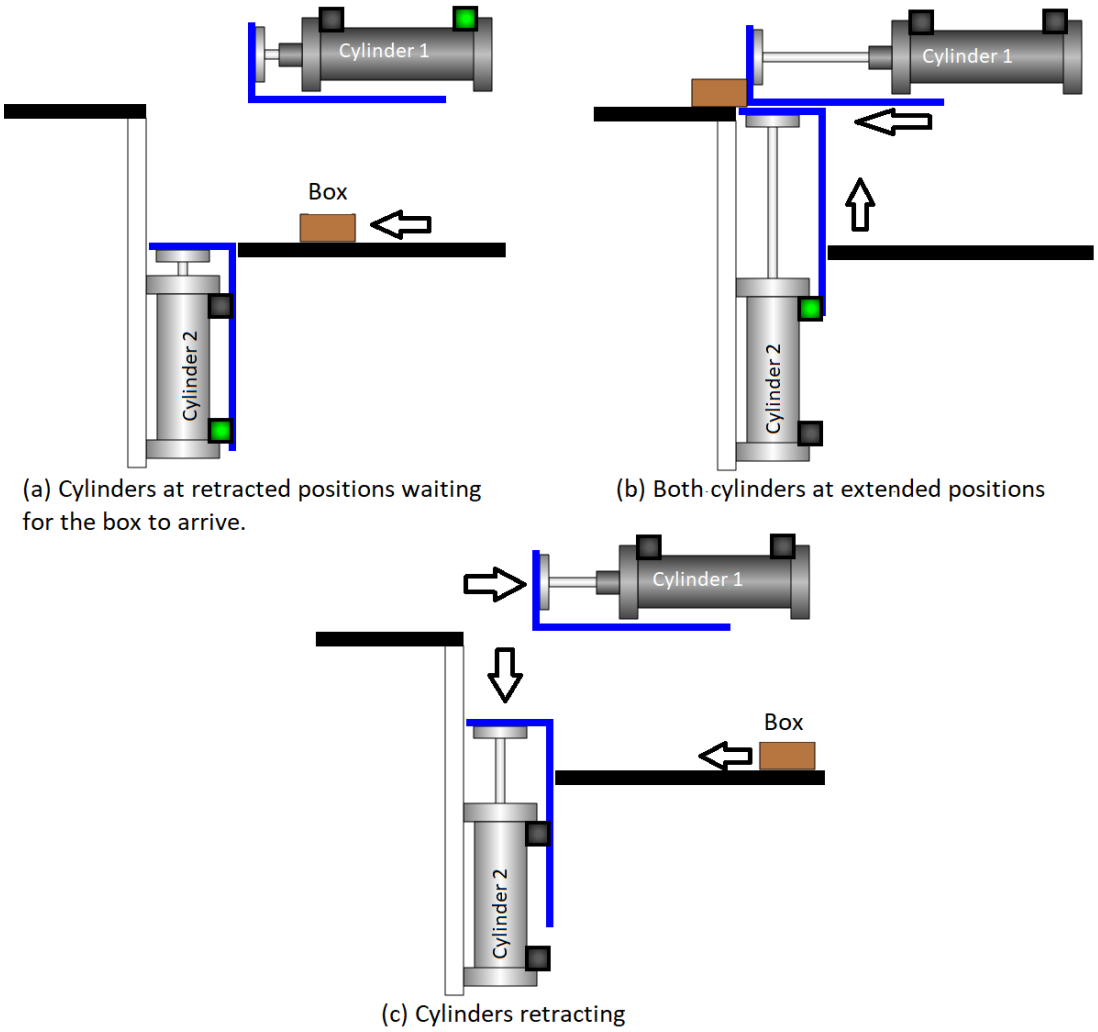

Figure 4.1: Illustration of scenario in which the proposed SIFB has been implemented.

Block (FB)s. Similarly `LiftCtl` and `QX` FBs reside on the other PLC that actuates `cylinder 2`. The `ThrustCtl` FB extends the cylinder on receiving the signal from PLC through `IX` FB. When fully extended, the `ThrustCtl` FB sends the `SharedVariable` to `LiftCtl` on the other PLC while retracting the `cylinder 1`. On receiving the `SharedVariable` data input, `cylinder 2` starts to retract as well.

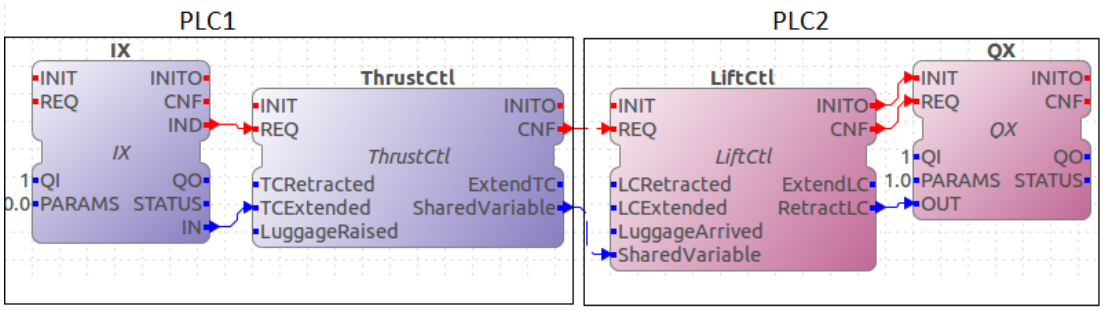

Figure 4.2: IEC 61499 implementation of case study in Figure 4.1

4.4.1 Attack with Malicious or Malformed Data

The application in Figure 4.2 (distributed amongst PLC1 and PLC2) transfers the `SharedVariable` using publisher/subscriber model using UDP multicast network. We assume that there is no confidentiality or integrity support for standard or propriety communication protocol used between the publisher and subscriber. The situation is true for legacy PLCs operating in small to medium-sized industrial setups (Johnson, 2010). Multicast and unicast modes of communication are defined in the HOLOBLOC profile for IEC 61499 standard. In this mode, any host can become a part of a multicast group and send messages to the participants of the group. We construct the following hypothesis for this scenario:

Hypothesis 1 (H1): An adversary can send malicious data to the subscriber or server block by masquerading itself as publisher/client Communication Service Interface Block (CSIFB), causing it to misbehave and subsequent disruption of the intended service.

Assumption(s):

1. There is no mechanism in place for confidentiality or integrity checks.

2. An adversary can send data to the participants in the multicast group.

The attack model is valid for unicast mode as well. IEC 61499 client/server CSIFBs use the TCP mode of communication. Although spoofing a packet is harder in TCP

than UDP, an attacker may apply a replay technique to spoof a TCP packet to hijack the session. In this case, `LiftCtl` FB will act upon the instructions sent in the packet by the attacker instead of the legitimate client.

The `ThrustCtl` FB in PLC1 is sending periodic data from a sensor to the instance in PLC2 that is controlling an actuator based on the received information. The data format can be as simple as presenting integers, in this case, a Boolean variable i.e., `SharedVariable`. An adversary may be able to intrude in the system and may realize the importance of periodic messaging by observing the behavior of the system. It can take advantage of non-existent security controls and may send malicious data to `LiftCtl` in PLC2 using packet crafting techniques by masquerading as PLC1. The subscriber or server block will receive out of sync messages to act on actuators that will cause the system to misbehave or may even cause permanent damage to the machinery. Figure 4.3 (a) illustrates the discussed scenario where an attacker sends a fake packet to `LiftCtl` FB containing `SharedVariable`, forcing uncoordinated actuation from both PLCs.

4.4.2 Application Level Flood Attack

Another scenario is the possibility of DoS as well as Distributed DoS (DDoS) attack where multiple attackers can choke the system to stop or delay the response of service. Figure 4.3 (b) depicts a situation where it is possible to suffocate `LiftCtl` and `ThrustCtl` FBs by sending the flood of malicious traffic from multiple attackers. DDoS attacks are difficult to manage if there are no appropriate security controls implemented in the boundary network. In this case, the communication between publisher and subscriber FBs will be rendered non-functional by the adversaries participating in the DDoS attack as the legitimate traffic from the publisher will not able to reach the subscriber.

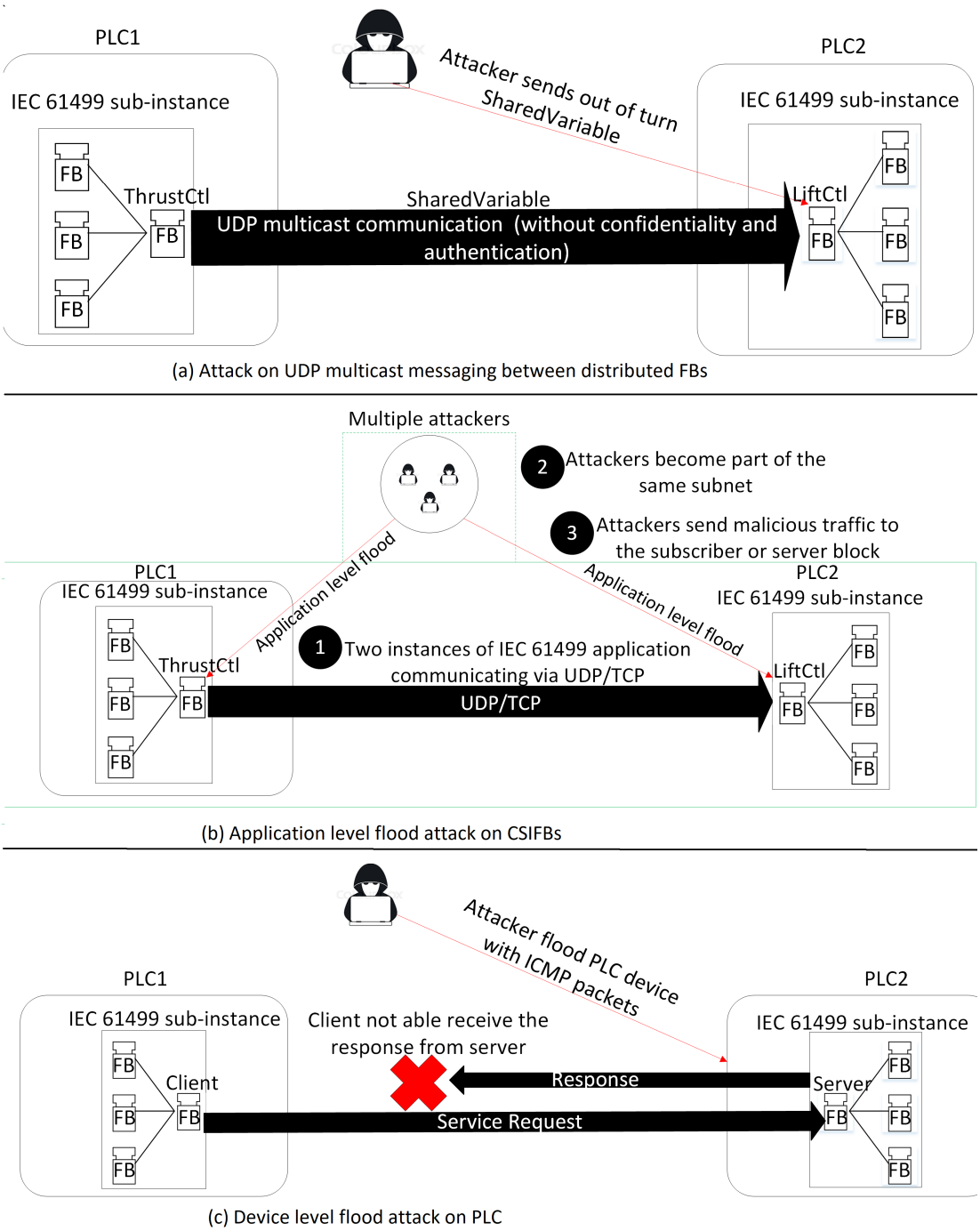

Figure 4.3: A subset of attack scenarios of IEC 61499 distributed applications

Hypothesis 2 (H2): One or multiple adversaries can become a part of the multicast group and flood the publisher/subscriber interface to make it unavailable or slow to respond to legitimate traffic.

Assumption(s):

1. An adversary can send data to the participants in the multicast group or TCP port.
2. An adversary has been able to penetrate through the network firewall.

Again, in the case of an application using client/server CSIFBs, an attacker can flood the port bound by the server CSIFB with invalid requests. TCP attacks such as SYN flood attack is particularly famous for causing a denial of service for TCP sockets. In this attack, repeated SYN packets are sent to the victim at a high rate. The victim will send SYN-ACK packets back to the attacker and wait for the ACK that never arrives, thus leaving the TCP session in a half-open state. If a large number of half-open connections reach a tipping point, the victim will start denying legitimate connection requests. Thus, a server CSIFB with a TCP socket is also vulnerable to such an attack.

The repercussions can be noticed if an attacker manages to flood the TCP/IP ports bound by `LiftCtl` and `ThrustCtl`. The application-level flood attack may overwhelm the FB interfaces with legitimate traffic. Such a scenario may incur no or out-of-sync communication between PLC1 and PLC2 that can halt the system or even inflict irreparable damage to the equipment. Such attacks on PLCs are not unheard of, at least in smart grids where (S. Liu, Liu & El Saddik, 2013) has shown a model of power systems with DoS attacks. It shows that communication channels leading to the load frequency controller of a power system can be jammed, resulting in the loss of telemetered measurements.

4.4.3 Device Level Flood Attack

A PLC may become vulnerable to flood attacks at device level such as ICMP or ping flooding. Such a type of attack inundates the device communicating through TCP/IP stack with brute force i.e., sending a large number of ICMP packets at a rapid rate by a single or multiple attackers. Inadvertently, the device starts consuming its resources to process these packets, consequently denying the resources to legitimate traffic. Although, the problem can be considered in isolation to IEC 61499 applications as it affects the whole system. However, denial of services provided by an instance of distributed IEC 61499 application will eventually affect the other instances running on other PLCs. The distributed nature of IEC 61499 applications means that the instances will be partially or entirely dependent on each other. Unavailability of the services an FB under attack may cause the dependent FB unable to perform a critical system functionality unless mechanisms are put in place in the design of a distributed application by foreseeing DoS attacks. Figure 4.3(c) represents a scenario where a PLC device acting as a server, may be swamped by ICMP flood attack so that it stops responding to legitimate requests by a client device. Therefore, the security of the IEC 61499 distributed environment against device-level DoS attacks may not be considered as a stand-alone problem. Corresponding to the case study in Figure 4.1, a device level flood attack on either PLC1 or PLC2 may overwhelm the PLCs slowing down the response time or may lead to an unresponsive state.

Hypothesis 3 (H3): One or multiple adversaries can flood the PLC running an instance of IEC 61499 distributed application to make it unavailable for other dependent instances.

Assumption(s):

1. An adversary knows the IP address of the PLC device.

2. An adversary has been able to penetrate through the network firewall.

4.5 An SIFB based IDPS

Application and device level attacks can be prevented by a firewall. A firewall's main goal is to block illegitimate traffic based on a packet's header information. On the other hand, an IDPS is also able to analyze the packet's data to raise an alert. IDPS has a distinct advantage over a firewall because it can detect the abnormal behavior of an attacker once it gets in. Thus, an IDPS can essentially be considered as the last line of defense to secure a communication system. Figure 4.4 shows a generic scenario of a network configured to have an IDPS.

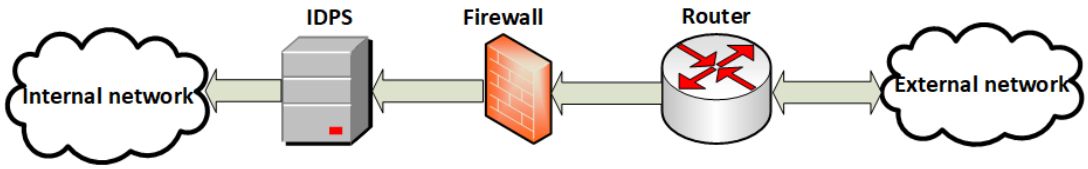

Figure 4.4: A simple network configuration containing IDPS

In small to medium-sized industrial settings, a dedicated IDPS hardware solution is not always cost-effective. The absence of intrusion detection controls puts such environments at high risk. However, IDPS can be deployed in a PLC device, as demonstrated in (Alves, Das & Morris, 2018). The modular structure of IEC 61499 FBs allows the provision of deploying an IDPS module as an FB. In this section, we propose an IEC 61499 distributed application model incorporating intrusion detection and prevention capabilities. Moreover, we also discuss the interface design and utilization of SIFB embedding IDPS services. IEC 61499 standard allows distributed instances of an application to communicate through publisher/subscriber or client/server CSIFBs that put or acquire the data to or from the network. Figure 4.5 illustrates our proposed model of an IEC 61499 application comprising of an IDPS. A CSIFB provides an interface

to the underlying network. It then transfers the data to or from the FBs providing the application's business logic. In the case of Figure 4.5, the CSIFB is configured to receive an input from the network and pass it to the back-end logic component. Therefore, the back-end component is dependent on the CSIFB. The proposed model places an IDPS in between network interface and logic components so that it acts as a Boolean switch. That is, if the IDPS component detects an attack, it shuts off the connection between CISFB and logic components until the attack is on. The designer may decide to suspend the application instance or completely shut down the process depending on the requirements.

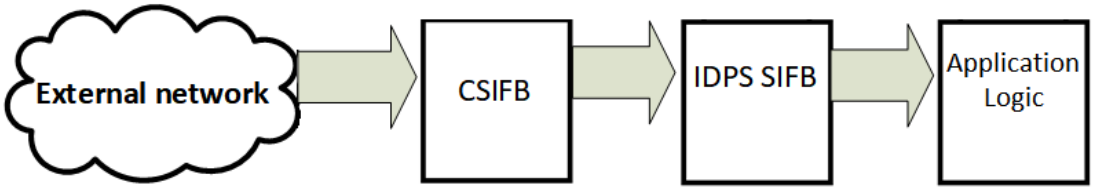

Figure 4.5: Proposed configuration of IDPS SIFB in IEC 61499 distributed applications

The model can essentially be realized by creating a SIFB that starts the IDPS process in the background upon receiving the `INIT` event. The `stop` event is used to stop the IDPS process. The `PARAMS` data input is used to provide the necessary configuration parameters to the process. The `STATUS` data output can be used to check the state of the process. The `IDPS_SIFB` is essentially an integral part of a composite FB (CFB) called `IDPS_CFB`. Figure 4.6 shows the internal FB network of `IDPS_CFB`. It includes `IDPS_SIFB` and an `ALERTCHECK` FB which continuously checks the output of `IDPS_SIFB` to determine the occurrence of an attack by looking for alerts generated by an IDPS. If the attack is on, `ALERTCHECK` FB will raise a Boolean flag through `QO` variable that is connected to `A` data output of the CFB. The value can subsequently be consumed by the encapsulating application to enable the appropriate action if a PLC device or an application is under attack.

The most important point of reference in the system is the extended position of both

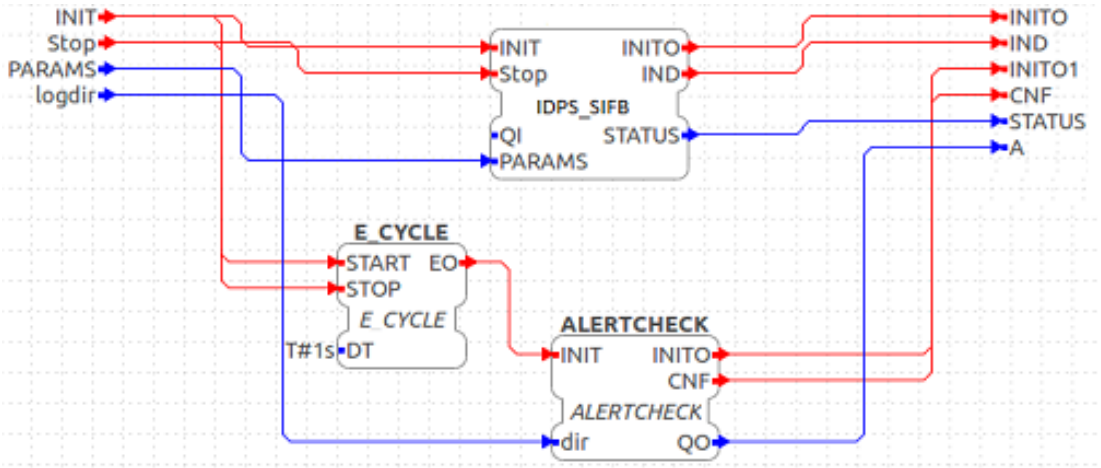

Figure 4.6: A CFB containing IDPS_SIFB and associated ALERTCHECK FB.

the cylinders. At this stage, the `ThrustCtl` FB causes the `cylinder 1` to retract, also switching on the `SharedVariable` Boolean variable and sends it to `LiftCtl` controlling `cylinder 2` on the other PLC. On the receiving `SharedVariable`, the `LiftCtl` FB signals the PLC to retract `cylinder 2`. Therefore, the only point of interaction between two PLCs is the exchange of `SharedVariable`. In the case when Ethernet connects the PLCs, the data travels in TCP or UDP packets depending on the mode selected i.e., client/server or publisher/subscriber model. In the case study, the communication between two PLCs controlling the cylinders is vulnerable to the availability attacks presented in Section 4.4. For example, an attacker can interrupt the transmission of `SharedVariable` by rendering one of the PLC unresponsive through a DoS or DDoS attack. In that case, the congruity of the system shall be affected that may lead to irrecoverable physical damage to the equipment in focus.

However, the availability of a mechanism that can detect such attacks may enable designers and developers of an IEC 61499 to pre-empt against such attacks. In the case of IDPS, the attack can be prevented and blocked so that the system may continue to perform its routine task. On the other hand, an Intrusion Detection System (IDS) can report the occurrence of an attack, prompting the system to go into a defensive

posture e.g., halting the system for a certain amount of time or any discretionary actions. A possible solution is provided in Figure 4.7 to establish the application of the aforementioned `IDPS_CFB`. In an event of availability attack on the PLC controlling `cylinder 2`, the `IDPS_CFB` is able to report on the attack. If the Boolean variable `A` is true, the `LiftCtl` FB is not able to receive the shared variable resulting in halting the process. Such a mechanism is useful if an out of sync `SharedVariable` input arrives due to an ongoing attack.

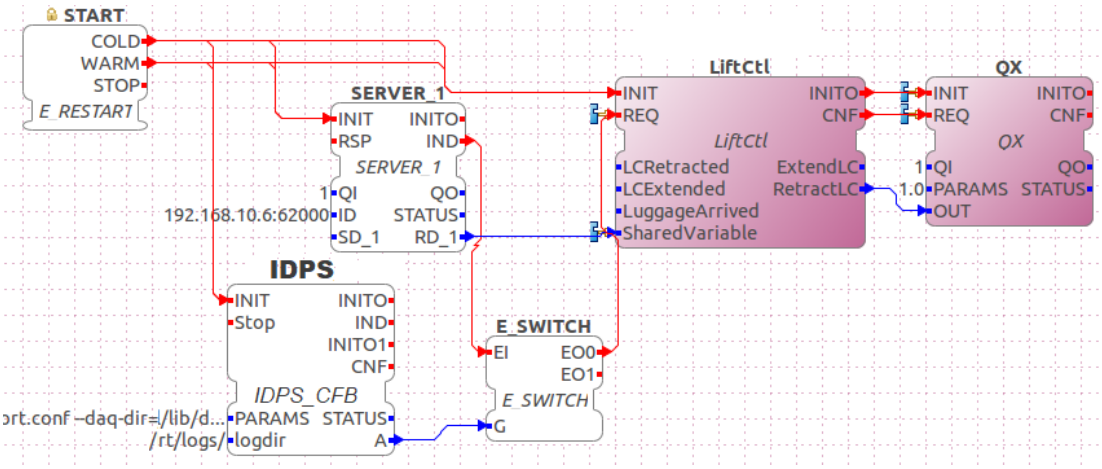

Figure 4.7: Application of `IDPS_CFB` with `LiftCtl` FB.

Moreover, the optional `E_SWITCH` FB in Figure 4.7 connects the `IDPS_CFB` to the application logic. In intrusion prevention mode, it performs an important job to switch off the application logic during the occurrence of an attack. However, in the intrusion prevention mode, when `IDPS_CFB` is actively blocking the malicious traffic, the `E_SWITCH` FB can be removed to create a direct connection between IDPS and application logic. It will enable the designer to log the defined attacks while the application is performing its regular tasks. It will positively affect the availability in a way that the application will have minimal downtime and also allows mitigation measures simultaneously.

4.5.0.1 Active Security Protection

An important factor in protecting against availability related attacks is to put the system into a security posture that is active/proactive rather than reactive. Although organizations are investing more in security infrastructure as IACS are becoming more social, attackers are also evolving armed with sophisticated attack techniques. The signature or rule-based IDPSs, although convenient, cannot detect an innovative attack that does not make part of their ruleset or signatures. A possible proactive approach is to use anomaly-based IDPS using Machine Learning (ML) techniques to extract useful and harmful patterns. Thus a model can be obtained for correct system behavior. Alarms are raised when such an IDPS detects anything that defies the model. However, ML techniques usually require more processing power that is a limitation for PLC devices in IACS. Therefore, lightweight ML algorithms may be chosen to get the best of both worlds in such devices. An advantage of using the ML-based approach, striving to provide active security protection is its suitability for detecting zero-day attacks to some extent. The method to use rule-based Snort IDPS with ML algorithms has been discussed in (Shah, Issac & Jacob, 2018). It uses ML-based intelligent plugin along with Snort to classify good and bad traffic. The ML plugin runs parallel to Snort's detection engine with the ability to set alarms for newly identified attacks. However, a possible extension to this approach will be to feed newly detected attacks back to Snort in the form of new rule sets. A proactive security posture can be achieved for IEC 61499 environment by using the proposed `IDPS_SIFB` using lightweight ML algorithms.

4.6 Implementation and Results

This section partially implements the case study described in Section 4.4. Two separate Wago 750-8206 PFC200 PLCs control both cylinders. Each PLC has an ARM one core

Cortex A8 600 MHz microprocessor with 256 MB main memory running real-time Linux. The Wago PLC supports IEC 61499, which was used to develop the application. The PLCs communicate over Ethernet connected via a switch. We used the 4diac IDE and 4diac FORTE (*Eclipse 4diac™ - 4diac IDE and 4diac FORTE runtime*, 2021) to develop the application logic, and then cross-compiled it to run on real-time Linux in the Wago PLCs. Sensor input to PLC1 indicates that `cylinder 1` is fully extended. In response, it sends the `SharedVariable` signal to PLC2. Consequently, PLC2 commands `cylinder 2` to retract in the form of an actuator signal. In our set up, we used a metallic proximity sensor as the sensor input to PLC1. Figure 4.8 shows the experimental set up used to imitate the partial system behavior.

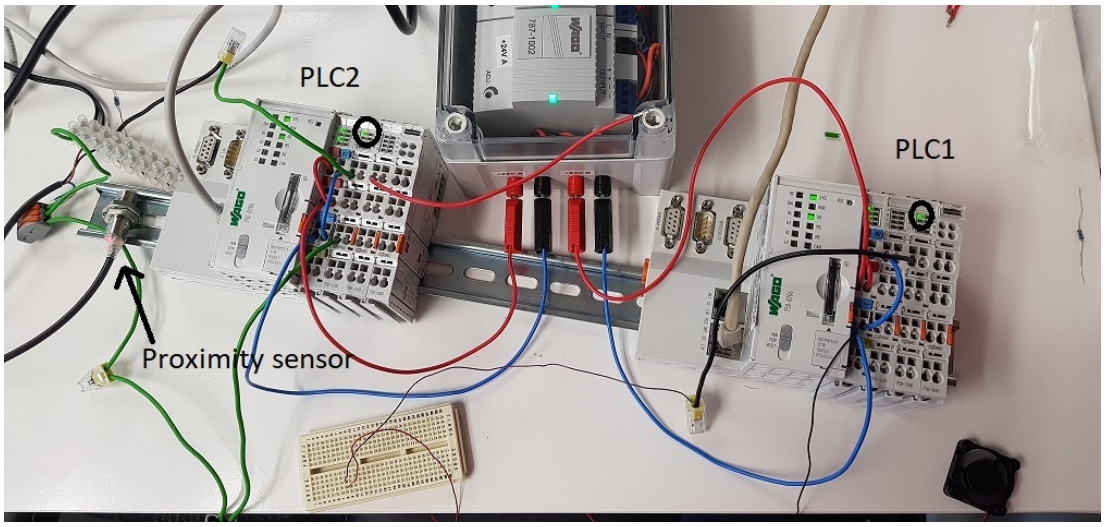

Figure 4.8: Experimental set up with Wago 750-8206 PFC200.

We have used the Snort IDPS (*Snort - Network Intrusion Detection and Prevention System*, n.d.) to implement `IDPS_CFB`, the block that implements the IDPS at the application level. Snort can run in IDS as well as IDPS modes, but we have so far only implemented the IDS mode in `IDPS_CFB`. We emulate DoS attacks using the $hping3$ tool (*hping - Active network security tool*, n.d.) which sends a large number of ICMP and TCP packets to PLC1. The `INIT` event of `IDPS_CFB` starts a new Snort process externally to the 4diac FORTE runtime. Snort rules are configured in a configuration

file to detect availability attacks. Such rules enable Snort to detect any unwanted traffic and raise alerts or block traffic in IDPS mode. As Figure 4.9 shows, `SnortCFB` sets variable `A` when it detects traffic violating the rules.

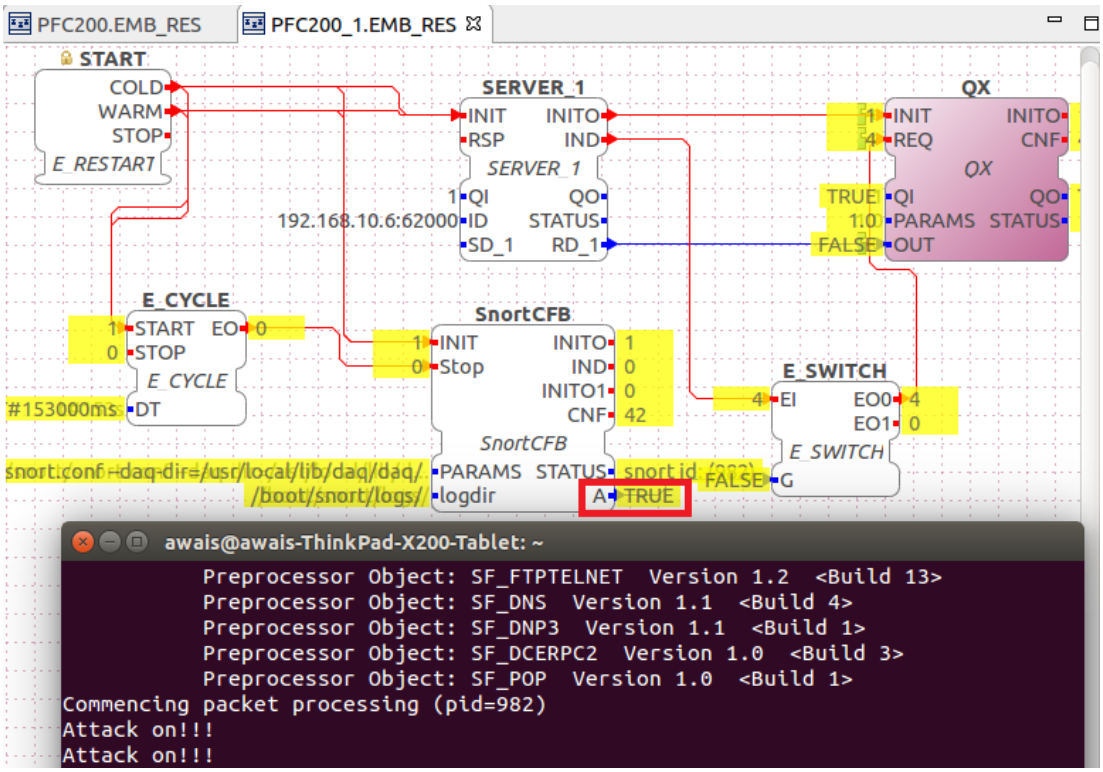

Figure 4.9: Snort based `IDPS_CFB` detecting attacks.

To prove hypothesis *H1* in Section 4.4, the *PackETH* (*packETH*, n.d.) tool is used to send a spoofed UDP packet to `cylinder 2` containing the value of `SharedVariable`. It causes out-of-sync actuation of `cylinder 2` at PLC2. For hypotheses *H2* and *H3*, *hping3* is used to send a large number of packets to the PLC host and its ports used by the 4diac FORTE runtime.

It is observed that the number of packets dropped by Snort as the packet frequency increased. In high-intensity attacks, Snort fails to log the attack correctly. When *hping3* was configured with the *–faster* option to send packets each microsecond, the PLC itself becomes completely unresponsive which halts the operation of the system. It shows that a sufficiently powerful attacker can succeed even in the presence of an IDPS

system embedded in the IEC 61499 application. However, the application-level IDPS can be very useful in logging and/or filtering out illegitimate traffic that escapes other mitigation strategies during low to medium intensity attacks.

The demonstrated behavior of IDPS shows the viability and applicability of PLC-based solutions in industrial settings that may improve the overall availability of the PLC and thus the system under consideration. The `IDPS_CFB` is essentially the last resistance in the path of an attacker if it can breach upper layers of an IACS. The ideal choice in such a setup is an underlying lightweight IDPS to serve the balance between security and other properties of the system that are affected by the PLC such as safety and efficiency. However, identifying the appropriate IDPS suitable to PLCs deployed in different environments is a constraint on the solution provided in the research.

4.7 Conclusions and Future Work

In this paper, we have enumerated the most common attack vectors that may affect the availability of distributed industrial applications. We propose an IEC 61499 SIFB based IDPS solution to protect PLCs against such availability attacks. An implementation A prototype application was built to implement this solution using Snort IDPS. Results show that under low to medium attacks, the IDPS can successfully log illegitimate traffic, providing the last line of defense against attackers. This work is a first step towards the use of more sophisticated detection and prevention algorithms to provide active security protection at the application level. Future research directions include formalizing the solution as a replicable design pattern in IEC 61499, creation of a SIFB library for different detection and prevention algorithms, and the design of high availability IACS through system-, PLC-, and application-level mitigation. Moreover, we also plan to carry out the detailed comparison, evaluation, and benchmarking of `IDPS_CFB` using multiple off-the-shelve IDPSs in the future.

Chapter 5

Secure Links: Secure-by-Design Communications

5.1 Prelude

The following chapter is published as a journal article in ***IEEE Transactions on Industrial Informatics*** under the title of ***Secure Links: Secure-by-Design Communications in IEC 61499 Industrial Control Applications***.

This work improves and consolidates the concept of secure links first initiated in Chapter 3. Secure links are design-time abstractions proposed as an extension to the IEC 61499 development standard. A novel concept of secure links' interaction with the TORUS traceability tool is introduced for providing end-to-end traceability for the security requirements.

5.2 Abstract

Increasing automation and external connectivity in Industrial Control Systems (ICS) demand a greater emphasis on software-level communication security. We propose a

secure-by-design development method for building ICS applications, where requirements from security standards like ISA/IEC 62443 are fulfilled by design-time abstractions called *secure links*. Proposed as an extension to the IEC 61499 development standard, secure links incorporate both light-weight and traditional security mechanisms into applications with negligible effort. Applications containing secure links can be automatically compiled into fully IEC 61499-compliant software. Experimental results show secure links significantly reduce design and code complexity and improve application maintainability and requirements traceability.

5.3 Introduction

Application-level communication security has become a key concern in Industrial Control Systems (ICS) that control critical systems like smart grids, manufacturing plants, and nuclear facilities. ICS run highly distributed software applications deployed on resource-constrained devices. Combined with increasing automation and external connectivity, these factors make ICS very vulnerable to security attacks. Malicious disruption of ICS can be extremely expensive (Ashibani & Mahmoud, 2017), which necessitates the use of *secure-by-design* (Mouratisis, 2010) approaches for developing ICS applications. Modern ICS must adhere to security standards such as ISA/IEC 62443 (62443-1-1, 2009), Common Criteria, and NIST SP 800-82. ICS in critical infrastructure are increasingly being constructed through the reuse of certified secure components (Lendvay, 2016). These trends mean that current methodologies for developing ICS applications must urgently move from a “security as an after-thought" approach to a systematic, secure-by-design mindset (Graham, Hieb & Naber, 2016). ICS applications are deployed onto multiple devices and involve extensive communication between devices. Frequent reconfiguration is supported by standards like IEC 61499 (Vyatkin, 2009), but current approaches lack a secure-by-design approach to

communication security.

Industrial Mixer Control System (IMCS), shown in Figure 5.1, is used to highlight the communication security challenge in ICS and subsequent concepts introduced in this article. The IMCS is a distributed IEC 61499 application executing on two Programmable Logic Controllers (PLCs) $PLC1$ and $PLC2$. The PLCs control the sensors and actuators associated with tanks 1 and 2, respectively. Tank 2 is responsible for maintaining a desired viscosity and level (volume) of a fluid solution. The (part of the) application running on $PLC2$ controls the inlet valve IV1 through which the solution enters the tank, the mixer motor MM to carry out the mixing, and the outlet valve OV1. The application uses readings from the level sensor FS2 and the viscosity sensor VS to ensure correct operation. Tank 1 heats the fluid solution to the desired temperature. The application running on $PLC1$ controls the inlet valve and flow restrictor IVF to allow the pre-mixed solution from tank 2 to enter tank 1, the cartridge heater CH to heat the solution, and the outlet valve OV2. The application uses readings from the level sensor FS1 and the temperature sensor to ensure correct operation.

Safe operation of the IMCS requires coordination between the applications running on the two PLCs. When $PLC1$ detects that more fluid is required in tank 1, it opens the IVF valve and sends a fluid acquisition request to $PLC2$ for it to release the OV1 valve. When the fluid in tank 2 is ready, $PLC2$ opens the OV1 valve to move the fluid from tank 2 to tank 1. It is vital that OV1 is only released after IVF to ensure that no fluid is left sitting in the pipe between tank 2 and tank 1. If the fluid is not mixed or heated for a prolonged time, it can harden and cause blockages. The PLCs also interact with an HMI to provide a real-time view of the system.

The IMCS contains typical security vulnerabilities found in ICS. In a *replay attack*, an attacker is able to replay messages between the PLCs. It can request $PLC2$ to open OV1 when IVF is still closed, causing the fluid solution in the pipe to harden and cause blockages. The IMCS is also vulnerable to a *man-in-the-middle attack* (Staggs

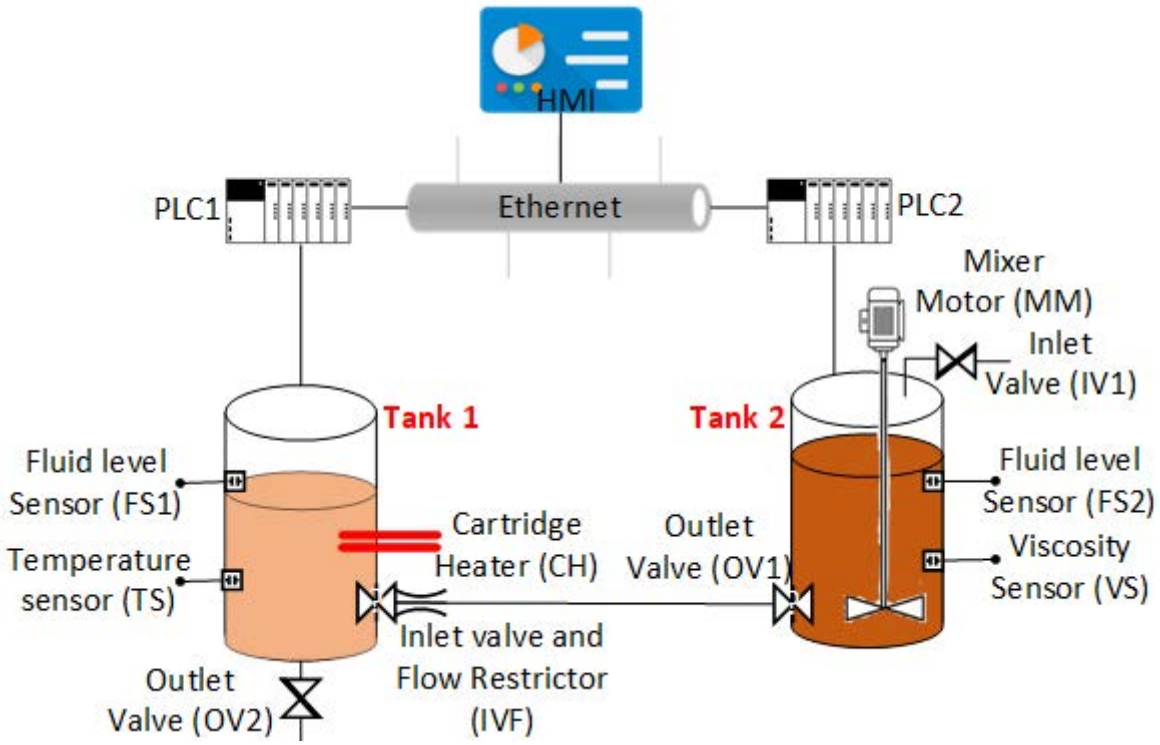

Figure 5.1: Industrial Mixer Control System.

et al., 2017), where an attacker intercepts the messages between the PLCs and/or the HMI. It can starve tank 1 by blocking all OV1 release requests from $PLC1$ to $PLC2$. Alternatively, the attacker can send sporadic false commands to the PLCs to cause further disruptions or unsafe operation. Such attacks affect the safety and availability of the IMCS, potentially causing physical damage to the system itself and disrupting other systems dependent on the IMCS.

Traditional, widely-used approaches for communications security in other domains are not well-suited for use in ICS due to the security-performance trade-off (Hogan, Piccarreta, Group et al., 2018). Approaches like Transport Layer Security (TLS), Datagram TLS, and Virtual Private Network (VPN) induce higher communication latency which may prevent ICS applications from meeting real-time requirements. Secondly, methods like TLS, for example, are point-to-point secure communication protocols more suitable for client-server architectures. ICS, on the other hand, tend to use unidirectional and

multicast communications between application slices. Thirdly, traditional methods may require more resources than are available on resource-constrained PLCs. Subsequently, maintaining a balance between the demands of communication security in ICS becomes a key challenge (Müller, Walz, Kiefer, Doran & Sikora, 2018). Hence, for an ICS application developer, the only available option to protect against such attacks is to include communication security mechanisms manually into the application. This approach does not allow applications to be easily reconfigured, sacrificing a key feature of ICS. For the IMCS, while one can use TLS to secure all communications between the two PLCs, the overhead introduced by this approach may affect the safety of the system, which is dependent on real-time responsiveness. For example, depending on the use of a primitive traditional method, may result in missing the threshold time for opening/closing of IVF and OV1 valves that may cause significant damage to the IMCS by overfilling tank 1.

Developing standards-compliant secure ICS applications has further challenges. Standards like ISA/IEC 62443 provide security requirements that must be met by certified systems. Certification requires a rigorous and expensive process of ensuring the system has met each security requirement. The precise set of requirements to be met depends on factors like target security levels and the security-performance trade-offs. Hence, a one size fits all approach, such as TLS, is rarely a feasible solution. Unfortunately, there is currently no systematic way to *trace* or link requirements from standards to parts of code that they are addressed in (62443-4-1, 2018), making it extremely difficult to certify ICS applications that are often reconfigured and have real-time performance requirements.

A secure-by-design approach is proposed for developing ICS applications called the *Secure Links Development Method* (SLDM). SLDM, shown in Figure 5.2 as a UML Component Diagram, is a systematic process to develop secure ICS applications compliant to standards like IEC 62443. SLDM has been designed to adhere to two major

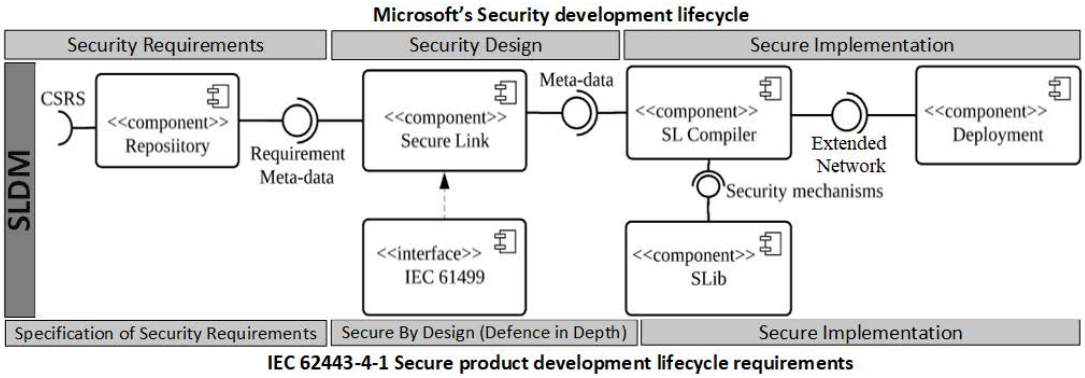

Figure 5.2: An overview of the SLDM

security development lifecycles: Microsoft's security development method (Howard & Lipner, 2006) and the secure product development lifecycle requirements process from IEC 62443-4-1 (62443-4-1, 2018). To manage requirements from security standards, SLDM uses the Traceability Of Requirements Using Splices (TORUS) framework (Sinha et al., 2018), presented in Section5.5.1, which provides a systematic way to organise requirements from a system's Cyber Security Requirements Specification (CSRS) (Gunter, Medoff & OBrien, 2018) and links them to application designs and code that implement them. Next, we introduce design abstractions called *secure links*, presented in Section 5.5.2, which help include communication security mechanisms into IEC 61499 applications in a uniform, consistent, maintainable, traceable, and reusable manner. Secure links contain references to requirements from the TORUS repository and reusable communication security mechanisms that can help meet these requirements. SLDM enables designers to flexibly select the most suited secure communication mechanisms for each part of their ICS applications, from lightweight cryptography to traditional TLS based approaches, depending on target security requirements. It prevents the need to manually code security mechanisms, resulting in significant reductions in developing secure applications. SLDM is fully compatible with the traditional way of constructing ICS applications so that security mechanisms can also be manually coded into an application, in addition to using secure links when developing secure-by-design

applications.

A secure link compiler, presented in Section 5.6, converts application designs into fully IEC 61499-compliant code that can be deployed onto target devices like PLCs. The compiler transforms secure links into code for the chosen security mechanisms by using pre-verified implementations of these mechanisms from a security library. The compiler considers the current configuration of the ICS application to secure only those communications that happen between parts of the application that are deployed over different devices. When the configuration changes, the compiler can re-generate efficient code that is suitable for meeting the updated communication security needs. The code generated by the compiler retains its link to the requirements in the CSRS, considerably reducing the effort required for certification. Experimental results, presented in Section 5.7, reinforce that updating, adding, and deleting secure links in an application requires a negligible effort, and applications remain highly maintainable. On the other hand, when the same systems are developed using a traditional approach, each additional security mechanism introduced into the software causes a significant increase in software complexity and a similar decline in maintainability.

5.4 Related Works

Recently updated security standards such as ISA/IEC 62443 have become important cornerstones of ICS security in the industry. The recently-formed Global Cybersecurity Alliance (GCA) comprising of leading industrial players like Schneider, Rockwell and Honeywell Automation, has decided to use ISA/IEC 62433 as an interchangeable medium to share security-related knowledge and expertise with end-users, manufacturers, and government agencies (Drivers & Controls, 2019). Certification schemes for ISA/IEC 62433, such as Embedded Device Security Assurance and Component Security Assurance, have also gained in popularity, resulting in several devices from

large-scale industrial manufacturers being certified (*IEC 62443 Conformance Certification*, 2019). Moreover, the manufacturing industry is rapidly moving towards Industry 4.0 (Lasi et al., 2014) that extensively utilises ICS. Standardisation efforts of cyber security requirements for such an environment have focused on the introduction of new standards as well as the adaption of more generic standards like ISO 27001. Some limitations of industry 4.0 and IoT compatible security standards like ISO 27001, IEC 62443, ETSI TS 103 645, and the recommendations by the European Union Agency for Network and Information Security, have been discussed in (Culot, Fattori, Podrecca & Sartor, 2019). The study finds that the implementation of these standards is complex and time/resource intensive, and requires better processes and frameworks in the future.

Several existing works support organising requirements in a systematic manner. Microsoft's security development lifecycle (Howard & Lipner, 2006) provides a generic approach to integrate security requirements into software development. IEC 62443-4-1 (Secure product development lifecycle requirements) (62443-4-1, 2018) provides a more ICS-specific approach. Other works like the Security Requirements Engineering Process (Mellado et al., 2007) use standards like Common Criteria for eliciting requirements. In (Morimoto, Horie & Cheng, 2006), relational databases are used to organise requirements from standards for use during development. In (Mellado et al., 2007), a security resources repository is integrated with requirements from Common Criteria. In (Toval, Nicolás, Moros & García, 2002), a repository is used for requirements reuse. These existing approaches either rely heavily on textual requirements or do not specify a storage method for requirements. In (W. Wang, Niu, Alenazi & Da Xu, 2018), a survey of different mechanisms to create and maintain traceability in PLC and SysML models is presented, which highlights the need for automatically creating and maintaining traceability between requirements and system components. TORUS (Sinha et al., 2018) presents an ICS-specific approach towards requirements traceability, but it has not been used for security.

Several existing frameworks offer a structured approach to building ICS applications. A design framework that inherently supports formal validation and verification of automation applications, through integrating Computer-Aided Design, Unified Modeling Language, MATLAB/Simulink, and IEC 61499 is proposed in (Vyatkin, Hanisch, Pang & Yang, 2008). Another design and development framework, called “methodology for industrial automation systems" (Alvarez, Sarachaga, Burgos, Estévez & Marcos, 2016), generates analysis and design documentation of functional and non-functional requirements as structured documents and use-case diagrams, subsequently used to generate code structures. This framework, however, lacks traceability support. vueOne (Harrison, Vera & Ahmad, 2016) is an end-to-end IDE for developing Industry 4.0 applications, but it does not support requirements or their traceability. In (Fay et al., 2015), a model-based engineering workflow for IEC 61131-based distributed manufacturing automation systems is presented. Requirements modelled using SysML are traced through the design and development phases. However, the overall approach does not support security engineering.

Secure communication is of particular concern in IEC 61499 applications. Publisher/subscriber models for unidirectional communications use the multicast mode of communication according to conformance profiles like HOLOBLOC and protocols like PROFINET. Bidirectional communications using the client/server model may use Transport Layer Security (TLS) between application slices deployed onto multiple devices. However, secure multicast communication is challenging in ICS when using Datagram TLS or IPSEC-based virtual private networks (Müller et al., 2018). Other challenges of applying TLS may include performance trade-offs, channel multiplexing, constrained components, time synchronization. There is a strong case of using OPC-UA security for ICS communication; however, its implementation in an ICS is not straightforward, mainly due to the large resource footprint and lack of real-time

capabilities (Veichtlbauer, Ortmayer & Heistracher, 2017). In an ICS, not all communications need securing, with partial protection of data being sufficient for many applications (Yang, Lu, Choo & Liu, 2015). Selective protection of “data of interest" helps in better system performance while ensuring adequate security. In such cases, a blanket approach like TLS may induce latency overheads (Müller et al., 2018).

Lightweight cryptography includes transforms for resource-constrained devices featuring better memory footprints and performance than traditional cryptographic primitives. Ongoing efforts to standardize lightweight security mechanisms include NIST’s new lightweight cryptography standardization selection process (Turan, McKay, Çalık, Chang & Bassham, 2019). Mainstream IT system security solutions like TLS and IPSEC do not contain lightweight ciphers (Hogan et al., 2018). Lightweight VPNs such as wireguard (Donenfeld, 2017) are expected to find use in industrial settings in the future.

Current security standards provide comprehensive sets of requirements. However, in the absence of processes that integrates a security focus and the development process, the implementation of such standards remains challenging. Standalone methods and framework discussed above are disjointed and do not provide a comprehensive end-to-end engineering solution for secure-by-design ICS applications. Current literature reports gaps like a lack of security standard requirements specification, end-to-end requirements traceability, and change management. Development standards like IEC 61499 do not inherently support secure-by-design constructs. Moreover, the IEC 61499 network communication models and constrained ICS environments pose a significant challenge in balancing security and performance. This article proposes a process for developing secure-by-design ICS applications, which alleviates several gaps in current research.

5.5 Secure Links for Application Design

5.5.1 Security Requirements Repository

IEC 62443-4-2 “Security for industrial automation and control systems - Part 4-2: Technical security requirements for ICS components" (ISA/IEC, 2018) provides generic requirements for achieving specific *security levels*. Security requirements from possibly multiple standards are organized in a Cyber Security Requirements Specification (CSRS). Consider, for example, two requirements for the IMCS case study:

- $SR1$*: prevent disclosure of critical parameters to an unauthorized party during transmission on external interfaces.* $SR1$ is linked to *Foundation Requirement 4* of IEC 62443 relating to data confidentiality, and more specifically to IEC 62443-4-2 Component Requirements (CRs) *CR4.1* (Information confidentiality) and *CR4.3* (Use of cryptography), for security level L4.

- $SR2$*: ensure appropriate data integrity of critical parameters during transmission on external interfaces.* $SR2$ is related to *Foundation Requirement 3, CR3.1* of IEC 62443-4-2 that deals with communication integrity.

The TORUS (Sinha et al., 2018) framework is used to organise security requirements and achieve traceability (illustrated as phase 1 in Figure 5.2). Figure 5.3 shows how requirements are organised in a TORUS-based CSRS repository. The cluster labelled as “IEC 62443" shows how TORUS uses a graph structure to retain the hierarchy of requirements from a security standard. The cluster labelled “CSRS" shows the references to system requirements. Cross-linkages or *splices* are used to connect generic requirements from the standard, such as *CR4.1* and *CR4.2*, to system-specific requirements, such as $SR1$. Requirements engineers first create these splices. A TORUS-based

requirement repository RR can be seen as a set of requirements $\{r_1, r_2, ...r_n\}$. For Figure 5.3, RR = $\{SR1, SR2\}$.

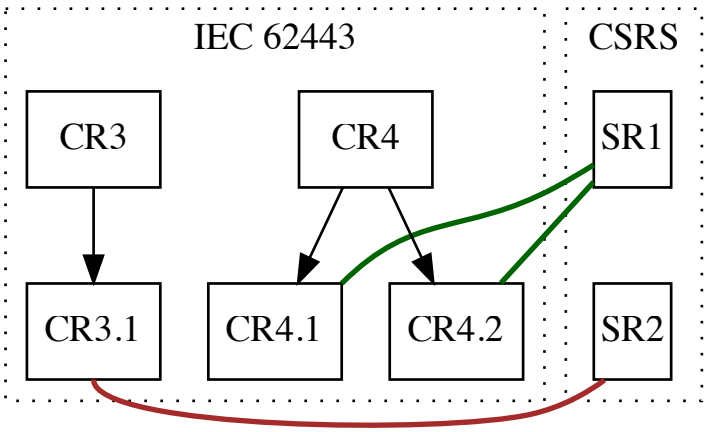

Figure 5.3: A TORUS based CSRS repository

TORUS provides a number of useful features for traceability. Each requirement in the repository can be annotated with additional meta-data including its *scope* (system or component) and the target security-level. SLDM is concerned only with communications security and therefore operates on a subset of the requirements in the CSRS. These requirements tend to be component-level in nature as they relate to the interaction between components. TORUS can automatically update splices when the documents containing standard and/or the system-specific requirements change. TORUS provides graph algorithms to determine important metrics like coverage, which are described in more detail in (Sinha et al., 2018). As we describe later in Section 5.6, TORUS is used to trace standards-based requirements all the way to the deployed code, which significantly reduces security standard compliance costs.

5.5.2 Secure Links

Adding security mechanisms to an ICS application requires using the same constructs, such as function blocks (FBs) in IEC 61499, that are used to build the control logic and algorithms. While this provides familiarity, it can obscure the boundaries between

security mechanisms and other parts of the application, affecting code maintainability, application reconfigurability, and standard certification costs. In this section, we propose *secure links*, which are reusable design abstractions for building secure-by-design IEC 61499 applications. Secure links allow designers to flexibly include both lightweight and traditional methods like TLS to secure different parts of an ICS application, depending on the target security requirements and considering the security-performance trade-off. Secure links extend *function block networks*.

Definition 1. FB Network An FB network is defined as a pair $fbn = (FB, C)$ where FB is a set of function block instances. For each $fb \in FB$, we define the following:

- $fb.IE$ and $fb.OE$ are sets of event inputs and outputs.
- $fb.IV$ and $fb.OV$ are sets of data inputs and outputs.
- An association function $fb.\mathcal{A} : fb.IV \cup fb.OV \rightarrow fb.IE \cup fb.OE$ maps each input/output variable to a unique input/output event associated with it[1].

C is a set of connections. Each connection $c \in C$ is described as a pair (src, trg) such that:

- $c.src \in \bigcup_{i \in |FB|} (fb_i.OV \cup fb_i.OE) \cup \{\square\}$ is the source.
- $c.trg \in \bigcup_{i \in |FB|} (fb_i.IV \cup fb_i.IE) \cup \{\square\}$ is the target.

$\square$ indicates an unconnected input or output. Furthermore, C is restricted as follows:

- For any connection $c \in C$, if $c.src$ is an event (var), then $c.trg$ must also be an event (var) or $\square$, and vice versa.
- C cannot contain a connection $(\square, \square)$.

[1]Variables can be associated with multiple events. Restricting to one-to-one associations is for readability and does not affect the approach's generality.

- For any *data* connections $c_1, c_2 \in C$, $c_2.trg \neq c_1.trg$.

Figure 5.4 shows the FB network for the IMCS application containing four interconnected FB instances. The instance `MTank` of type `MainTank` contains input/output events and variables like `INIT`, `ViscS`, `INITO` and `InReq`. FBs contain associations, such as between `STank.InReq` and `STank.IR` (not depicted in network diagrams). Unidirectional event and variable connections include `(MTank.IR,STank.IR)` and `(MTank.InReq,STank.InReq)`. Open connections include `(MTankPub.MData,□)`. Note that top-level FB networks, such as in Figure 5.4, are *applications*, which can be deployed onto multiple devices based on a configuration mapping. In Figure 5.4, instances `MTank` and `MTankPub` are deployed onto *PLC1*, and remaining blocks are deployed onto *PLC2*, as illustrated by the coloring of the instances. Formally, given an application fbn and a set D of available resources, the deployment configuration or *mapping* is defined as $\mathcal{D} : fbn.FB \rightarrow D$.

When using secure links, designers choose appropriate security mechanisms for which pre-verified implementations are available in a *security library*.

Definition 2. Security Library A Security Library $SLib$ is a finite set of *security mechanisms* (FB networks) $\{sm_1, \ldots, sm_n\}$, such that each $sm \in SLib$ has a signature $sm.\mathcal{S} = (cin, cout, params)$ where:

- cin is a variable connection where $cin.src = \square$.
- $cout$ is a variable connection where $cout.trg = \square$.
- $params \subset \bigcup_{i \in |FB|} fb_i.IV$ is a set of unconnected input variables or parameters.
- $sm.FB$ is partitioned into sets $sm.FB_a$ and $sm.FB_b$. $sm.FB_a = \varnothing$ implies a `receive` mechanism where $sm.FB_b = \varnothing$ implies a `send` mechanism. Otherwise, the mechanism is `send-receive`.

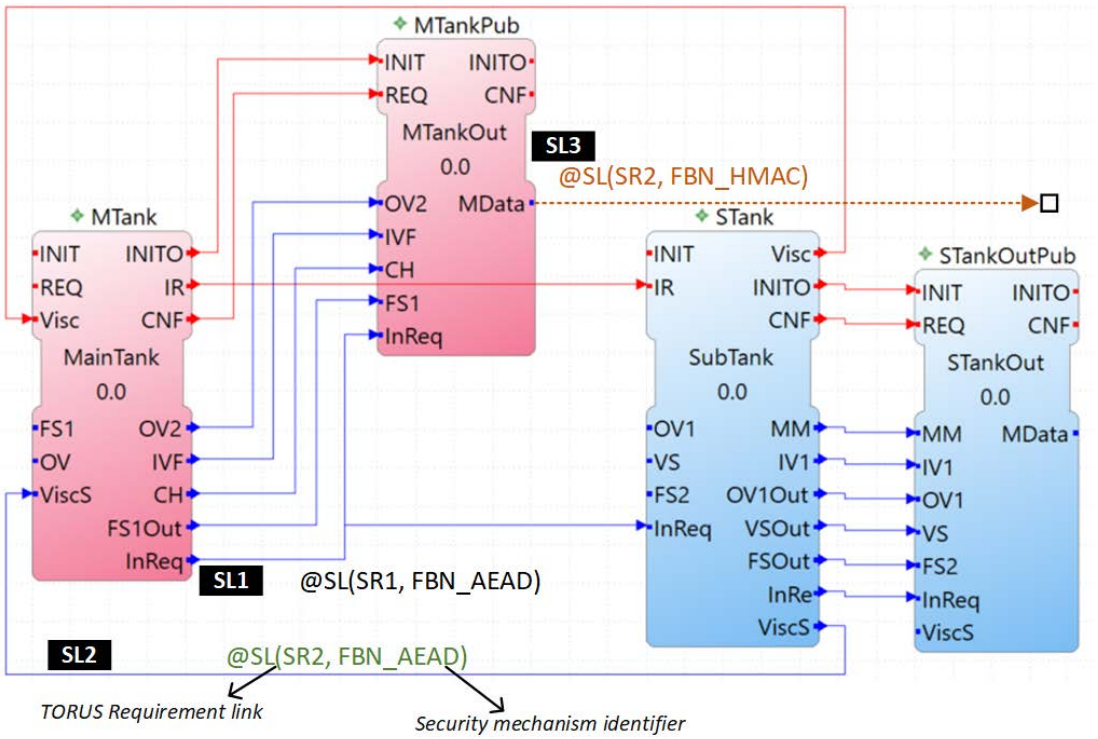

Figure 5.4: IEC 61499 implementation of IMCS including secure links. The blocks are coloured *wrt* to device mapping.

Figure 5.5 shows an example of a security mechanism implementing lightweight secure hashing. The FB instances of the mechanism are partitioned into two sets FB_a and FB_b, each containing three blocks, indicated by the colouring of the blocks. As both FB_a and FB_b are non-empty, this is a `send-receive` mechanism, which requires deployment into two separate devices, responsible for sending and receiving the communications. This partitioning also makes security mechanisms distinctly different from *composite* function blocks in IEC 61499, which contain networks that must be deployed on the *same* device. This mechanism contains a set of parameters $params$ = `{datalen,QI,ID}` that can be configured manually or be assigned default values.

Key exchange is an integral part of many security mechanisms, but it becomes increasingly complicated for larger sets of devices. In our case, where needed, each security mechanism includes the required logic for key exchange, in the form of additional FB instances, providing a dedicated (up to) two-device key exchange.

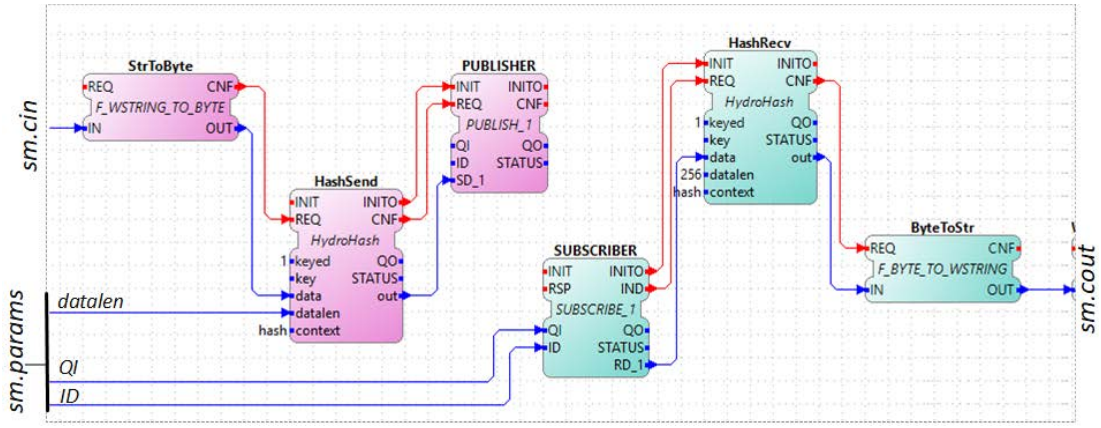

Figure 5.5: A send-receive security mechanism for lightweight hashing.

Definition 3. Secure Links For a FB network $fbn = (FB, C)$, $SL \subset C$ is a set of secure data links such that for each $sl \in SL$ contains the following *annotations*:

- $sl.r \in RR$ is a requirement from a repository RR.

- $sl.sm$ is the reference to a *security mechanism* and $sl.\bar{pv}$ are parameter assignments needed to instantiate $sl.sm$.

The use of secure links forms the second part of the SLDM process described in Figure 5.2. A possible approach, such as, with TORUS, may involve using a graph database to store the requirement repository RR and the security library $SLib$ that allows designers to interactively query these resources to identify security requirements and then choose a suitable security mechanism to address each requirement. A secure link simply annotates a variable connection of a FB network with two annotations: a requirement from the CSRS stored in the repository RR and a configured security mechanism from the secure library $SLib$. Figure 5.4 illustrates the use of secure links in the IMCS application. Consider the data connection/secure link sl = `(MTank.InReq,STank.InReq)`. Its annotations include $sl.r = SR1$ which is one of the security requirements in the TORUS repository, discussed in Section 5.5.1. Additionally, the second annotation $sl.sm = FBN\_AEAD$ mentions the security mechanism chosen by the designer from the *security library*. The assignment of parameters $sl.\bar{pv}$ allow the designer to configure the mechanism appropriately. Note that configured parameters are not illustrated in Figure 5.4 for readability.

SLDM depends on the availability of a comprehensive security library, which does require initial effort to develop. Moreover, the inclusion of newer security mechanisms, such as for active security protection, requires this library to be frequently updated and maintained.

5.6 Compiling Secure Links into IEC 61499-compliant Software

Applications containing secure links can be transformed into IEC 61499-compliant applications that can then be deployed to any PLC. Alg. 1 shows the steps involved in this transformation. Overall, the algorithm replaces each secure link by a configured instance of the referenced security mechanism (lines 2–22), *only* if the two ends of the link connect FB instances deployed onto different devices (line 3). The chosen mechanism is first retrieved from the security library (lines 4–7) and is added to the application in lieu of the original secure link (lines 7–8). The device partitions FB_a and FB_b of the mechanism are mapped to be deployed onto the separate devices hosting the communicating blocks in the input network (lines 10 and 15). The data connection in the secure link is replaced by adding in the mechanism via connections cin and $cout$ of the network (lines 11 and 16), with a similar adjustment in associated events (lines 12 and 17). The parameters for the mechanism are adjusted according to the values provided by the designer (line 19) or by fetching the stored default values for the parameters. In line 20, the compiler signals to TORUS that the secure link has been replaced by a configure network for the security mechanism, for traceability purposes.

Figure 5.6 shows the fully IEC 61499-compliant network obtained via the compilation of the application shown in Figure 5.4. The FB network added for every security mechanism is illustrated by a bounding box annotated with the text of the corresponding secure link. $SL1(FB_a)$ and $SL1(FB_b)$ are the send and receive parts of the security mechanism, respectively providing TLS based AEAD security. $SL2(FB_a)$ and $SL2(FB_b)$ are the send and receive parts of the security mechanism respectively providing lightweight AEAD security. $SL3(FB_a)$ denotes only the sending part of a lightweight hashing security mechanism. Key exchange server and the client are the

Algorithm 1 Secure Link Compilation

input: Input network fbn_{in}, Library $SLib$
output: Transformed network fbn_{out}

Initialize fbn_{out} = fbn_{in}
for all $sl \in fbn_{in}.SL$ **do**
  if $(\mathcal{D}(fb(sl.src)) \neq \mathcal{D}(fb(sl.trg))$ **then**
    Retrieve $sl.sm$ from $SLib$
    Instantiate $sl.sm$ to network fbn_s
    Name each instance fb in fbn_s as fb_sl
    Add fbn_s to fbn_{out}
    Remove sl from fbn_{out}
    if $sl.src \neq \square$ **then** (`receive` mechanism)
      map $fbn_s.FB_a$ to $\mathcal{D}(fb(sl.src))$
      Set $fbn_s.\mathcal{S}.cin.src$ to $sl.src$
      Replicate all associations of $fbn_{in}.\mathcal{A}(sl.src)$ in $fbn_s.\mathcal{A}(fbn_s.\mathcal{S}.cin.src)$
    end if
    if $sl.trg \neq \square$ **then** (`send` mechanism)
      map $fbn_s.FB_b$ to $\mathcal{D}(fb(sl.trg))$
      Set $fbn_s.\mathcal{S}.cout.trg$ to $sl.trg$
      Replicate all associations of $fbn_{in}.\mathcal{A}(sl.trg)$ in $fbn_s.\mathcal{A}(fbn_s.\mathcal{S}.cout.trg)$
    end if
    assign $fbn_s.\mathcal{S}.params$ = $sl.\bar{pv}$
    Signal to TORUS that fbn_s replaces sl
  end if
end for
return fbn_{out}

send and receive parts of a key exchange protocol provides key generation for both $SL2$ and $SL3$. Generating common key exchanges for secure links distributed over the same pairs of devices is a simple compiler optimization for faster performance. $SL1$ consumes keys generated during TLS handshakes, and so it does not require dedicated key exchange.

The compiler generates additional splices for TORUS to link the generated code directly to security standards. Figure 5.7 shows how TORUS uses explicit references to the requirements contained within secure links to automatically create splices between requirements and secure links, shown as linkages between the clusters “CSRS" and “Secure Links" in Figure 5.7. During compilation, additional splices between a secure link and the function block instances added to the network are created (line 20 of Alg. 1). More persistent linkages are created by naming the instances appropriately in line 6 of Alg. 1. As Figure 5.7 shows, during compliance checks, component requirements *CR4.1* and *CR4.2* from IEC 62443 standard can be automatically and directly linked directly to specific function block instances that address them.

Currently, we have developed and tested a Java implementation of the algorithm in an IEC 61499 compliant Eclipse-based 4diac IDE plugin.

5.7 Experimental Results

Intuitively, using secure links allows for an easier way to construct complex ICS applications with secure-by-design communications. In this section, we quantify the impact of secure links on critical measures like *design complexity* and *maintainability*, which measure the effort required to build and refine secure ICS applications. We perform this analysis over two case studies: the IMCS case study and a large Baggage Handling System (BHS) application (Black & Vyatkin, 2009).

Currently, the SLDM design process and compilation are implemented as a single

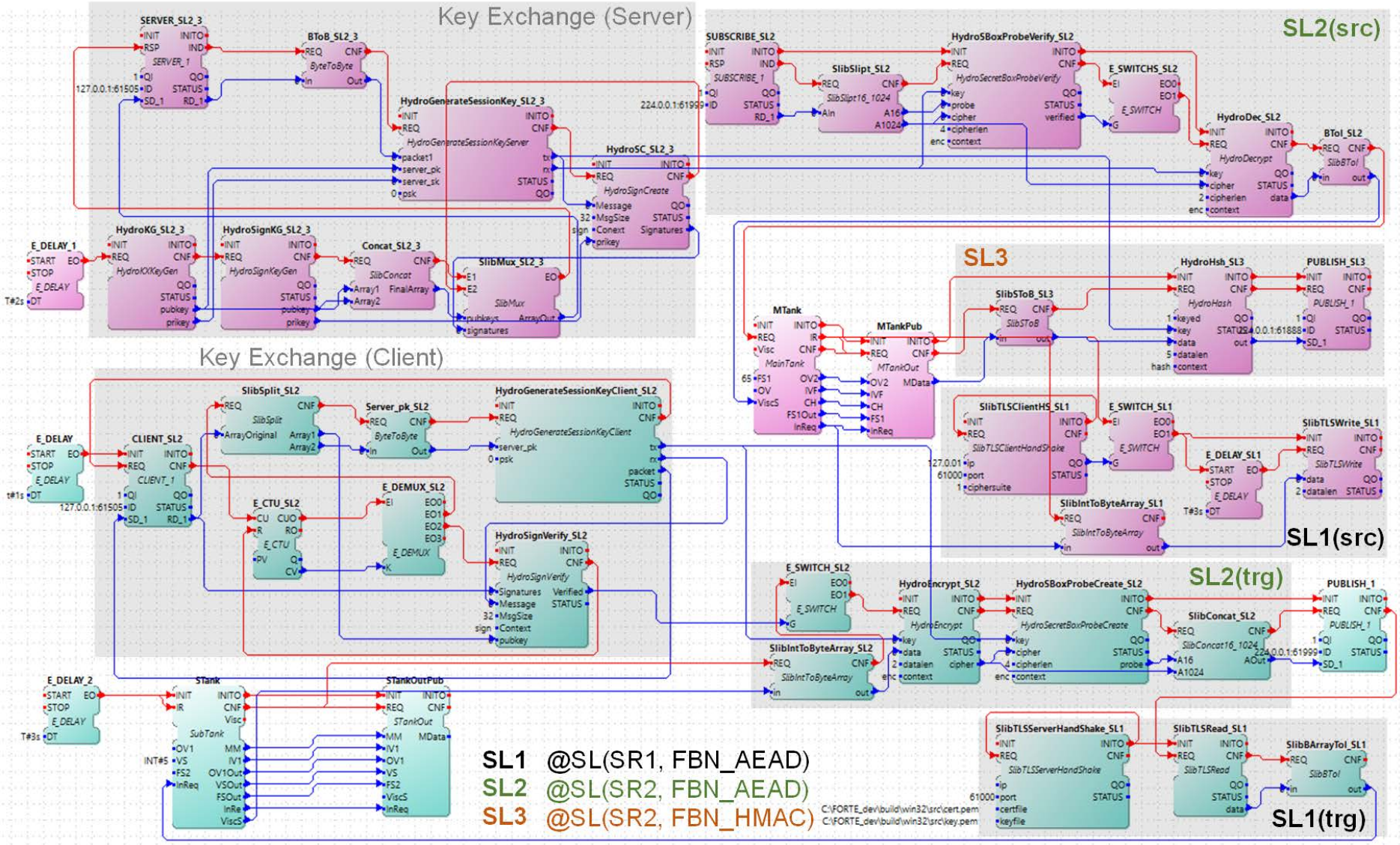

Figure 5.6: Fully IEC 61499-compliant IMCS application generated by Alg. 1

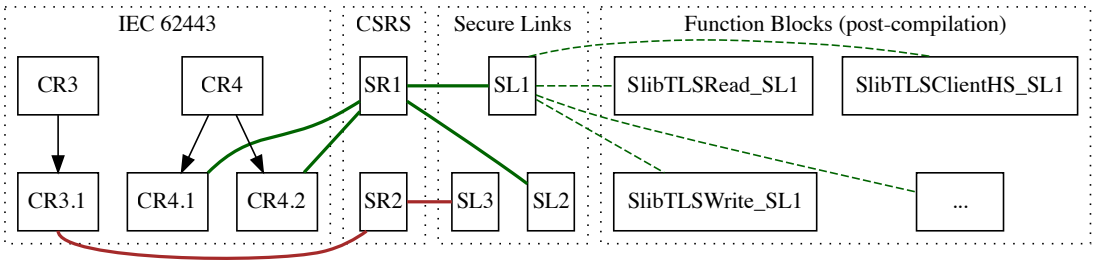

Figure 5.7: Tracing requirements from security standards to final implementation using TORUS

Eclipse plugin for the 4diac IDE. Designers can add secure link annotations over data connections and then compile designs into fully IEC 61499 and 4diac FORTE compliant applications. The security library is implemented as a resource library for 4diac FORTE, and each security mechanism in the library is implemented using standard IEC 61499 features and is stored using the 4diac FORTE compatible XML format. SLDM is independent of the IDE used and can be extended to other IDEs in the future. The SLDM plugin also provides an interface to TORUS for end-to-end traceability between requirements from standards and final implementations. Currently, some manual effort is required to invoke TORUS when an artefact (requirement, application design, secure link or security mechanism) changes. A fully automated interface with TORUS is another important future implementation task for SLDM. All final implementations used in our experiments execute using the 4diac FORTE runtime environment and are distributed over Wago PFC200 PLCs. For the IMCS application shown in Figure 5.4, the FB instances `MTank` and `MTankPub` execute on $PLC1$, and `STank` and `STankOutPub` execute on $PLC2$[2].

Measuring the complexity of IEC 61499 applications requires adapted measures that provide more accurate results than generic measures like program length. Specifically, we calculate the following three measures, adapted for IEC 61499 and presented in (Zhabelova & Vyatkin, 2015), to characterize the complexity of a function block i:

[2][Online]. Available: https://github.com/emsoftaut/securelinks-iec61499

- Structural complexity $S(i) = f_{out}^2(i)$ where f_{out} is the number of outgoing connections or afferent coupling.
- Data complexity $DC(i) = [NI + NO]/[f_{out}^2(i) + 1]$ where NI and NO are the numbers of data inputs and outputs, respectively.
- System Complexity $C(i) = S(i) + DC(i)$.

Table 5.1 shows the complexity values for the IMCS and BHS applications (without and with secure links) as well as for each of the security mechanisms we have implemented into the security library. Table 5.1 contains the aggregated values of S, DC, and C for each network, represented as S^*, DC^*, and C^* respectively and defined as the sums of all $S(i)$, $DC(i)$, and $C(i)$ for each FB instance in the network.

Table 5.1: Complexities for applications and security mechanisms

Function block Network	f_{out}	**NI**	**NO**	S^*	DC^*	C^*
IMCS	4	29	23	8	29.6	37.6
IMCS + secure links	4	41	23	8	41.6	49.6
BHS	48	133	82	196	37	233
BHS + secure links	48	217	82	196	121	317
Design Complexities for security mechanisms						
sm (Key Exchange (L))	29	54	64	69	29.8	98.8
sm (Key Exchange (TLS))	4	11	8	8	3.8	11.8
sm (AEAD (L))	17	46	43	27	30.1	57.1
sm (AEAD (TLS))	7	14	17	9	14.3	23.3
sm (Hash/HMAC)	6	27	23	6	25	31

Figure 5.8 shows the cumulative increase in pre-compilation complexity when security mechanisms are successively added using secure links or through manual coding. The bars labelled “IMCS” represent the base complexity of the system. “Secure Links” represents IMCS system complexity when secure links are added for three mechanisms: AEAD(TLS), AEAD(L) and Hash. The groups “+AEAD(TLS)”, “+AEAD(L)”,

and “+Hash” show the complexities when the corresponding mechanisms are added successively to the base design represented by the “IMCS" group.

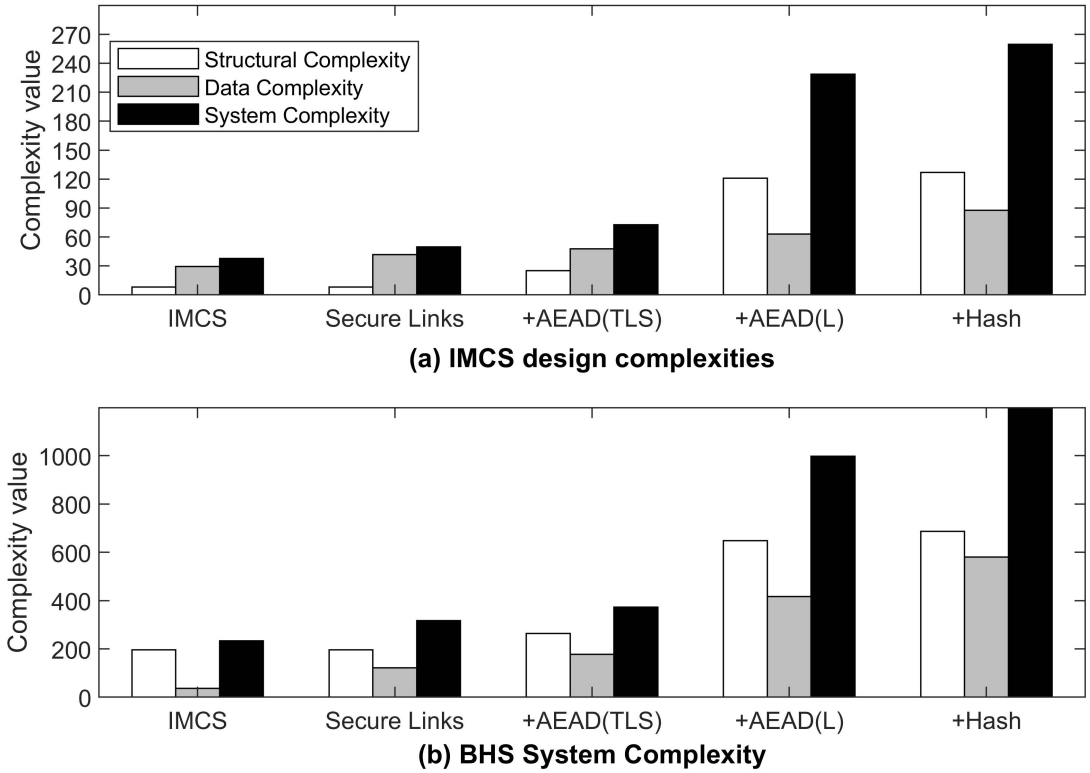

Figure 5.8: Cumulative effect on the design complexity of IMCS and BHS after secure links are processed by algorithm 1.

Using secure links has a negligible effect on application complexity. For IMCS, system complexity rises to 49.6 from 37.6. For BHS, system complexity increases to 317 from 233. On the other hand, when all three mechanisms are added manually, the system complexity S^* for the BHS and IMCS rises from 37.6 to $259.6.6$ and 233 to 1198.6, respectively. Note that the complexity, shown by the “+AEAD(TLS)”, “+AEAD(L)”, and “+Hash” groups is the same as the *post-compilation* complexity of the fully IEC 61499 compliant code generated by the SLDM compiler from the secure links designs represented by the “Secure links" group.

Figure 5.8 shows a rise in complexity when (more) security mechanisms are added manually. Mechanisms coded into an application can be either lightweight or

traditional. Traditional methods like TLS need support through established protocols such as openSSL which exist outside of the application. However, this is a blanket approach and does not provide flexibility as *all* inter-device communications are secured instead of only the ones marked secure by the designer. It sacrifices performance, especially in ICS where applications execute on resource-constrained PLCs. In contrast, lightweight methods provide the flexibility of securing a subset of inter-device communications which results in better performance. However, as Figure 5.8 shows, coding the lightweight methods like AEAD(L) adds more complexity to an application than traditional methods because of the larger number of FBs are required to implement such mechanisms completely (and without external infrastructure). The lightweight Hash algorithm is an exception because it causes a relatively smaller increase in complexity. It is because hash functions are inherently simpler to implement, and only one (sending) part of the mechanism is added to each network in our experiments.

Table 5.2: Pre-compilation program complexity and maintainability

	Halstead's Metric (M_H)							**McCabe's Metric**			**Maintainability**
Name	N	n	$\hat{N}$	PR	V	D	E	$V(alg)$	$V(cf)$	M_M	MI
IMCS (overall)	44	35	110.09	5.20	183.94	11.50	1207.80	4	4	8	93.51
IMCS+secure links	62	53	124.35	8.77	213.94	14.50	1222.80	7	6	13	89.31
BHS (overall)	1632	272	1605.57	8.17	12101	241.79	2583135	60	11	71	42.19
BHS+secure links	1650	290	1619.83	11.74	12131	244.79	2583150	63	14	77	42.07
Program complexity and maintainability index for cryptographic function blocks of security mechanisms											
HydroKxKeyGen	51	33	106.06	4.76	217.43	8.46	1351.38	7	3	10	82.66
HydroSignKeyGen	47	33	107.01	4.99	199.59	6.68	930.90	6	3	9	85.76
HydroGenerateSessionKeyServer	70	40	148.31	4.87	326.07	6.27	1490.13	8	3	11	77.07
HydroSignCreate	61	40	144.99	5.15	282.35	7.41	1555.54	8	3	11	78.58
HydroGenerateSessionKeyClient	62	41	150.86	5.22	289.75	7.13	1529.96	8	3	11	78.44
HydroSignVerify	70	46	178.07	5.36	342.35	8.71	2322.81	8	3	11	76.10
HydroEncrypt	97	48	186.10	4.68	487.82	16.83	7080.21	11	3	14	71.58
HydroDecrypt	113	52	209.57	4.63	586.42	19.20	9914.63	13	3	16	68.68
HydroSecretBoxProbeCreate	77	43	160.08	4.84	369.59	11.09	3319.27	7	3	10	76.65
HydroSecretBoxProbeVerify	89	45	170.85	4.67	436.50	12.79	4658.63	7	3	10	74.72
HydroHash	121	59	251.02	4.88	654.53	21.11	12320.04	10	3	13	64.35
SlibTLSServerHandShake	123	57	246.33	4.80	658.76	12.31	6911.19	11	3	14	65.91
SlibTLSClientHandShake	120	56	238.24	4.78	638.70	13.86	7624.16	9	3	12	66.53
SlibTLSRead	54	38	132.34	5.23	243.81	8.06	1457.61	4	3	7	83.68
SlibTLSWrite	55	40	143.72	5.43	253.20	7.39	1373.46	4	3	7	84.47

Program-level complexity and maintainability can be measured by IEC 61499 relevant measures proposed in (Zhabelova & Vyatkin, 2015). The Halstead's metric for static source complexity involves computing measures $N, n, \hat{N}, PR, V, D$ and E corresponding to the number of operators and operands, program vocabulary, estimated length,

purity ratio, program volume, program difficulty, and program effort, respectively. Due to space limitations, we include the formula used to compute program difficulty D, and refer to reader to (Zhabelova & Vyatkin, 2015) for details on how other measures are computed. D is computed as

$$D(fbn) = \frac{\sum_{i=1}^{n} D(fb_i)}{n}; D(fb) = \sum_{i=1}^{n} D(alg) + D(cf)$$

$D(fbn)$ and $D(fb)$ are the program difficulty of an application and a constituent function block, respectively. Similarly, McCabe's cyclomatic complexity metric M_M is calculated as:

$$M_M(fbn) = \frac{\sum_{i=1}^{n} M_M(fb_i)}{n}; M_M(fb) = \sum_{i=1}^{n} V(alg) + V(cf)$$

$M_M(fbn)$ and $M_M(fb)$ represent the cyclomatic complexity of an application and a constituent function block, respectively. $M_M(fb)$ calculations use the value V which is the sum of the cyclomatic complexity of algorithms $V(alg)$ and the cyclomatic algorithms of the control flow $V(cf)$ in a function block. $M_M(fbn)$ is simply the average of all $M_M(fb)$ values in the network.

Table 5.2 shows M_H and M_M for the IMCS and BHS applications (without and with secure links) and the security mechanisms we have implemented. To calculate M_H and M_M for applications containing secure links, we consider each link as an operator and its parameters as operands. For example, SL1 secure link annotation is considered as an operator, and the parameters *SR1* and *FBN_AEAD* are considered as operands. The M_H and M_M values are integrated into a single value called the Maintainability Index (MI)[3]. A higher MI value indicates a more maintainable program.

Program difficulty D is a vital metric that has a direct influence on the overall implementation effort of a module. Figure 5.9, shows that final cumulative program difficulty and cyclomatic complexity V increase sharply as security mechanisms comprising of

[3]Further details of all calculations can be found in (Zhabelova & Vyatkin, 2015).

individual FBs from Table 5.2 are manually added to the IMCS and BHS applications. When three security mechanisms are added, D rises from 11.5 to 178.7 for the IMCS and from 241.7 to 409 for the BHS. Other measures in M_H and M_M show a similar trend of sharp increases for manual coding.

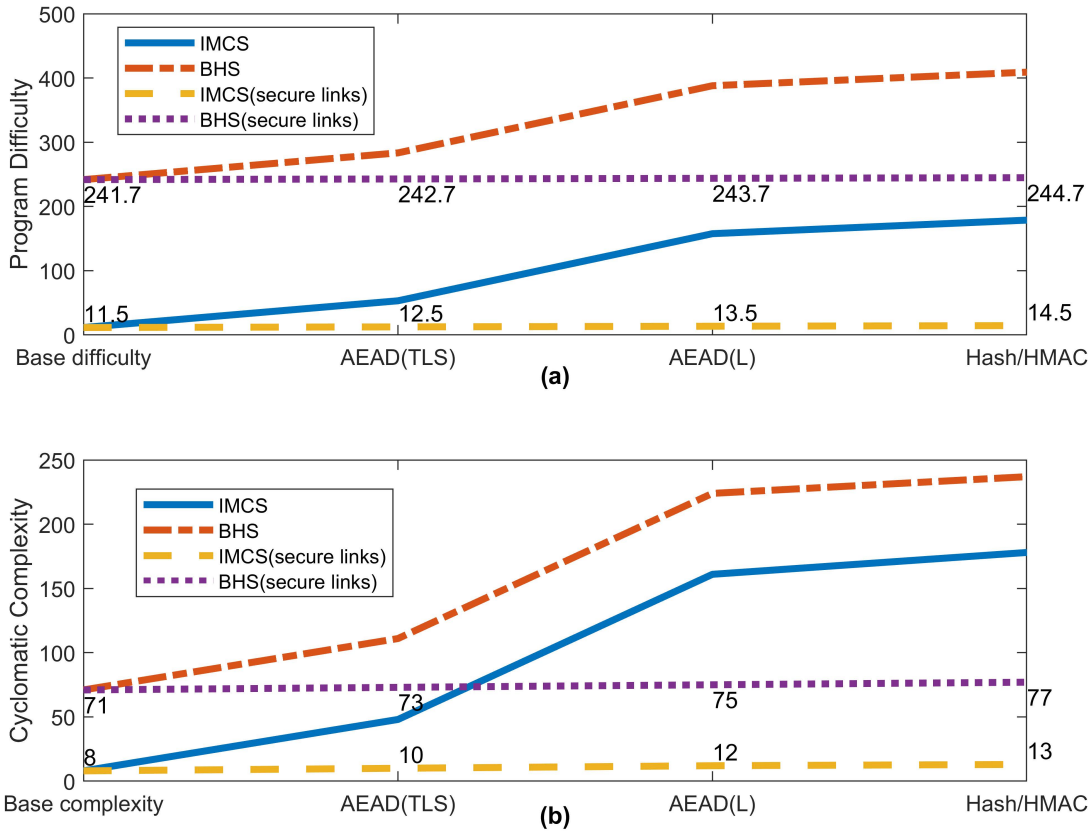

Figure 5.9: Manual addition of security mechanisms vs the use of secure links: cumulative program difficulty (D) and cyclomatic complexity (V) comparisons.

Figure 5.10 shows a significant decline in maintainability when security mechanisms are added manually. On the other hand, using secure links results in a minimal drop in maintainability. Adding three secure links to the IMCS results in only a slight drop in maintainability from 93.5 to 89.3, which is also shown in Table 5.2. The BHS application was tested for multiple security levels provided by IEC 62443-4-2 (CR 3.4). Security level L1 can be achieved by providing integrity, while L2 requires the implementation of integrity and authentication mechanisms such as HMAC. L4 requires

encryption for some component requirements. Overall, eight different secure links corresponding to varying security levels were added to the BHS. Figure 5.10(b) shows that MI decreases from 42.19 to below 20 for L4 when mechanisms are added manually. With secure links, MI sees only a minor drop from 42.19 to 41.9.

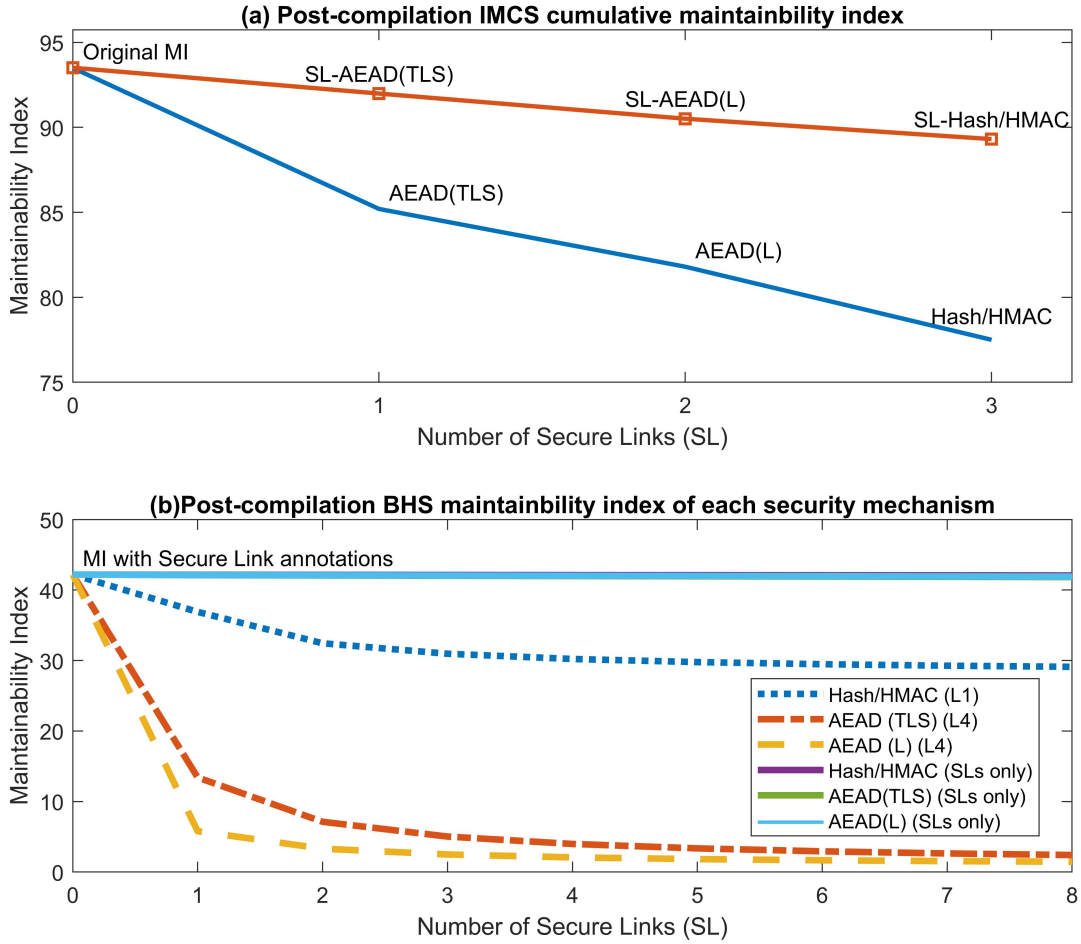

Figure 5.10: Maintainability of IMCS & BHS with and without secure links.

As discussed previously, and illustrated in Figure 5.8, SLDM supports both lightweight and traditional security mechanisms. Table 5.3 shows the average latency of a hundred transactions between the two Wago PFC200 PLCs while using different security mechanisms in the IMCS application. The clocks on both PLCs were synchronized using the network time protocol. We use `libhydrogen` for lightweight mechanisms and `openssl` to support TLS. Both libraries use fast `Curve25519`

elliptic curve-based key exchange with `libhydrogen` employing lightweight `Gimli` permutation operations that further suit resource-constrained environments. For all transactions, the lightweight approach takes significantly lesser time than the traditional TLS approach. TLS has a 32% overhead for key exchange over the lightweight method. For Authenticated Encryption with Associated Data (AEAD), `libhydrogen`'s secret box API consumes 3.34 ms as compared to TLS using `AES128-GCM-SHA384`, that has a higher latency of 4.90 ms. Achieving security level L3 in the IMCS requires authentication and integrity with no encryption. In this case, generic hashing with the `libhydrogen`'s key API has an overhead of 1.87 ms, compared to 3.03 ms consumed by TLS (`NULL-SHA256`).

Table 5.3: Latency comparison between lightweight and TLS security mechanisms implementations.

	Lightweight	**TLS**
Key Exchange	373.27 ms	492.65 ms
AEAD	3.34 ms	4.90 ms (AES256, SHA384) 4.80 ms (AES128, SHA256)
CBC Mode		4.03 ms (AES128, SHA256)
Integrity + Authentication	1.87 ms	3.03 ms

Overall, experimental results show that secure links and the automatic compilation process assist in building applications with low complexity and high maintainability. Although the results are specific to the implementations of TLS and lightweight mechanisms that we have implemented, the use of secure links is independent of the implementations of security mechanisms. SLDM allows adding new mechanisms or optimizing currently included mechanisms at any time without affecting secure links based application designs. Our results show that traditional approaches like TLS result in slightly lesser complexity increases than lightweight approaches but have much higher performance overheads and lack the flexibility required in choosing a specific subset of inter-device communications to secure. Manually adding mechanisms, both traditional and/or lightweight, causes a significant drop in application maintainability

and a corresponding rise in complexity, which can be avoided by using secure links. Even when a security mechanism is not available in the security library, it is more beneficial to implement it first as a security library element which can then be used flexibly using secure links. Secure links-based applications are easier to refactor when security requirements or target security levels change. Additionally, the integration of TORUS into SLDM allows end-to-end traceability between standards and final implementations (as shown in Figure 5.7). Finally, secure links are highly reusable design abstractions. The BHS case study implementation contains multiple composite FBs where each block controls a conveyor belt. All such conveyor belt controllers must satisfy the same set of communication security requirements. Hence, a secure link used to secure one such controller can simply be reused for all others.

5.8 Conclusions

Secure-by-design approaches can significantly reduce the effort required to build certifiably secure ICS applications. We propose an ICS development method called the Secure Links Development Method (SLDM) encapsulating the concept of secure links that are design abstractions supporting the development of secure-by-design IEC 61449 distributed applications. Secure links provide clear traceability between security requirements from ISA/IEC 62443 and application code, which helps reduce the efforts required for certification. Necessary design automation, proposed as a fully IEC 61499-compliant compilation of secure links based applications, ensures that very intricate designs can be built with a negligible impact on design-time complexity and application maintainability. Future directions for this work include the development of mature tools that can be integrated into mainstream ICS development workflows and tools, as well as extending the work to include additional ICS security and development standards.

Chapter 6

Security Requirements Repository

6.1 Prelude

The following chapter has been submitted as a journal article in ***Springer Journal of Software and Systems Modeling*** under the title of ***Tracing Security Requirements in Industrial Control Systems using Graph Databases***.

This work proposes a novel model of a security repository. The repository stores labelled property graphs for cybersecurity requirements specifications and IEC 62433-4-2 security requirements in multiple partitions while emphasising requirements structure and relationships. The repository aims to add to the work done in previous chapters by providing a security requirements engineering mechanism that can be consumed by the secure links and the TORUS to provide end-to-end traceability of security requirements in ICS applications. This work also forms a part of integrated techniques discussed in Section 7.2 that provide an approach to develop certified ICS applications while remaining compliant to traditional and ICS-specific development methodologies and frameworks such IEC 62443-4-1 and the traditional software development life-cycle.

6.2 Abstract

Requirements traceability from security standards requirements specification is critical for security certified Industrial Control Systems (ICS). Integration of system and security standards requirements grows into large requirements set where the requirements relationships and hierarchies must be explicit. Current security requirements specification methods lack focus on the requirements structure, making requirements management and tractability difficult to achieve specifically for certified ICS. We propose a novel requirements repository model for ICS that uses labelled property graphs to structure and store the security standard and system-specific requirements using well-defined relationships. Furthermore, we integrate the repository with the design-time ICS tools to establish requirements traceability. A wind turbine case study is used to illustrate the repository utilisation in ICS. Moreover, we demonstrate the construction of the requirements traceability matrix that emerges as a natural consequence of using a labelled property graph repository. Finally, compatible requirements change management procedure has been illustrated that aids in adjusting against the changes in ICS application development and certification schemes.

6.3 Introduction

Industrial Control Systems (ICS) deployed in critical infrastructures put high reliance on security and safety features. Therefore, security requirements specification and management is a crucial part of the overall requirement engineering process of ICS.

Critical components deployed in ICS must comply with security standards. ICS-specific security standards such as IEC 62443 (Kronfuss, 2018) provide robust yet generic sets of security requirements. Such requirements sometimes can be intricate to understand in terms of their applicability for a specific project. The certification process

around security standards also requires multiple parties to concur on a set of security requirements specific to a particular product. Therefore, stakeholders must agree on a security requirements engineering approach in a security certification process that is feasible to all the parties involved, such as the user, vendor and the certificate authority. Correct mapping of system requirements to security standards requirements is critical to ensure the security of the whole process (Rosenstatter & Olovsson, 2018).

Current requirements elicitation and specification approaches are not rigorous enough to scale to developing certified large-scale ICS that require continuous progression analogous to technology advancement. Security requirements specification techniques such as SIREN (Toval et al., 2002; Mellado et al., 2007) focus on the reuse of requirements with the help of repositories. However, they use textual and semi-formal approaches that are laborious to express and comprehend. Security standards are also written in a natural language containing requirements specification that is abstract and often ambiguous (Fenz, Plieschnegger & Hobel, 2016; Giannakopoulou, Pressburger, Mavridou & Schumann, 2020; Bruel et al., 2021). For example, a requirement such as *"The ICS shall ensure the security of its critical parameters through the use of cryptography."*, is abstract and vague since it does not expand the terms such as "critical parameters" and "cryptography'.

Security standard requirements must be mapped to product-specific system security requirements usually contained in a Cyber Security Requirements Specification (CSRS) (Gunter et al., 2018) document. This document acts as a stepping stone to cater for secure-by-design practices by detailing the security requirements and the capability levels across the ICS zones. The interpretation of the standard against system security requirements also presents a unique challenge regarding the correct mapping of requirements (Mussmann, Brunner & Breu, 2020). The current ICS security requirements engineering practices do not offer formal and expressive specification techniques to map the system's security requirements to ICS specific security standards. An expressive

formal requirement provides the ability to convey the detailed meaning of a security requirement and remove any ambiguity that is inherent in the natural language.

Furthermore, (Beckers, 2015) discusses the extension of security requirements engineering methods to the ISO 27001 (Calder, 2013) security standard. However, ISO 27001 is a general information technology standard, not specific to ICS. A recent research (Constante et al., 2021) discusses the integration of security standards to ICS development life-cycle; however, it lacks the internal details about security requirements mapping with security standards. Moreover, (Ehrlich et al., 2020; Bicaku et al., 2021) discuss the security standard requirements compliance, yet generally focus on security requirements verification of ICS end products. Lack of system to security standard requirements mappings reduces the traceability of security requirements during the development of security certified ICS components.

We addressed the problems stated above by exploring the following research questions:

RQ1: What techniques can manage and express the relationship between system and security standard requirements?

RQ2: How can security standard requirements be integrated with the design and implementation of ICS applications to achieve end-to-end traceability?

We followed an adapted case study method to address these questions. A safety-critical wind turbine system was modelled and implemented to study the various issues in managing and tracing security requirements in ICS. An overview of this solution appears in Section 6.4.

The primary contributions of this article are as follows:

1. We propose a novel model of a repository that stores LPGs for CSRS and IEC 62433-4-2 security requirements in multiple partitions while emphasising requirements structure and relationships.

2. We present the formal definition of the IEC 62443-4-2 extended requirements structure that helps select standard cryptographic primitives to guide the implementation of IEC 62443-4-2 requirements.

3. We propose and demonstrate a process to integrate the repository with ICS design tools to support end-to-end requirements traceability.

The rest of this article is organised as follows. Section 6.5 presents the background of techniques and tools used in the solution while Section 6.6 discusses the related literature review and current works. Section 6.7 provides a formal definition of extended IEC 62443-4-2 requirements structure that relates to the contribution 2 listed above. Section 6.8 presents the method to generate IEC 62443-4-2 and CSRS LPGs. Section 6.9 demonstrates the repository architecture and implementation that relates to the contribution 1. Section 6.10 demonstrates design-time ICS tool integration for requirements traceability purposes. It relates to the contribution 3 of this article. Section 6.11 presents the results and discussions while Section 6.12 discusses some of the limitations and validity threats of our proposed solution. Section 6.13 finally concludes the article and proposes future works.

6.4 Overview of the Proposed Solution

Figure 6.1 shows an overall process that integrates the proposed repository model to the design-time tools for ICS. Such integration allows the security requirements in the repository to be linked with the design and the implementation enabling end-to-end requirements traceability.

A safety-critical wind turbine ICS case study is adopted to demonstrate the repository usage and its associated traceability process. Multiple Programmable Logic Controllers (PLCs) are installed in a master-slave topology in the wind turbine system. A master

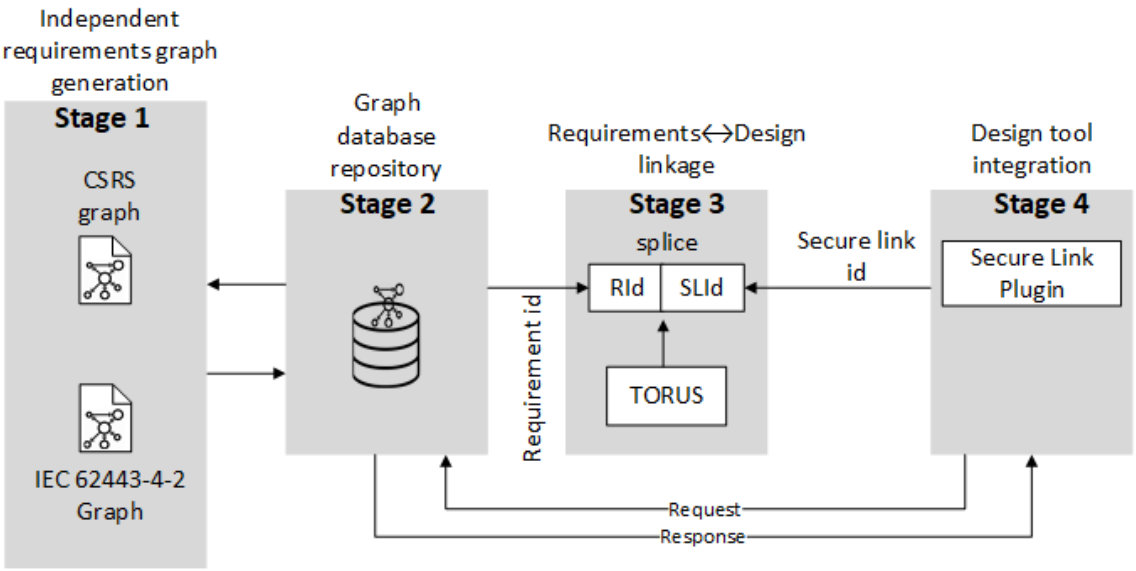

Figure 6.1: Process overview of creating an LPG security requirements repository and its association with secure-by-design and traceability tools. RId and SLId in stage 3 refer to requirement and secure link id respectively, while TORUS is a requirement traceability tool.

PLC is usually placed at the wind turbine base that commands the slave PLCs in the nacelle. Moreover, the data from PLCs is transmitted to the control system and subsequently to Supervisory Control and Data Acquisition (SCADA) or enterprise systems. The communication between master and slave PLCs is critical because it essentially controls the physical processes of the wind turbine. The control device network can be compromised if an attacker can install a rogue device between master and slave PLCs to sabotage the operation of the components in the nacelle and the pitch gears. Such an ICS demands trustworthy communication security between all the components of an ICS. Table 6.1 shows the CSRS extract of system security requirements of a wind turbine system derived for PLCs arranged in master-slave topology. Column 1 of Table 6.1 shows the respective security goal to be achieved for the system. *RID* in column 2 represents the requirement and sub-requirement identifier in the CSRS while "security level" in column 4 represents the intended security level respective to IEC 62443-4-2. For example, *CR* is overall requirement for confidentiality

Table 6.1: Wind turbine security requirements

Goal	RID	Example of Security Requirement (SR) of Wind turbine PLCs	Security level
Confidentiality	CR	The master and slave PLCs shall ensure the confidentiality of the data in transmission and at rest.	
	CRa	Data communication between master and slave PLCs shall use appropriate encryption algorithms.	SL-C 2
	CRb	Critical parameters shall be not be persisted on the master and slave PLCS in order to ensure the confidentiality of data for discharged devices from the system.	SL-C 4
Authentication	AR	Any access to the PLC (Master/Slave) shall be provided after appropriate authentication based on role-based identification.	SL-C 1
Integrity	IR	The system shall ensure the integrity of ingress and outguess data.	
	IRa	Communication between master PLC and external components shall use appropriate methods to ensure the integrity of the data.	SL-C 4
	IRb	Communication between master and slave PLCs shall support communication integrity checks.	SL-C 4

nevertheless broken down into *CRa* and *CRb* portraying the precise requirements for communication and critical parameter storage. An analyst can choose a particular security level for each sub-requirement based on the system constraints.

The first stage, *requirements graph generation*, involves creating LPGs of CSRS for the wind turbine system and the security standard. The wind turbine security requirements listed in Table 6.1 form the CSRS. An LPG is thus created based on these requirements where each requirement/sub-requirement such as *CRa*, *CRb*, *AR*, *IRa*, and *IRb*, is considered as a graph node. Such an approach of structuring the security requirements from multiple specification documents helps highlight the relationships for highly connected hierarchical requirements. The graphs are created independently of each other such that each system CSRS graph may have a unique graph while the security standard graph remains constant to be used with multiple CSRS graphs. While each CSRS graph may have a unique requirements structure, IEC 62443-4-2 has a settled requirements structure described formally in this article.

The second stage, *graph database repository*, creates the repository by storing the LPGs created in the previous step in a graph database tool. An LPG provides a structure that is useful in efficient storing and managing the repository. A key advantage of using LPG is that the relationships between data are computed and stored at the database creation stage (C. Sharma & Sinha, 2019). The repository enables querying of complete

requirement trees by specifying graph patterns and filtering the result sets based on requirement identifiers and properties (C. Sharma, Sinha & Leitao, 2019; C. Sharma, 2020). This article uses this querying ability to integrate CSRS and security standard graphs to get a holistic view of a security requirement's hierarchy and dependent requirements. We also discuss that multiple partitions of the proposed repository are highly reusable for vendors and the ICS development community.

The third stage of *Requirements↔Design linkage* uses TORUS (Sinha et al., 2018) for requirements traceability. TORUS is a requirement traceability tool that uses *splices* to link the requirements to their implementation. TORUS's splice metadata maintains the repository link using a requirement identifier (e.g. *IRa* from Table 6.1) from the CSRS of the wind turbine system. The use of TORUS with the repository enables it to act as a bridge to provide traceability between the requirements in the security repository and the application design.

The final stage of *design tool integration* uses secure links – a secure-by-design tool – for ICS (Tanveer et al., 2020) that briefly introduces the idea of integrating secure links and TORUS with a requirements repository. However, we consolidate the concept by demonstrating its practical use with the LPG repository proposed in this article. It is achieved by storing the secure link and the wind turbine CSRS requirements identifiers (such as listed under *RID* column of Table 6.1) in a TORUS splice enabling end-to-end requirements traceability. The application of the proposed repository model shows that it produces a transparent and detailed traceability matrix that is beneficial in the verification and validation of security requirements.

A detailed process for the requirements change management illustrated in Section 6.11 aids in the requirements addition, removal and modification from the repository. The results also show an improvement in requirements extraction effort from the security standards. We also show that using graph-based repository aids in the efficient requirements traversal and analytics using Cypher (Francis et al., 2018) language.

6.5 Background

IEC 62443-4-2 Technical requirements for ICS components standard is the focus of this research. This standard describes Component Requirements (CRs) derived from Foundational Requirements (FRs) defined in IEC 62443-1-1 (Commission et al., 2016). The requirements specified in the standard are derived from seven FRs defined in IEC 62443-1-1. These requirements include 1) Identification and authentication control, 2) Use control, 3) System integrity, 4) Data confidentiality, 5) Restricted data flow, 6) Timely response to events, and 7) Resource availability. Each subsequent part of the IEC 62443 series expands FRs into sub-requirements in the context of various dimensions related to ICS security such as security management system, solution suppliers, risk assessment, system security, component security. Sub-requirements can be standalone or are enhanced by *Requirement Enhancements (REs)* that determine the level of security for a particular requirement. REs are meant to intrinsically provide additional security for a particular requirement implementation. Table 6.2 provides an excerpt of IEC 62443-4-2 security requirements composition. For example, CR 4.2 (Information persistence), the sub-requirement of FR4 (Data confidentiality), provides two REs requiring erasure and erase verification of critical device data in the ICS devices. The standard requires the implementation of these REs to achieve higher security levels.

Table 6.2: A sample composition of security requirements in IEC 62443-4-2

Foundational Requirement	Component Requirement	Requirement Enhancements
FR2 - Use control	CR2.1 - Authorization enforcement	*RE1*: Authorization enforcement for all users
		RE2: Permission mapping to roles
		RE3: Supervisor override
		RE4: Dual approval
FR4 - Data Confidentiality	CR4.2 - Information persistence	*RE1*: Erase of shared memory resources
		RE2: Erase verification

IEC 62443 standard defines four Capability Security Levels (SL-C) that depend

upon the severity of the attack and the attacker's skills, resources, and motivation level. For example, SL-C 1 is described as the protection against passive attacks such as eavesdropping or casual exposure and, unintentional incidents. Higher SL-Cs are characterized by deliberate attempts to inflict harm on an ICS. Since an attacker's ability can vary depending upon the degree of resources, skills, and motivation, SL-Cs assist in categorizing the ability of an attacker to inflict harm. Therefore, SL-C 2, 3 and 4 relate to low, moderate and high ability, respectively.

The presence of one or more RE is usually associated with an enhanced SL-C. However, in some cases, implementing a CR with no REs may suffice to the maximum SL-C. For example, CR 2.1 (Authorization enforcement) shown in Table 6.2 is a sub-requirement of FR2 (Use control) of IEC 62443-4-2, has four REs. Figure 6.2 shows a CR 2.1 REs to SL-C mapping. In order to achieve SL-C 1, CR 2.1 needs to be implemented in its basic form. RE 1 and 2 should be implemented along with base CR 2.1 to achieve SL-C 2. Moreover, CR 2.1, RE 1, 2 and 3 should be implemented for SL-C 3, and for SL-C 4, RE 4 should also be implemented.

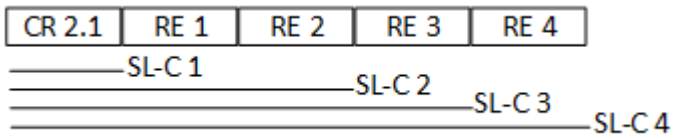

Figure 6.2: An example of IEC 62443-4-2 SL-C mapping

Cyber Security Requirements Specification (Gunter et al., 2018) contains the overall security requirements of a system exhibiting its security aspects. The security requirements cover multiple sub-systems across the different ICS layers. CSRS can be developed as a separate artifact or integrated with the existing requirements specification of a system. Current literature lacks details about the composition of the CSRS. Its characteristics are briefly discussed in (Gunter et al., 2018) that closely relates it to the system requirements specifications. Nevertheless, requirements representation in

CSRS would deal with the same issue of generic requirements specification, i.e. lack of formal methods. In (Martins & Gorschek, 2016), the authors note that the industry practitioners still prefer the natural language to specify the safety requirements and that there is a dearth in using formal specification techniques. There are methods such as (Ramadan, Salnitriy, Strüber, Jürjens & Giorgini, 2017) that use semi-formal techniques like UMLsec to verify the architecture and design. However, their empirical affinity with ICS design and implementation is yet to be seen. Moreover, IEC 62443-4-1 (62443-4-1, 2018) recommends that security requirements specifications shall be traceable to the ICS design and implementation which needs to be explored regarding CSRS.

A labelled property graph database uses a graph structure for storing and managing data, allowing the modelling of real-world entities as nodes and edges (C. Sharma, Sinha & Johnson, 2021). Nodes are used to store data, and relationships between data are stored as edges (C. Sharma et al., 2019; C. Sharma & Sinha, 2019; C. Sharma, 2020). In the LPG database, nodes and edges have labels associated with them. Labels serve as a medium to associate nodes and edges with certain groups. An LPG can also have properties associated with nodes and edges (C. Sharma et al., 2021). Properties exist in the form of *key–value* pair, which serve as an internal structure for storing data in an LPG database. We illustrate a comprehensive example of an LPG containing requirement and requirement-enhancement nodes along with their properties in Figure 6.3, Section 6.7.

6.6 Related Works

A literature review was conducted by choosing the relevant works important to our proposed approach. We examine the existing methods of storing requirements in a requirements repository in diverse database technologies. Relevance of the current methods to our research is discussed below:

The concept of requirements repository has been proposed for requirements reusability in various processes and models such as SIREN (Toval et al., 2002), and SREP (Mellado et al., 2007). However, both methods use repositories or catalogue in textual or semi-formal forms (Souag, Mazo, Salinesi & Comyn-Wattiau, 2016). Also, (Palomares, Quer & Franch, 2017) finds out that most of the requirements reuse techniques are textual copy based and that there is a direct relationship between the requirements reuse and the adopted technology. Moreover, methods that extensively adopt requirements repositories such as (Toval et al., 2002), (Karatas, Iyidir & Birtürk, 2014), (Schmitt & Liggesmeyer, 2015) use them to construct requirement specification documents. The repository usability reduces at the design and implementation stage due to the manual linkages of the requirements.

SREP (Mellado et al., 2007) is closely related to our research such that it integrates Common Criteria (CC) (ISO/IEC 15408) security standard in the early stages of the software development process. One of the fundamental techniques used by SREP is a repository containing assets, threats, and security requirements for reuse. Another CC-based method for early requirement elicitation process is presented in (Houmb, Islam, Knauss, Jürjens & Schneider, 2010). It uses a heuristics tool for elicitation of CC requirements and uses UMLSec for security requirements' design that traces back to security requirements. It recommends using the requirements repository for storage, although the repository is not part of the model. Also, the method in (Houmb et al., 2010) does not explore ICS design and its related development standards such as IEC 61499. We explore using a security standard based requirement repository for the later phases of the ICS product development covering the whole spectrum of development life-cycle contrary to SREP that uses the repository at the early stages of development.

After going through the literature that uses requirements repositories extensively, it can be observed that these approaches do not particularly emphasize how data are being stored inside the repositories. Most of the repositories store data in the textual

form written in natural languages. Essentially this means that the repository does not capture links between requirements rendering repositories less reliable (Z. Wang, Chen, Zheng, Li & Khoo, 2019).

An approach to store security requirements in a schema-less XML database has been discussed in (Morimoto & Cheng, 2007). However, this approach is semi-automated and only supports the representation of a tree-structured graph data model. Furthermore, having an entirely schema-less graph database can increase the chances of data corruption (Pokornỳ, 2016, 2015).

A schema-based approach to organising information related to security standards has been discussed in (Morimoto et al., 2006). The discussed schema uses a hierarchical structure to organise information related to components of security objectives. The authors have used a relational database to manage and store information, and they use Structured Query Language (SQL) to retrieve information stored in the database. A relational query language like SQL may result in a longer response time when data is highly interconnected. Graph databases are efficient in storing and managing highly interconnected data. Furthermore, relationships between data are calculated at the database creation stage; therefore, unlike relational query languages, the relationships are not calculated when data are retrieved from the database. It can become a predicament when the requirement set is large, and the relationships or hierarchies are modified frequently between the requirements.

Resource Description Framework (RDF) for requirement engineering and management is proposed in (Cornière, Fortineau, Paviot & Lamouri, 2016; Runde, Fay & Wutzke, 2009; Ahsan, Motla & Azeem, n.d.). RDF is a W3C recommendation for data exchange over the web and provides a query language SPARQL. However, storing data as an RDF graph often results in a dense graph (Chawuthai & Takeda, 2015). Furthermore, RDF graphs support the storage of property labels in the edges of the graph (C. Sharma et al., 2019; C. Sharma & Sinha, 2019; C. Sharma et al., 2021). The

use of LPG databases is advantageous in this context, as metadata related to edges can be stored as properties in an LPG database (C. Sharma et al., 2021).

The current approaches to store requirements using relational or XML databases do not mainly concentrate on their relationships. However, ICS is evolving rapidly as a consequence of technological evolution. Due to their semantic rigidness, such methods also do not scale well to changing requirements in ICS development scenarios. Moreover, current literature does not focus on integrating system and ICS security standard requirements and relationships, which is eminent in ICS security certifications.

We solve the lack of emphasis on robust requirements relationships by storing and integrating CSRS and 62443-4-2 security requirements in the form of a repository. Integration of requirement sets requires a properly defined relationship between requirements and any obligatory sub-requirements, especially in safety-critical ICS projects. Therefore, graph databases help produce highly connected requirements since graph-based solutions emphasise relationships rather than individual entities and requirements.

6.7 Extending IEC 62443-4-2 Requirements Structure for Standard Implementations

This section discusses the IEC 62443-4-2 requirements structure and proposes extending its existing structure by providing directions for requirements implementation using industry-standard methods. Furthermore, a formal definition of the extension in terms of LPG is specified related to contribution two listed in section 6.3.

We extend the basic structure of IEC 62443-4-2 requirements specification by adding guidelines regarding standard cryptographic methods and algorithms for ICS security certifications. IEC 62433-4-2 specifies CRs at an abstract level. The detailed implementation of such requirements depends on the capability of components. A CR may

be implemented using different security methods based on the required level of security. Therefore, “standard" methods and algorithms are added as implementation guidelines for a possible satisfaction of a particular requirement. Moreover, a requirement may further direct the use of another security standard for selecting methods and algorithms, thus forming a chain of security standards based on which a particular requirement is realised (Mussmann et al., 2020).

The requirement structure is essentially divided into two parts i.e. the inherent requirements in the standard and standard implementations of such requirements in the form of methods and associated algorithms. For example, the general practice uses the Message Authentication Code (MAC) method to enforce integrity. However, MAC can further be implemented by various algorithms such as SHA1, SHA256, SHA512. Some requirements may be implemented by external components such as anti-virus/malware tools or Intrusion Detection and Prevention Systems.

In this article, we use LPGs for formally defining the IEC 62443-4-2 requirements structure. As mentioned in Section 6.5, nodes and edges in an LPG have labels associated with them. Let $\mathcal{L}_{\mathcal{N}}$ be a set of node labels and $\mathcal{L}_{\mathcal{E}}$ be a set of edge labels such that $\mathcal{L}_{\mathcal{N}} \cap \mathcal{L}_{\mathcal{E}} = \varnothing$.

Nodes and edges in an LPG also have properties associated with them. Properties exist in the form of key-value pair where properties values are atomic entities. Let $\mathcal{K}$ be a set of keys (e.g. id, isValid, etc.) and $\mathcal{V}$ be a set of values (e.g. 1426, `TRUE`, etc). We define a set of properties $\mathcal{P} \subseteq (\mathcal{K} \times \mathcal{V})$.

Definition 4. `IEC 62443-4-2 graph` An IEC 62443 requirement graph is defined as a tuple $\mathcal{G} = \langle \mathcal{N}, \mathcal{E}, \mathcal{L}_{\mathcal{N}}, \mathcal{L}_{\mathcal{E}}, \mathcal{P}, \eta, \xi, \rho \rangle$ where

- $\mathcal{N}$ is a finite set of nodes and $\mathcal{E}$ is a finite set of edges and $(\mathcal{N}, \mathcal{E})$ is a directed acyclic graph
- $\eta : \mathcal{N} \rightarrow \mathcal{L}_{\mathcal{N}}$ is a node labelling function which maps all nodes to labels in the set

of node labels $\mathcal{L}_\mathcal{N}$ such that $\mathcal{L}_\mathcal{N} = \{\texttt{R}, \texttt{RE}, \texttt{M}, \texttt{A}\}$ where:

  - R is a label for requirements
  - RE is a label for requirement enhancements
  - M is a label for methods
  - A is a label for algorithms

- $\xi : \mathcal{E} \rightarrow \mathcal{L}_\mathcal{E}$ is an edge labelling function which maps all edges to labels in the set of edge labels $\mathcal{L}_\mathcal{E}$.
- $\rho : (\mathcal{N} \cup \mathcal{E}) \rightarrow 2^\mathcal{P}$ is a property labelling function which maps all nodes and/or edges to all possible subsets of the property set $\mathcal{P}$.
- $\mathcal{E}$ is restricted such that for any $n_1 \rightarrow n_2$
  - $\eta(n_1) = \texttt{R} \Rightarrow \eta(n_2) = \texttt{RE}$
  - $\eta(n_1) = \texttt{RE} \Rightarrow \eta(n_2) \in \{\texttt{RE}, \texttt{M}, \texttt{A}\}$
  - $\eta(n_1) = \texttt{M} \Rightarrow \eta(n_2) \in \{\texttt{M}, \texttt{A}\}$
  - $\eta(n_1) = \texttt{A} \Rightarrow \eta(n_2) = \texttt{A}$

Figure 6.3 illustrates the formalism shown in definition 4. The IEC 62443-4-2 requirement *CR4.2* requires a component to erase all the information if it is being discharged from an ICS. It has two REs, i.e. *RE1* simple data erasure for SL-C 2 and further erasure verification to achieve SL-C 4. However, Figure 6.3 is an excerpt of the requirement *CR4.2*. We show only one RE in the graph for the sake of simplicity. The standard NIST SP 800-88: Guidelines for media sanitization (Kissel, Scholl, Skolochenko & Li, 2006) suggests *clear*, *purge* or *destroy* methods to achieve data erasure. To implement such methods, a variety of data wiping algorithms such as

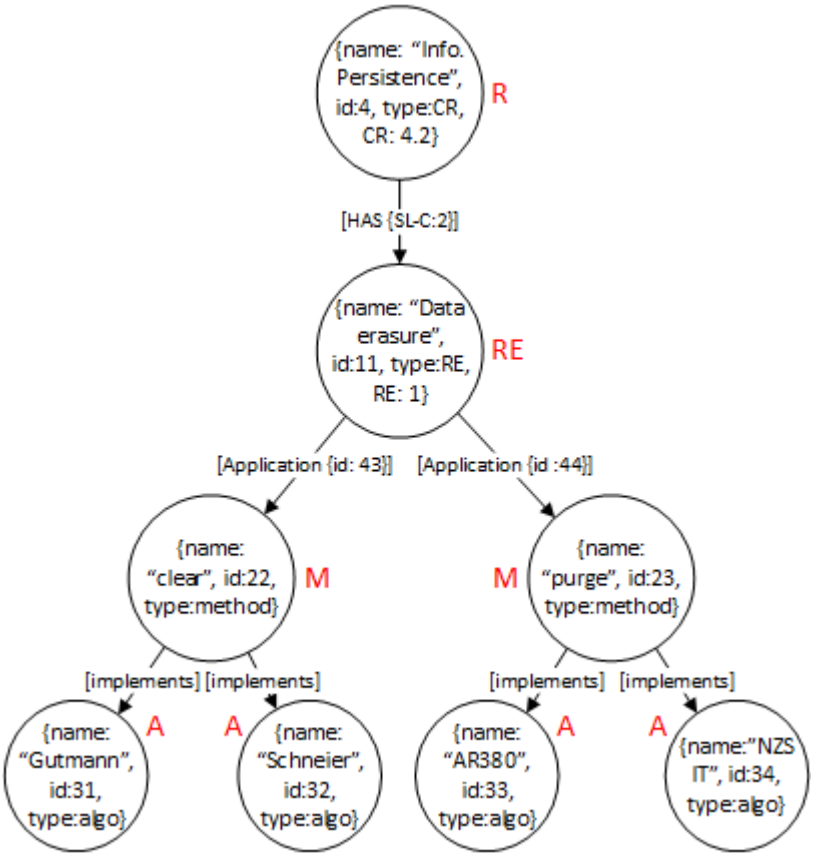

Figure 6.3: IEC 62443-4-2 Extended requirements structure based on formalism described in Definition 4

Schneier, *Gutmann*. are available. One algorithm can be chosen for the implementation based on what is appropriate for the target ICS context and environment.

The slow-changing nature of the IEC 62443 standard makes its LPG graphs highly reusable for other projects that need to fulfil the standard requirements. Therefore, it is beneficial to store them in a repository for later referencing.

6.8 Security Requirements LPGs

This section relates to *requirements graph generation* stage shown in Figure 6.1. This stage captures the security requirements from a CSRS and the IEC 62443-4-2 security standard. The process involves organising the requirements in a hierarchical structure that is a challenge, especially for the large IEC 62443-4-2 requirements document. Such an application enables the linking of security requirements benefiting in intuitive traceability of requirements.

6.8.1 IEC 62443-4-2 Property Graph

We demonstrate the IEC 62443-4-2 security requirements LPG in Neo4j (Lal, 2015) graph database tool to show the practical implementation of formalism presented in Section 6.7. Although there are seven FRs in the standard, we show an example of LPG specification of *FR3* that makes the basis to achieve the integrity goal in ICS. *FR3* is chosen due to its applicability for integrity requirements *IRa* and *IRb* for the wind turbine system, as listed in Table 6.1. *IRa* requires data integrity between the wind turbine PLCs and external components, while *IRb* requires data integrity between master and slave PLCs. Both the requirements relate to *FR3* that is a system integrity requirement specified in IEC 62443-4-2.

The property graph for *FR3* is shown in Figure 6.4. IEC 62443-4-2 breaks down an FR into more than one CRs. Furthermore, the LPG also illustrates the IEC 62443

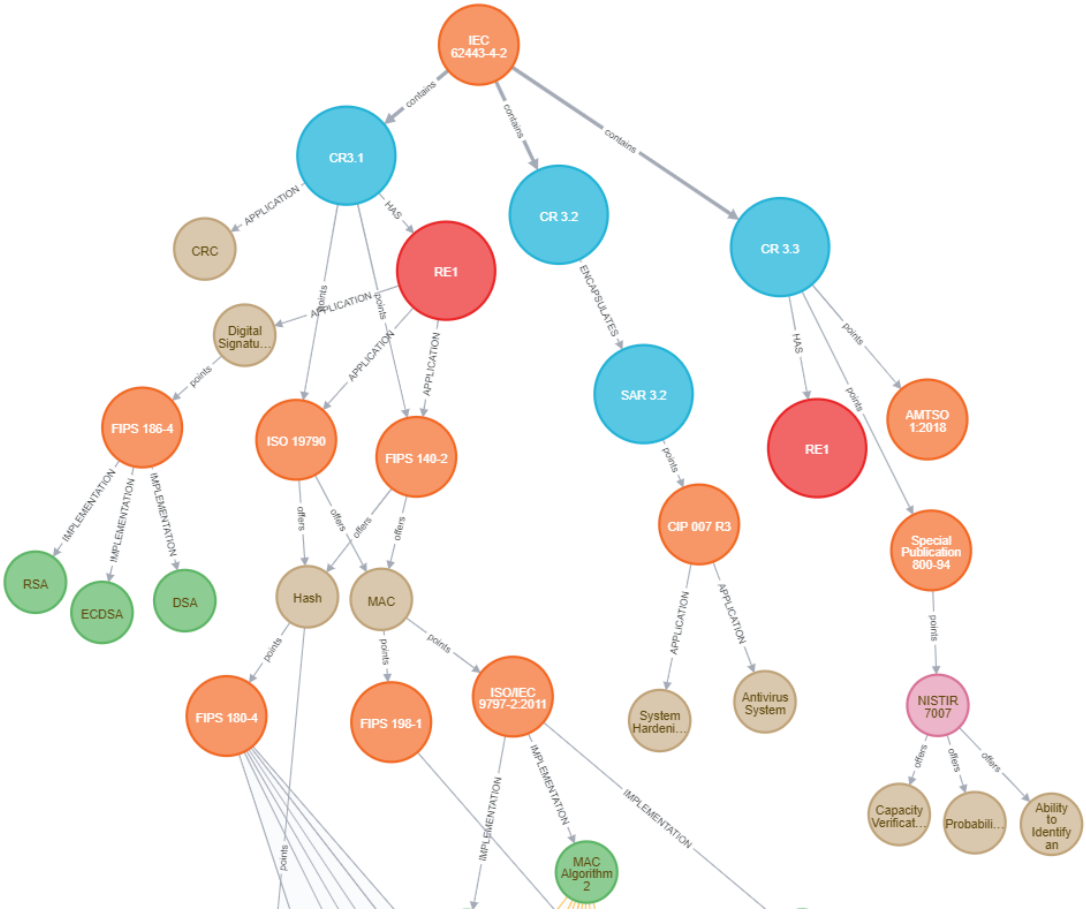

Figure 6.4: IEC 62443-4-2 Integrity property graph produced by Neo4j

security standard's requirement structure extension as described in Section 6.7. IEC 62443-4-2 node in the graph specifies three sub-requirements *CR3.1*, *CR3.2* and *CR3.3*. Furthermore, *CR3.1* and *CR3.3* have specific REs that determine the capability security level of the desired requirement. We have extended the graph by adding methods and algorithms required to implement these requirements and REs. Each method/algorithm is derived from the various recognised security standards. For example, *RE1* of *CR3.1* can be implemented using digital signatures. Therefore, the security capability level for *CR3.1* can be achieved using digital signature algorithms as advocated by the FIPS 186-4 standard. Similarly, *RE1* of *CR3.1* also implies the use of hashing and MAC methods. ISO 19790 (Kusumah & Andriawan, 2019) offers such algorithms. Implementation of these algorithms is further discussed in FIPS 180-4 and FIPS 198-1. Edges between a CR and its RE also contain the SL-C (not depicted in Figure 6.4). Such a feature helps create a path or extraction of subgraph to show the selection of methods/algorithms for the desired SL-C. Extraction is made possible with the help of queries. We use Neo4j to generate LPGs. Neo4j supports the Cypher query language to extract the data from an LPG.

6.8.2 CSRS Property Graph

We use an LPG to organise and structure the CSRS security requirements. We use this data model because an LPG assists in embedding additional data related to nodes and edges as key-value pairs. One of the main advantages of using this technique is the compatibility with the formal syntax of the proposed extended 62443-4-2 requirements organisation proposed in Section 6.7.

The structure of a CSRS incorporates system-level and component type requirements. The system-level requirement may be fulfilled with third-party systems, e.g. an obligation to install the anti-virus software on the machines at a higher level of the ICS

or a provision of Intrusion detection and prevention system at ingress points of the system or the zone boundaries. On the other hand, the component type requirements deal with security provided by a software or hardware component that in-house developers or a third-party vendor may implement. An example of such a requirement can be communication confidentiality between sensor nodes and the PLCs or between multiple PLCs in the wind turbine system. This article's primary focus is the component type requirements of the wind turbine system linked to IEC 62433-4-2 CRs.

A CSRS property graph of the wind turbine system is illustrated in Figure 6.5. It primarily shows the component type requirements listed in Table 6.1. A CSRS document acts as a root node of a CSRS graph. The graph is divided into two branches of system-level and component type requirements. It is further sub-divided into confidentiality, authentication and integrity nodes to represent the security goals. Each goal has respective security requirement nodes. LPGs allow marking the edge label of the nodes. Therefore, each component type requirement in the CSRS is marked with a label associated with the SL-C of the individual requirement as shown in Table 6.1. Such an edge label acts as filtering criteria for searching appropriate requirements in the IEC 62443-4-2 graph. The filter allows the extraction of requirements and guidelines to their standard implementation from IEC 62443-4-2, corresponding to the SL-C of the particular CSRS requirement.

6.9 Security Requirements Repository

This section describes the proposed LPG security requirements repository (contribution one listed in Section 6.3). We define the requirements repository as integration of CSRS and IEC 62443-4-2 property graphs. The conceptual architecture of the repository in terms of its logical and process views is discussed. Cypher query implementation of the repository is also carried out to realise the repository's concept.

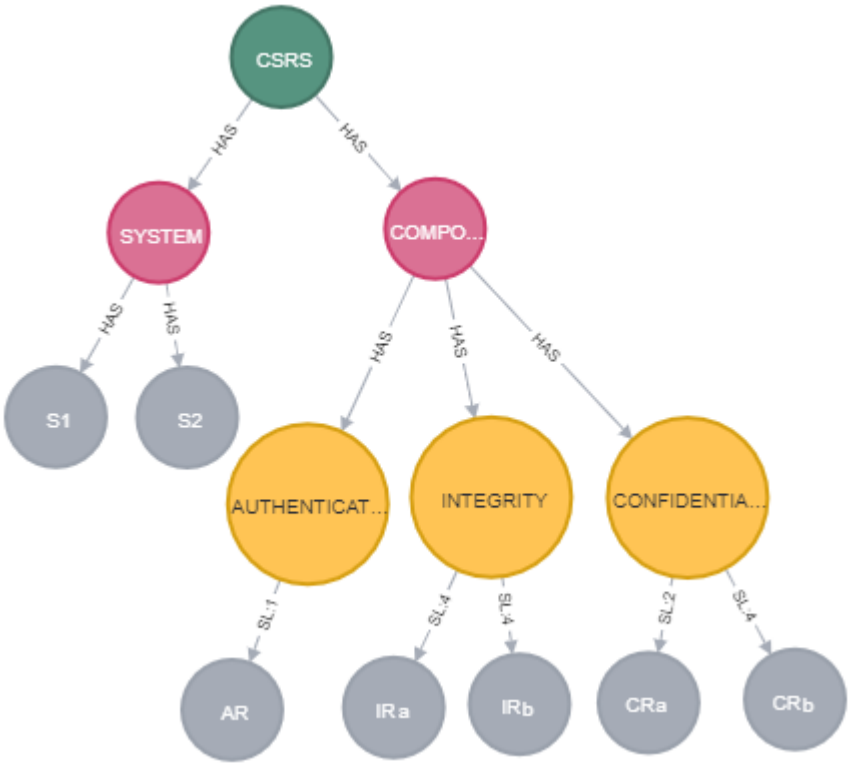

Figure 6.5: A CSRS graph of a wind turbine system created in Neo4j

6.9.1 Repository Architecture

Figure 6.6 shows the storage and integration process of CSRS and IEC 62443-4-2 graphs discussed in Section 6.8, to promote their reusability for different ICS projects. Individual graphs are created in isolation and stored in the repository. Cypher queries are used to extract the associated requirements from IEC 62443-4-2 graph based on a CSRS requirement. This integration of graphs results in a unique requirements subgraph structure for each CSRS requirement. Graph database tool such as Neo4j is leveraged to store the graphs. Optionally, the resulting subgraphs can also be stored in the repository for later reference.

Definition 5. **`LPG repository`** Given a CSRS graph G_{csrs} and n ($n \geq 1$) security standard requirements graphs $G_{s1}, \ldots, G_{sn}$, a requirements repository for a single system is defined as:

$$G_{rep} = G_{csrs} \cup G_{s1} \cup \ldots \cup G_{sn}$$

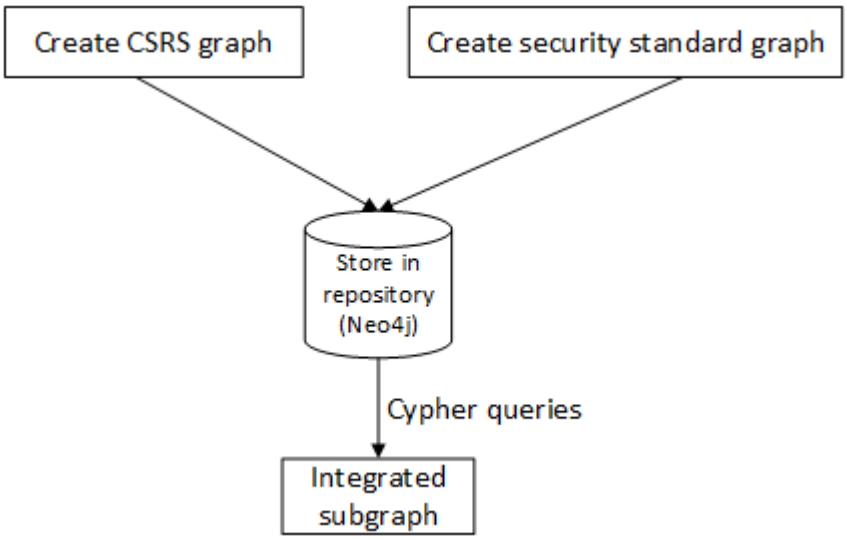

Figure 6.6: Process for storing and integrating CSRS and security standard graph

The combination of CSRS and IEC 62443-4-2 LPGs forms a repository. For a single system, an instance of a repository (G_{rep}) consists of a CSRS and one or more security standard graphs as shown in definition 5. In an overall context, the repository acts as a container of graphs that can also be used to collect trees from different projects. At any point in time, the repository contains the graphs created from the security standard that are static because of their consolidated requirements. The CSRS requirements graphs, however, are dynamic since the system requirements in CSRS may change often. Therefore, the repository can be divided into two partitions, i.e. static and dynamic partition.

Figure 6.7 shows an abstract view of the two logical partitions of the repository. Multiple CSRS property graph specifications belonging to different projects can be stored in the repository's dynamic area, thereby increasing the size of the partition over time. Such a scheme allows one-to-many relationships between CSRSs and the security standard, reflecting the common development practice. Therefore, different projects can take advantage of static graphs of IEC 62443-4-2 and linking them to multiple CSRSs

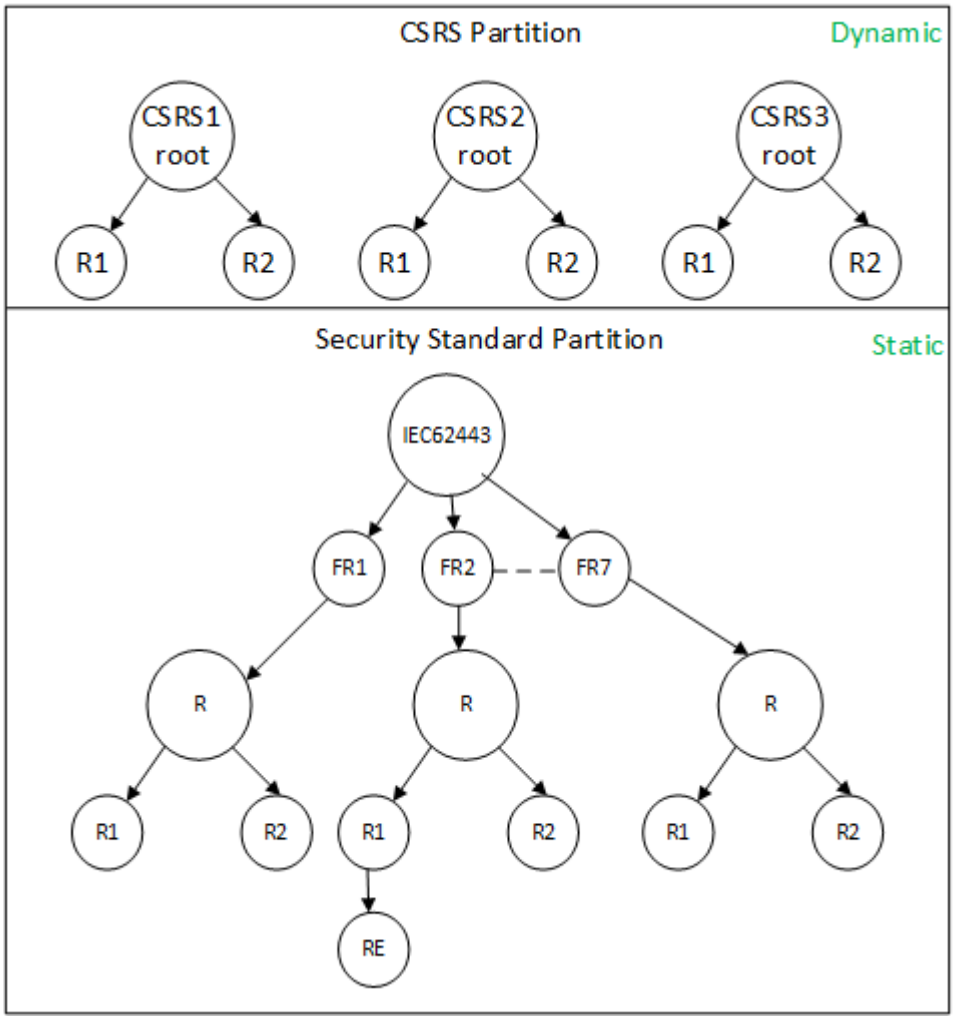

Figure 6.7: Repository logical partitions

reducing overall time and effort from requirements elicitation.

6.9.2 Repository Implementation

For the wind turbine case study, Neo4j stores the CSRS and IEC 62443-4-2 property graphs in a single graph file that creates a logical repository. Both property graphs are created separately through the use of Cypher queries.

The Cypher query in Listing 2 is applied to create the wind turbine CSRS property graph described in Figure 6.5. The IEC 62443-4-2 graph can be created using similar queries. Listing 3 shows an example query used to create the relationships between the IEC 62443-4-2 requirement nodes from Figure 6.4. SL-C labels used between the edges of CRs and the REs are a significant set of information. They are critical to filtering out the security requirements from the standard based on the capability security level specified in a CSRS requirement. For example, SL-C:4 on line 4 of Listing 3 specifies the label over the edge of CR3.1 and its RE1. It helps define a path to the associated cryptographic primitives required to achieve capability security level 4 for CR3.1. Listing 3 is a snippet of a larger query that is not included here due to space concerns.

The built-in feature of Neo4j that executes and stores a set of queries in a designated database also complements the idea of our proposed repository. The results of an executed query become a part of the database that allows registering different database views for later use. The wind turbine CSRS and IEC 62443-4-2 *FR3* requirement graphs resulting from queries in listings are stored as separate entities in the same graph database. Therefore, both the graphs can be merged, or the data can be extracted through subsequent Cypher queries. For example, wind turbine's security requirement *IRa* requires to be implemented at SL-C 4 as specified in CSRS. Listings 4 shows the Cypher queries to deduce the required standard method/algorithms necessary to

Algorithm 2 CSRS graph creation

```
CREATE (cr:CSRS{name:"CSRS"})-[:HAS]->
(co:COMPONENT{name:"COMPONENT"}),
(cr)-[:HAS]->(sy:SYSTEM{name:"SYSTEM"}),
(sy)-[:HAS]->(:LEAF{name:"S1"}),
(sy)-[:HAS]->(:LEAF{name:"S2"}),
(co)-[:HAS]->(con:CONFIDENTIALITY{name:"CONFIDENTIALITY"}),
(co)-[:HAS]->(auth:AUTHENTICATION{name:"AUTHENTICATION"}),
(co)-[:HAS]->(inte:INTEGRITY{name:"INTEGRITY"}),
(con)-[:SL-C{type:2, name:"SL-C:2"}]->
(:CON_REQ{name:"CRa"}),
(con)-[:SL-C{type:4,name:"SL-C:4"}]->
(:CON_REQ{name:"CRb"}),
(auth)-[:SL-C{type:1,name:"SL-C:1"}]->
(:AUTH_REQ{name:"AR"}),
(integ)-[:SL-C{type:4,name:"SL-C:4"}]->
(:INT_REQ{name:"IRa"}),
(integ)-[:SL-C{type:4,name:"SL-C:4"}]->
(:INT_REQ{name:"IRb"})
```

Algorithm 3 IEC 62443-4-2 requirements node relationships creation

```
CREATE (s6244311)-[:contains]->(fr3),
(fr3)-[:points]->(s6244342),
(s6244342)-[:contains]->(cr31),
(cr31)-[:HAS {SL-C:4}]->(cr31RE1),
(cr31RE1)-[:APPLICATION] ->(sISO19790),
(cr31RE1)-[:APPLICATION]->(sFIPS1402),
(cr31RE1)-[:APPLICATION]->(mDigitalSig),
(mDigitalSig)-[:points]->(sFIPS1864)
```

implement *IRa*. A portion of the resulting graph is shown in Figure 6.8.

Algorithm 4 Create relationship between CSRS and IEC 62443-4-2 FR3

```
MATCH (a)-[s:SL-C]->(b)
WHERE s.type = 4 AND b.name CONTAINS "IR_a"
WITH DISTINCT b.name as IR
MATCH (x)-[h:HAS{SL-C:4}]->(y)-[*]->(z)
WITH DISTINCT x.label as lbl, IR as ir
MATCH (m{name:ir}), (n{label:lbl})
MERGE (m)-[:NEWCON{createdOn:datetime()}]->(n)
```

Figure 6.8: A partial resulting graph from the execution of Cypher query in Listing 4.

6.10 Design-time Tool Integration

This section discusses contribution three of this article listed in Section 6.3 that extends the use of the repository by integrating it with design-time and traceability tools to

create practical and maintainable ICS security applications.

We use a secure-by-design approach called “secure links" presented in (Tanveer et al., 2020) to link the CSRS LPG graph requirements to the design and implementation of IEC 61499 applications. Secure links development methodology proposes the abstractions for security requirements repository along with TORUS (Sinha et al., 2018) for tracing requirements in ICS applications. We illustrate the detailed use of the LPG repository with the secure links for more practical purposes in Figure 6.9.

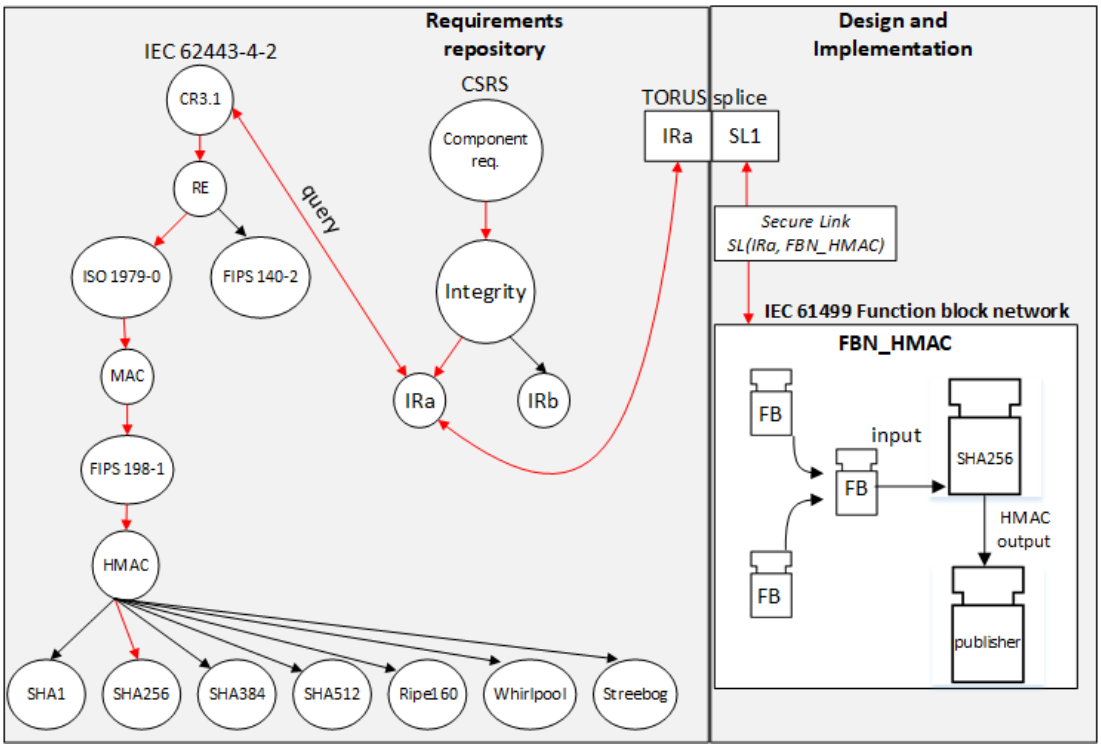

Figure 6.9: An example of TORUS and secure links working with the repository for the wind turbine requirements

The approach of secure links helps in linkages between security requirements and their realisations. Such linkages identify the function block implementation of a security requirement. It results in intuitive requirements tracing capabilities for the designers and the developer during the system development and the requirement change process. Secure links and the repository complement each other by behaving like anchors in

the form of security methods/algorithms presented on the leaf nodes of the IEC 62443 property graph in the repository and its function block implementation in the IEC 61499 ICS application. Each secure link is identified by a unique identifier that references it. For example, a secure link *SL(IRa, FBN_HMAC)* specifies the requirement from wind turbine CSRS where *IRa* is an integrity requirement mentioned in Table 6.1. *FBN_HMAC* is the function block network implementing an integrity mechanism from secure link security library as shown in the design and implementation part of Figure 6.9.

Secure links methodology inherently uses TORUS (Sinha et al., 2018) to link secure links to requirements in the repository. TORUS is a tool that provides tractability for a requirement using its core feature called *splice*. A splice contains a requirement identifier and its meta-data along with a secure link identifier for its implementation. In the current context, it uses the requirements nodes of the CSRS graph as a security requirement. In this case, the meta-data may include abstract level information, e.g. requirement identifier, splice identifier, capability security level, type of security requirement (system or component), and precise information such as security method and algorithm. A design application can leverage a splice to pack the meta-data of a requirement as a link to the implementation. Therefore, the TORUS's use provides the ability to trace the requirements from the requirements to the implementation phase.

Figure 6.9 also illustrates TORUS's application to bridge secure links with the repository to achieve end-to-end requirements traceability for IEC 61499 ICS applications. For example, the splice in TORUS stores the secure link identifier and the wind turbine requirement identifier *IRa* as the meta-data. Such a method can take full advantage of TORUS's requirement tracing capabilities. Cypher queries are used to obtain the recommended methods/algorithms from the repository based on the capability security level. Such queries extract the complete path from the root node of CSRS down to the leaf nodes of IEC 62443-4-2 graphs by joining both graphs. Subsequently, an appropriate

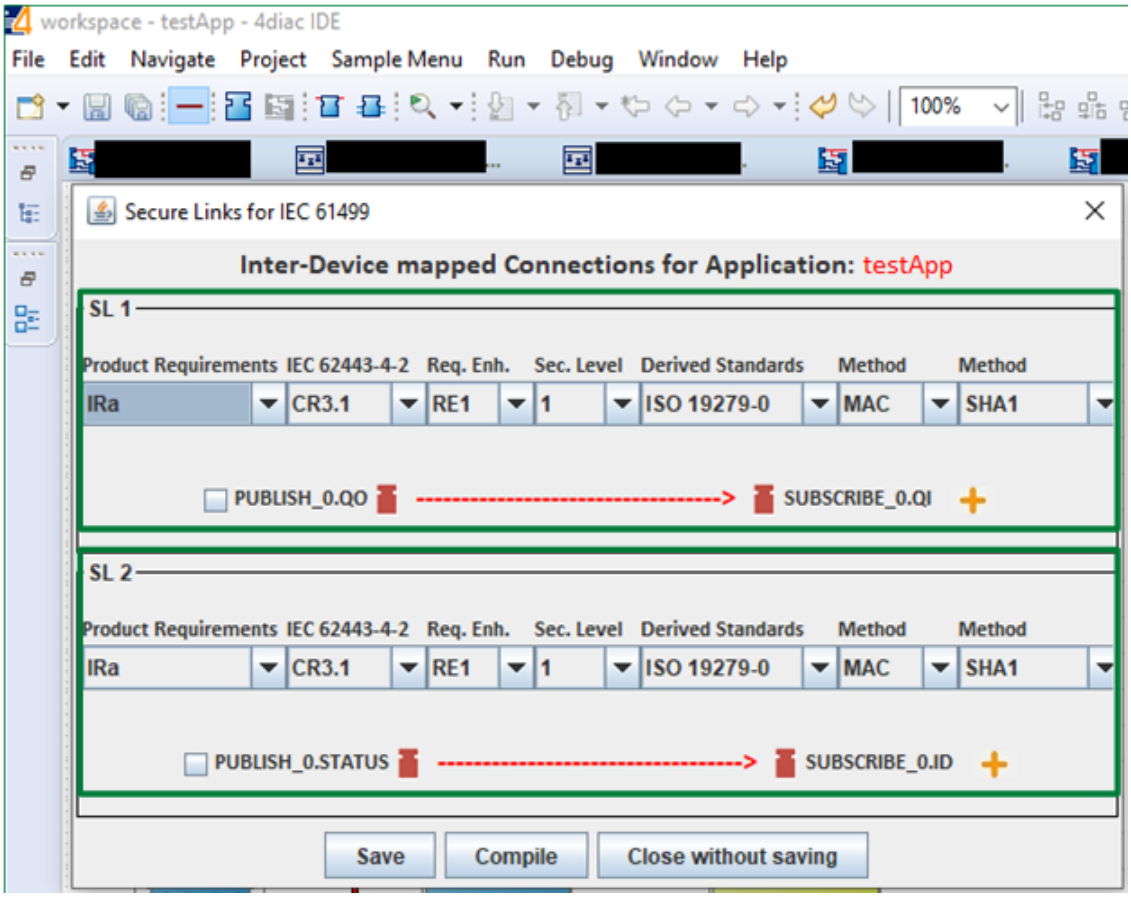

Figure 6.10: A secure links plugin developed using 4diac IDE

method or an algorithm can be selected to fulfil a particular security requirement. It can be achieved by writing queries such as shown in Listing 4, further extending them to filter the required algorithms/method at the leave nodes. The communication details between the repository and the secure links are left out of the scope of this article.

Figure 6.10 shows a screen capture of our implementation of a proof-of-concept plugin for an IEC 61499 based 4diac IDE (*Eclipse 4diac™ - 4diac IDE and 4diac FORTE runtime*, 2021). The plugin can list and identify potential secure links for inter-device mapped function blocks. Figure 6.10 shows two secure links for an IEC 61499 application i.e. *SL1* and *SL2* highlighted within green boxes. Users can select requirements from the repository and select an appropriate function block network based on the selected requirement using the drop-down lists in a secure links panel. The plugin communicates with the repository using Cypher queries to obtain the recommended cryptographic algorithms for the requirement implementation. On the other hand, TORUS links the repository and the secure link plugin semantically to enable requirements tracing.

6.11 Results and Discussions

6.11.1 Requirements Traceability for Industrial Control Systems

Requirements traceability is a critical property for an end to end method of security application development in ICS systems. The complexity of such systems demands forward, and backwards trace of requirements between the early phase of development and the implementation phase (Borg, de la Vara & Wnuk, 2016). The use of secure links and TORUS with the repository allows the security requirements mapping with the IEC 61499 function blocks.

Table 6.3 lists the TORUS splices we created based on the security requirements of

the wind turbine system listed in Table 6.1. The last column shows the actual secure link containing the requirement identifier and the IEC 61499 Function Block Network (FBN) (Vyatkin & of America, 2007).

Table 6.3: TORUS splices linking requirements and secure links for the wind turbine system. RID and SLID refers to requirement and secure link id respectively. SL (Secure Link)

Splice ID	RID	SLID	SL (description)
1	IRa	SL1	SL1(IRa, FBN_HMAC)
2	IRb	SL2	SL2(IRb, FBN_HASH)
3	CRa	SL3	SL3(CRa, FBN_ENC)
4	CRb	SL4	SL4(CRb, FBN_ERASE)

The current example shows the one-to-one relationship between a requirement and an FBN. However, two or more requirements may be implemented using a single FBN depending upon product requirements.

Requirements Traceability Matrix (RTM) essentially provides a map of relationships between requirements and the product artifacts such as design, implementation and tests. Table 6.4 shows resulting security RTM for the wind turbine system that can be readily generated by expanding a TORUS splice from the containing requirement and secure link identifier. It can be done by following the steps from the proposed method in the Section 6.9 i.e. generate the subgraph by combining CSRS and IEC 62443-4-2 graph through Cypher queries based on a requirement identifier. Each row of the table represents the result of a TORUS splice expansion. The first and last column of the table contains the requirement and secure link identifier, respectively. Columns 2–8 show the requirements, enhancements, associated standards, methods, and algorithms associated with the FBN implementation shown in column 9. For example, for the project requirement *IRa*, the RTM shows the appropriate selection of standard requirements such as *CR3.1* and the further REs from the standard required to implement for a particular security level (SL-C 4 in this case). The associated standards give further

guidance to implement requirements in terms of recommended methods/algorithms. For example, ISO 19279-0 recommends using MAC for data integrity and authentication to fulfil *CR3.1*. Furthermore, FIPS 198-1 describes the standard implementation of HMAC that requires hashing algorithms such as SHA1, SHA256. The designer/developer may choose one of the available hash functions for the HMAC in the implementation IEC 61449 function block network, as shown in column 8.

Security methods (columns 5 and 7) can also be selected straightforwardly without any associated intermediary standards. For some requirements such as *CRa* related to confidentiality, the IEC 62433-4-2 does not require RE to achieve the maximum capability security level. Table 6.4 shows the direct mapping of *CR4.3* to the *symmetric* method of encryption, further linking to IEC 18033-3 standard that lists down the recommended block cyphers to be chosen for the implementation of *FBN_ENC*.

Table 6.4: Requirements traceability matrix based on TORUS splices. (PR - CSRS Product Requirement, SR - Standard Requirement, RE - Requirement Enhancement, FBN - Function Block Network, SL - Secure Link)

PR	**SR**	**RE**	**Associated standard**	**Method**	**Associated standard**	**Method**	**Algorithm**	**FBN**	**SL**
IRa	CR3.1	RE1	ISO 19279-0	MAC	FIPS 198-1	HMAC	SHA1 SHA256 SHA224 SHA384 SHA512	HMAC	**SL1**
IRb	CR3.1	RE1	ISO 19279-0	Hash	ISO 10118-3	–	Ripe160 Whirlpool Streebog SHA1 SHA256 SHA224 SHA512	HASH	**SL2**
CRa	CR4.3	–	–	Symmetric	IEC 18033-3	–	TDES AES Camellia SEED MYSTY1 CAST-128 HIGHT	ENC	**SL3**
CRb	CR4.2	RE2	NIST SP 80088	Verify	–	–	DOD AFSSI NCSC Gutmann Pfitzner Schneier Random	ERASE	**SL4**

One of the main advantages of using such an RTM is producing document artifacts containing bidirectional traceability of security requirements. It helps develop ICS products that are meant to be certified against a security standard such as IEC62443-4-2. The other main advantage is showing the requirements to test-case traceability, e.g. linking a security requirement and a unit test of implementing function block in an FBN. It provides a bird's eye view to the stakeholders, such as users, vendors, and the certification bodies, that a product requirement has been fulfilled based on the particular requirement from the security standard. The associated standards also ensure that a robust set of guidance backs up each link in the requirements chain since they are approved by industry practitioners and government after rigorous technical reviews and security testing.

6.11.2 Requirements Change Management

Requirements change is an inevitable part of system development. Figure 6.11 shows the requirements change process that can be accomplished by the proposed method. Requirements may be changed in two ways such that:

1. A requirement may be added or deleted for the product needed to be implemented by a secure link. If a node is to be relocated under a different node within the graph, or if a security goal has been changed, it is deleted and added as a new node. For example, if the use-case for wind turbine requirement *AR* shown in Figure 6.5 is no longer desired or if there is a new use-case of merging *AR* with one of the integrity requirements *IRa* or *IRb*, then the node *AR* is deleted and introduced as a new requirement under the integrity node.

2. The nature of a requirement may get changed, e.g. change in capability security level or metadata. In such cases, a new graph based on a modified CSRS needs to be generated. It can be achieved by updating and re-executing the Cypher

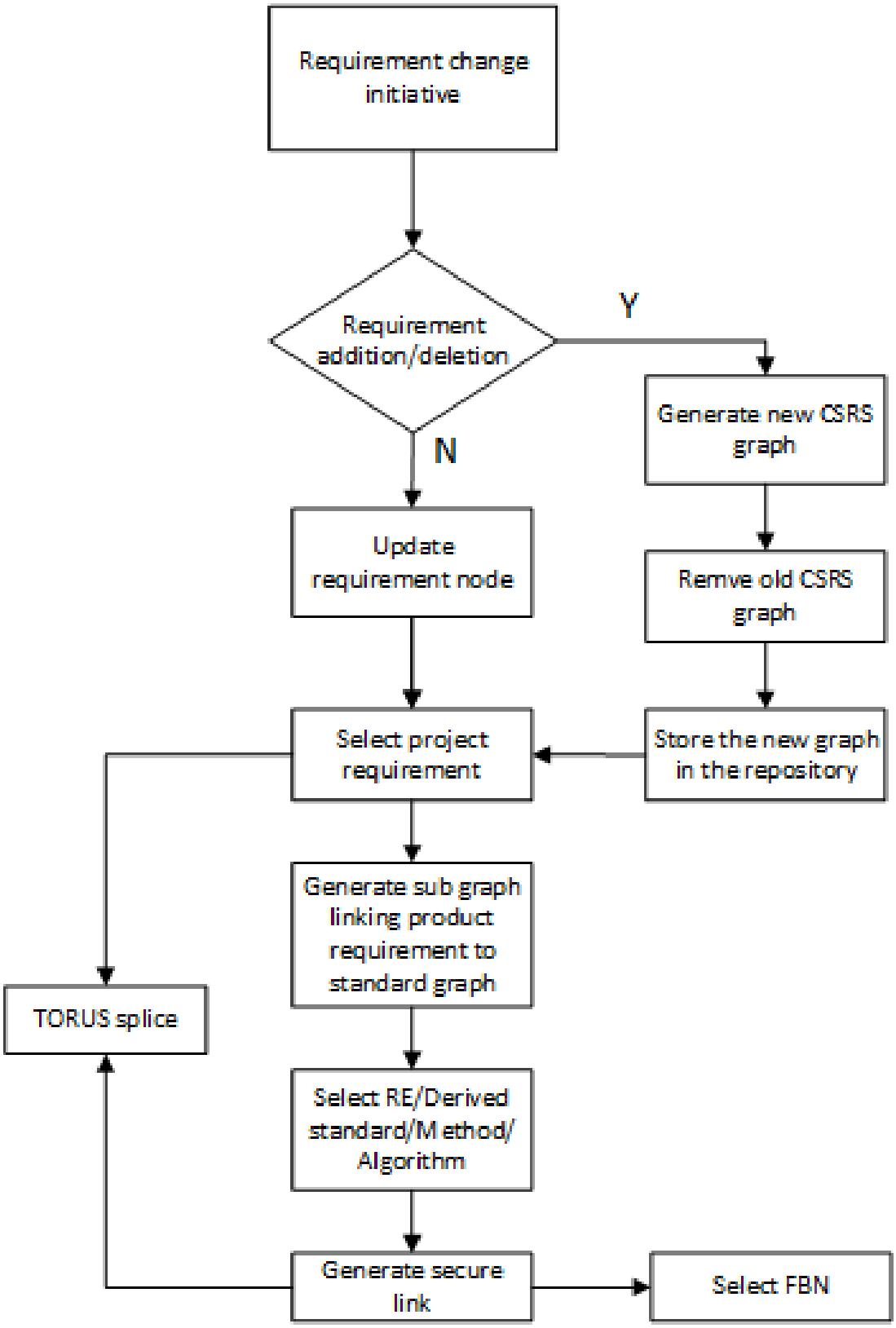

Figure 6.11: Requirements change process using security requirements repository and design-time tools.

queries, subsequently storing the new graph in the repository for later referencing. Regeneration of CSRS graph ensures the consistency and correct linking to security standard graph. If an existing requirement is changed, the node's metadata, such as node name and identifier, is updated. For example, if the SL-C of *IRa* in Figure 6.5 is lowered to level 3, the edge label needs to be changed to reflect the new level in the CSRS graph. The modified SL-C label may produce an entirely different subgraph from the linking of CSRS and IEC 62443-4-2 requirements, containing a different set of associated standards and methods/algorithms.

For any of the above scenarios, the CSRS graph must be updated and stored in the repository. The added/updated requirement can consequently be selected for a secure link. A new subgraph is generated to link it to related FR from the security standard graph stored in the repository. This step generates associated standards and methods/algorithms based on the FRs in the standard. The associated secure link and the CSRS requirement identifiers are stored in the TORUS splice for traceability purposes.

6.11.3 Requirement Traversal and Graph Analytics

Manual selection of requirements is a laborious task in the absence of automated tools, especially for large requirements set when requirements are interconnected and fulfilled by multiple sub-requirements, as shown in our graph model. LPG databases such as Neo4j can query the data stored in a graph database using the Cypher query language. By using Cypher, the requirement repository can be explored locally as well as globally (Needham & Hodler, 2019). Local graph search corresponds to searching smaller subgraphs in the graph database. This can be done by specifying the graph pattern in Cypher (C. Sharma & Sinha, 2019; C. Sharma et al., 2019; C. Sharma, 2020; C. Sharma et al., 2021). Cypher also enables the global search of the repository for efficient requirement traversal. It is facilitated by using the inbuilt graph algorithms

such as shortest path, all shortest path, single-source shortest path, minimum spanning tree, centrality and community detection (Needham & Hodler, 2019). For example, Listing 5 shows a query written in Cypher that can be used to search for all shortest paths starting from the root node of a requirement repository. The `MATCH` and `WHERE` clauses in lines 1–5 are used to search for the *root* node. The `CALL` clause in line 6 is used to specify that to search all shortest paths starting from the *root node*, Dijkstra's algorithm should be used. Finally, as shown in lines 7–8, `YIELD` and `RETURN` clauses are used to output the result set.

Algorithm 5 Search for all shortest paths starting at the root node

```
MATCH (root)
WHERE NOT EXISTS {MATCH ()->(root)<-()}
AND NOT EXISTS {MATCH ()->(root)->()}
AND NOT EXISTS {MATCH ()<-(root)<-()}
AND NOT EXISTS {MATCH (root)<-()}
CALL gds.beta.allShortestPaths.dijkstra.stream
('Conf-graph',sourceNode: id(root))
YIELD index, sourceNode, targetNode, totalCost, nodeIds,
costs
RETURN index, gds.util.asNode(sourceNode).name AS
sourceNodeName, gds.util.asNode(targetNode).name
AS targetNodeName, totalCost, [nodeId IN nodeIds |
gds.util.asNode(nodeId).name] AS nodeNames, costs
```

Moreover, using a graph-based approach to store requirements also means that advanced frameworks for graph analytics such as Spark can be utilized to perform large-scale data analytics.

6.11.4 Requirements Extraction

Requirement elicitation is an essential step at the inception of a project (S. Sharma & Pandey, 2014). The process of collecting and analysing the requirements in large-scale projects is often an exhausting task (Schneider, 2007). The task becomes more complicated when the project requirements are tied up to security standards, as shown in

this article. The analysts have to manually go through all system security requirements to link them with an extensive set of security standard documents.

In the following sub-sections, we compare manual requirements extraction effort against the extraction based on security levels using LPGs from the IEC 62443-4-2 combined with CSRS of the systems such as wind turbine discussed in this article and an Industrial Mixer Control System (IMCS) case study presented in (Tanveer et al., 2020).

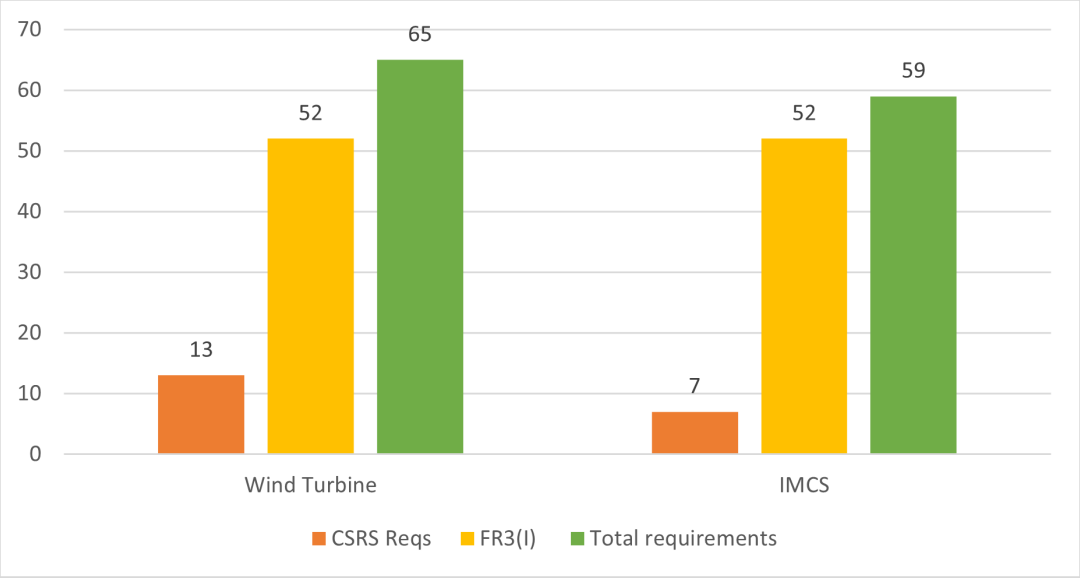

Figure 6.12: Manual requirements extraction

6.11.4.1 Manual Requirement Extraction

The manual process requires analysts to go through all the requirements and evaluate the appropriate methods/algorithms to implement the requirement based on a security level. Figure 6.12 shows the initial manual extraction effort for IEC 62443-4-2 *FR3*. It provides the total security requirements extracted from the CSRS and the security IEC 62443-4-2 standard document. For example, the total number of requirements nodes from wind turbine CSRS are 13. For *FR3* (Integrity), the total requirements to be extracted are 65 i.e. CSRS + *FR3* requirements. Also, the total number of manually

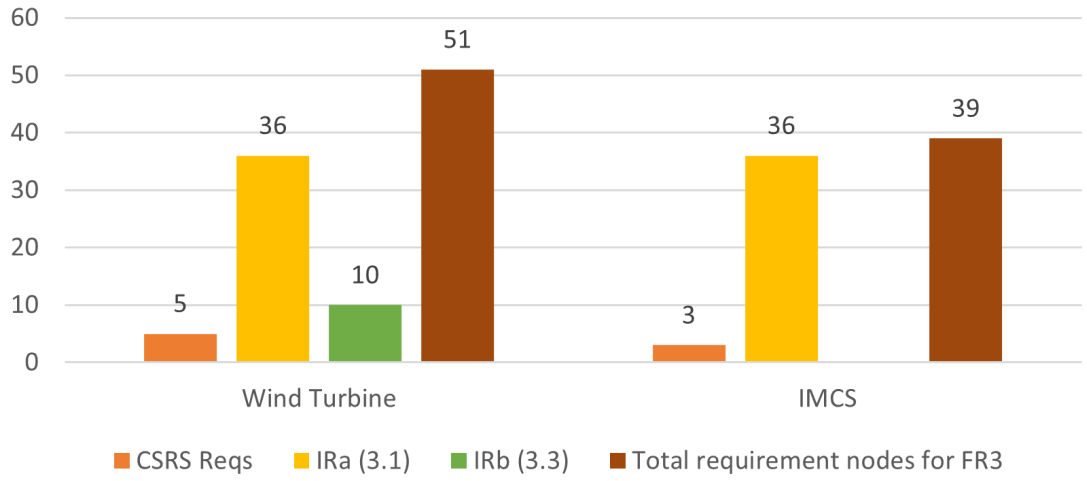

Figure 6.13: Requirements extraction based on security level using LPGs

extracted requirements comes to 57 for the IMCS. It can be observed that the increase in requirements depends on the number of requirements in the project, which is a dynamic component in the repository.

6.11.4.2 Requirement Extraction using LPG

The secure link plugin automates the process of security level based requirements extraction from CSRS and IEC 62443-4-2 LPGs using Cypher queries. Figure 6.13 shows the number of nodes from FR3 when an individual requirement is selected from CSRS using the secure links plugin. It shows that the designers and developers need to deal with the reduced number of requirements for each CSRS requirement selected from the plugin. For example, the total number of requirements for *FR3* are 52 (Figure 6.12). However, when *IRa* that is the sub-requirement of *FR3*, is selected from the secure link plugin, the number of extracted requirements comes down to 36 as shown in Figure 6.13.

From the above comparison, it can be seen that the total number of requirements is filtered down based on a security level when using LPGs and the secure link plugin compared to the manual process. That is, 51 compared to 65 for the wind turbine, and in

the case of IMCS, 39 compared to 59. Although these numbers may not be significant, there will be a considerable improvement if a system has an increased number of CSRS requirements, which may result in implementing more FR sub-requirements.

6.11.5 Repository Reusability

The graph storage feature of the repository renders it highly reusable for developers and testers to build and maintain IEC 62443-4-2 certified ICS applications. The ability to query the repository using Cypher ensures an easy requirements extraction due to the LPGs. The static and dynamic partitions of the repository illustrated in Figure 6.7 provide distinct levels of reusability.

The static partition, i.e. IEC 62443-4-2 graphs, provides high reusability since the standard does not often change. A one-time effort has to be made to create the graph for each security FR of the standard as discussed in Section 6.11.4. Once created, the FR graphs can be linked with the relevant requirements in CSRS. It is beneficial for the certifiers who may have to deal with various products at the same time. Reusable IEC 62443-4-2 requirement graphs help quickly validate and verify a CR or log the correction if needed. For example, a certifier may validate whether a vendor has correctly chosen the methods/algorithms according to the capability security level. Moreover, in conjunction with the graph repository, secure links also help in the implementation verification of the requirement. An RTM discussed in Section 6.11.1 can help in verifying a security requirement by tracing it to the particular FBN implementation. For instance, a certifier can generate an RTM shown in Table 6.4 using secure links plugin, thus verifying the implementation of *CRb* through *FBN_ERASE*.

A vendor may also use such IEC 62443-4-2 graphs as a guideline for designing and implementing security requirements against a particular security level. For example,

a CSRS containing integrity requirement can be linked with the graph of IEC 62443-4-2 *FR3*, as illustrated in Figure 6.8. Querying the graph based on the capability security level reveals the subgraph that conveniently guides the vendor to the required methods/algorithms implementations.

Another effective reusability strategy is to ensure the availability of the IEC 62443-4-2 static repository partition to the broader community of ICS application developers in the form of a library of Cypher queries. Aspiring vendors of 62443-4-2 certified ICS applications may import the graphs and reuse them readily in their projects, thus ensuring community collaboration. For example, a set of cyber queries for the generation of IEC 62443-4-2 FR graphs can be stored in versioning control systems, and the vendors can subsequently use these queries for the instant creation of the graphs. It also enables community contribution helping in maintaining and updating the repository in case of modification of the standard.

Similarly, vendors are also able to reuse the dynamic partition of the repository. Each CSRS graph for a particular project is an archive in the repository for future reuse, as illustrated in Figure 6.7. For example, a project may reuse the graph from an earlier project with similar security requirements, thus reducing the time against requirement analysis and extraction.

6.12 Limitations and Validity Threats

In this section, we discuss some of the pertinent limitations and threats to the validity of our purposed repository model that uses TORUS and secure links, such as illustrated in Figure 6.9.

6.12.1 Reliance on IEC 61499

IEC 61499 has some open issues regarding its industry-wide adoption. A state-of-the-art literature review (Lyu & Brennan, 2020) focusing on the applicability of IEC 61499 identifies its major challenges in industrial practices. The key issue is the industry's reluctance to deviate from legacy ICS development approaches and rely on an older standard like IEC 61131-3. The causes for such hesitation in adopting IEC 61499 are the cost to upgrade the legacy systems, lack of proven redesign methods, practitioners' expertise, and the lack of educational aspects around IEC 61499 regarding course designs and industrial training.

Secure links provide a secure-by-design development approach for IEC 61499 ICS applications. Similarly, the TORUS splice in our approach requires an input consisting of a secure link identifier that refers to the underlying IEC 61499 function block network. Although TORUS can be considered a generalised framework for requirements traceability, its applicability in the current literature is demonstrated majorly through IEC 61499. Therefore, the reliance on the IEC 61449 based tools for the end-to-end traceability limits the backward compatibility with the ICS using legacy standards such as IEC 61131-3.

6.12.2 Scope Limitations

The proposed LPG security requirements repository provide the implementation guidelines regarding security requirements in IEC 62443-4-2 in the form of LPG nodes. These guidelines include standard security mechanisms and their associated cryptographic primitives. The current collection of these guidelines is not extensive, i.e. it specifies limited numbers of standard cryptographic algorithms and methods on a limited set of security requirements in the IEC 62443-4-2 standard. The LPG graphs of the standard need to be comprehensive for the repository to be used in industrial-scale ICS projects.

Moreover, the LPG repository is limited to the IEC 62443 standard. Other ICS specific security standards such as NIST SP 800-82 (Stouffer et al., 2011), FIPS 140-2 (NIST, 2016), and ISO/IEC-15408 (Mellado et al., 2007) can be explored to be used with the repository. It will also allow industry-specific ICS such as smart grids, manufacturing, and gas and oil industries to take advantage of our proposed approach.

6.12.3 Scalability

The wind turbine case study presented in this article is an abstraction of a larger system containing complex requirements. We selected a set of key security requirements (listed in Table 6.1) of the wind turbine system in order to illustrate the traceability aspects of our solution. Therefore, the case study does not cater to all of IEC 62443-4-2 FRs and only the FRs related to the wind turbine requirements are selected for illustration purposes. The limited scope and scale of the requirements may pose a threat to the generalisation of the results. Several different FRs may need to be accommodated in an industrial-scale real-world system, and their respective graphs need to be generated. However, the effort required to extract the requirements from the standard through the manual process is directly proportional to the FRs applied on a system. In contrast, the LPGs provides the capability using Cypher queries to filter the appropriate requirements from the standard based on the security level, reducing the extraction effort as discussed in Section 6.11.4. Therefore, we believe that our proposed solution can scale with the increased number of FRs.

6.12.4 Error-prone Requirements Elicitation and Extraction

The proposed approach assumes that the generation of security standard property graph is comprehensive and accurately translates the implementation of abstract security standard requirements into further cryptography standards and algorithms. However,

such practice requires expertise and extensive knowledge of standards in the information and ICS security realm. For example, the IEC 62443 property graph illustrated in Figure 6.4 is a prototype created based on authors' expertise and knowledge. The interpretation of a security standard requirement may vary between different requirements analysts, especially in the case of assigning derived standards and algorithms to the appropriate security levels described in the standard.

Therefore, there is a risk of producing two distinct graphs of the same IEC 62443 functional requirement that may also induce difficulties in maintaining the property graphs, also providing inaccurate requirements traceability from requirements to code. Automatic security standard requirements extraction tools are needed to produce consistent graphs that will help in more consistent requirement implementations.

6.13 Conclusion and Future works

ICS applications aiming to conform to security standards need robust requirements structure and properly defined requirements relationships to verify and validate conformance to standards. This article proposes a multi-partitioned LPG security requirements repository to store and integrate system (CSRS) and IEC 62443-4-2 standard's security requirements. Furthermore, a formal extension of the IEC 62443-4-2 requirement structure is proposed to provide guidelines to the standard cryptographic primitives required to fulfil a security requirement according to the capability security level. The use of the repository with secure links and TORUS shows a high degree of end-to-end forward and backward traceability of security requirements. We also show an intuitive requirements change process aided by the repository. Furthermore, results significantly improve extracting requirements from large security standard documents by querying LPGs in the repository. Finally, the implications regarding the utility of the repository in the security certification process are discussed.

Future directions include mapping the maximum amount of IEC 62433-4-2 FRs into graphs to test the scalability of the proposed solution that can be achieved by using an extensive case study with additional security requirements. Moreover, exploring the storage of security requirements from ICS security standards other than IEC 62443-4-2 also tends to be an exciting challenge. We also aim to look at the opportunities for performing advanced graph analytics on the repository security requirements that can be performed using our approach.

Chapter 7

Discussions, Conclusions and Future Works

7.1 Introduction

This chapter provides a synthesis of the contributions of this research. It examines the proposed requirements engineering and secure-by-design techniques and their effects on the ICS software development life-cycle. A detailed discussion on the resulting artefacts produced through each iteration of the chosen design science research methodology is carried out. In addition, implications and limitations of the proposed techniques and associated artefacts are also presented.

The chapter is organised as follows:

1. Section 7.2 presents the researcher's perspective on an integrated configuration of the secure-by-design ICS software development techniques proposed in this research. In the rest of this chapter, this unification of techniques is referred to as ITSIS — Integrated Techniques for Secure-by-Design ICS Software. These techniques include secure links, a graph-based security requirements repository, and a function block library for implementing security requirements. The section

discusses the compatibility of the ITSIS with the prominent development methods practised by the industry.

2. Section 7.3 lists the main results of this research obtained through the applications of the proposed techniques. The results include a traceability matrix resulting from the use of TORUS with security requirements repository (Chapter 6) and an improved system complexity and maintainability for ICS (Chapter 5).

3. Section 7.4 discusses the implications and limitations of the artefacts produced in Chapters 3, 4, and 5 to achieve security goals in ICS applications. The artefacts include: *1)* IEC 61499 security layer for confidentiality, *2)* an IEC 61499 Intrusion detection and prevention system helping in highly available ICS devices, and *3)* and function block implementation of integrity mechanisms.

4. Section 7.5 discusses the details and future implications of the S-Lib function block security library that is perceived as a collection of implementation techniques discussed in Section 7.4.

5. Section 7.6 discusses broad implications of ITSIS that include its usability in ICS security certification processes and its compatibility with open-source development approaches.

6. Finally, Section 7.7 provides conclusions and focuses on limitations and future works regarding this research perceived as an integration of security requirements engineering and secure-by-design techniques for ICS software development.

7.2 Integrated Techniques for Secure-by-Design ICS Software (ITSIS)

Figure 7.1 shows the configuration of the proposed Security Requirements Engineering (SRE) and secure-by-design techniques and their position regarding major phases of other development methodologies. The yellow boxes indicate compatible development methodologies. Light green boxes show the proposed techniques in the research, and the dark green boxes show the artefacts of these techniques.

1. An SRE method that uses property graph based security requirements repository discussed in Chapter 6.

2. IEC 62443-4-2 security standard requirements extraction and specification discussed in Chapter 6.

3. IEC 61499 ICS application development standard incorporating a novel approach of secure links development methodology illustrated in Chapter 5.

The aim of ITSIS is to address the security challenges imposed by modern safety-critical ICS environments. It combines the ICS development and security standards such as IEC 61499 and IEC 62443 and applies them over an application development life-cycle compatible with mainstream and the recent development methodologies proposed by the literature. Thus, the method encompasses the traditional phases of development life-cycle such as (security) requirements engineering, a design phase adhering to secure-by-design paradigm and the implementation of security methods and algorithms. One of the main implications of ITSIS is its application in the security certification of ICS components due to its ability to produce conclusive end-to-end traceable requirements. Furthermore, the ability to cater for the requirements changes makes it an appropriate choice for the sophisticated security certified ICS application/component development

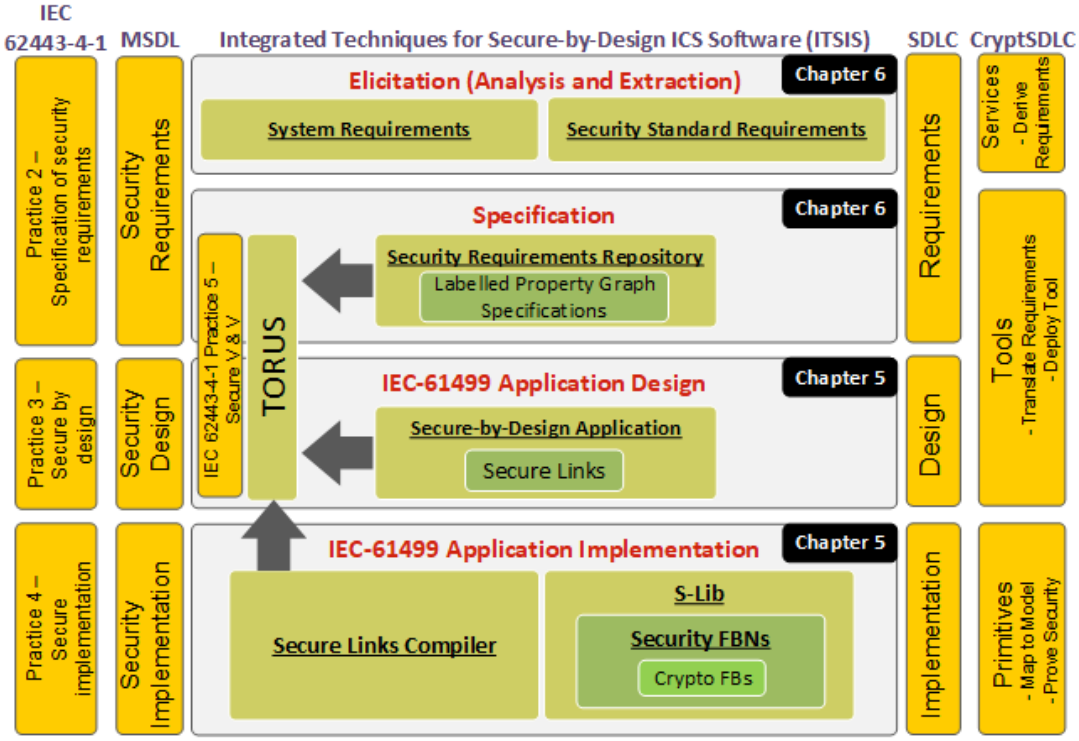

Figure 7.1: An overview of ITSIS

process. Figure 7.1 illustrates the ITSIS marking the superimposition of its techniques over other referenced security system development methodologies such as:

1. *IEC 62443-4-1 – Secure product development life-cycle (62443-4-1, 2018)* is a part of overall IEC 62443 security standard. It provides a set of practices for a development process to certify ICS components providing the security goals. This standard is the most relevant to the proposed methodology since it resides in the same compartment with IEC 62443-4-2. It essentially guides the stakeholders to produce certified products based on the security requirements set in IEC 62443-4-2.

2. *Microsoft's security development life-cycle (MSDL)* is the set of guidelines and a development process for the secure development of generic applications. Being a recognised set of industry practices, the general approach of MSDL provide a

helpful reference point to induce security in ICS.

3. *Cryptographic Software Design Life Cycle (CryptSDLC) (Loruenser, Pöhls, Sell & Laenger, 2018)* is a state-of-the-art architecture that focuses on security services, tools, cryptographic primitives and their artefacts during the security application development. It has been taken as a reference point due to its proximity with ITSIS regarding its artefacts.

4. *Traditional Software Development life-cycle (SDLC) (Howard & Lipner, 2006)* is the universal development methodology of software products. Its pioneering and distinguished approach to overall software development also makes it a reference point for the correlation.

The compatibility of the ITSIS with other security development methods ensures that the development process is consistent with already established approaches. It intends to complement the existing approaches with a novel application of the proposed techniques in each phase of the development life-cycle, rather than proposing a competing methodology proving to be muddled and incoherent for the practitioners. This research aims to propose tasks for primary phases of the application development life-cycles that benefit the state-of-the-art ICS security application development approach, especially required for security certifications based on ICS specific security standards such as IEC 62443.

Table 7.1 shows the compatibility matrix for ITSIS techniques regarding the above-referenced methodologies. The first column shows the list of prominent methodologies and their segments or guidelines. The header row shows the main artefacts of ITSIS. The symbols X and O denote the full and partial adherence of an ITSIS artefact with the stages/elements of correlated development methodologies. Full adherence means that the entities are fully compliant with each other. In contrast, partial adherence indicates that the ITSIS artefact complies only with those activities of a particular phase of a

Table 7.1: ITSIS consistency with other development methodologies

	ITSIS Artifacts						
Compatible Methodologies	System Requirements Property Graph	Security Standard Requirements Property Graph	Security Requirements Repository	TORUS	Secure links	Secure Links Compiler	S-Lib
IEC 62443-4-1							
Practice 1 – Security management			O				
Practice 2 – Specification of security requirements	X	X	X				
Practice 3 –Secure by design					X	X	
Practice 4 – Secure implementation					O	X	X
Practice 5 – Secure V & V			X	X	X		
Practice 6 – Management of security-related issues							
Practice 7 – Security update management			X	X	X		
Practice 8 – Security guidelines		X					X
Microsoft SDLC							
Security Requirements	X	X	X				
Security Design					X		
Secure Implementation		X				O	X
Verification							O
CryptSDLC							
Derive Requirements	X	X					
Translate Requirements			X		X		
Map to Model			X		X	X	X
Prove Security		X			X	O	X
Deploy Tool					X	O	X
Extract Capabilities			O	X	X		X
SDLC							
Requirements	X	X	X				
Design					X		
Implementation						X	X
Verification and Validation			X	X	X		

X — Full Compatibility, O — Partial Compatibility

methodology that fits in the scope of ICS security application development. Empty table cells imply the non-applicability of an artefact for a particular phase. The following discusses the phases in which ITSIS techniques are applied in correlation with the above-referenced methodologies:

7.2.1 Security Requirements Engineering

The SRE phase consists of two primary ITSIS artefacts: security requirements Labelled Property Graphs (LPG) and the storage of these graphs in the form of a requirements repository shown in *Specification* stage in Figure 7.1. These artefacts have been discussed in detail in Chapter 6. Therefore, this section will skip the details and focus on these artefacts' compatibility regarding the requirements engineering phase.

Practice 2 of the IEC 62443-4-1 provides a set of guidelines for the security requirements specifications. It also demands to identify a process to set the security capabilities for the component as prescribed in IEC 62443-4-2. The Cyber Security Requirements Specification (CSRS) and IEC 62443-4-2 LPGs provide the means of graph-based requirements specifications, enabling storing the capabilities of a component as a target security level property. Therefore, graph specifications are compliant with IEC 62443-4-1 practices. The graph-based requirements repository is essentially an integration of LPG specifications. Therefore, it provides a supplementary practice of requirements specification within the realm of IEC 62443-4-1. Moreover, the aspects of the requirements repository also comply partially with the security management of IEC 62443-4-1. This includes the identification of requirements applicability since the repository also contains the requirements from the CSRS that may only be fulfilled by external third-party components, e.g. a firewall, antivirus or an intrusion detection system.

Requirement analysis and specification are the core activities of MSDL's SRE phase.

The inception of this phase is started with the establishment of the security requirements. ITSIS coincides with MSDL regarding requirements analysis and extraction by creating graphs from the CSRS and the security standard. In order to generate the graphs, a requirements analyst needs to go through a one-time process of analysing and identifying the applicable security requirements from the CSRS document and IEC 62443-4-2 document. Notably, assimilation and gathering security requirements from the standard is a laborious process (El Houssaïni, Maskani & Boutahar, 2021). It includes the identification of foundation requirements, component requirements and their requirement enhancements. It also deals with the derivation of further supporting standards for applying cryptographic methods and algorithms, fulfilling another MSDL guideline that demands cryptographic standards when implementing cryptographic algorithms. Hence, such a practice satisfies the MSDL's requirements phase.

CryptSDLC requires the derivation of the essential requirements for the base cryptographic services and translate them into formal or semi-formal specifications. Regarding this, the semi-formal specification of requirements property graphs maps to this prerequisite of the requirements phase. Moreover, the CryptSDLC's map to model approach is satisfied by the requirements repository that maps the path from system requirements to cryptographic primitives on the leaf nodes of the IEC 62443-4-2 LPG. The requirements repository also adheres to CryptSDLC's proposition of storing requirements to cryptographic primitives association in an object storage structure.

The traditional SDLC forms the basis of the diversified development methodologies and also the methodologies mentioned above. Therefore, ITSIS naturally complies with it. The LPG specifications and the following requirements repository hold to be the artefacts supporting its requirements phase.

7.2.2 IEC 61449 Application Design

The secure links design method for ICS applications is the primary yield of the ITSIS regarding the design phase as shown in *IEC-61499 Application Design* stage in Figure 7.1. A secure link is an arbiter construct that contributes to the secure-by-design approach for IEC 61499 applications. Secure link primarily targets secure communications; however, it also provides the means for securing data at rest. They also act like a bridge that directs the implementation of security primitives in the form of Function Block Networks (FBNs) while also linking the requirements from the repository.

Practice 3 of the IEC 62443-4-1 recommends the secure-by-design approach. Secure links comply with related design principles put forth in the standard. One of these principles includes the identification of the internal or external accessibility of an interface. In this regard, a secure link formally defines the capability of an interface concerning its connectivity to external components or devices (See definition 1 in Chapter 5). Chapter 5 also discusses the implications of selecting various design-time options in the form of FBNs, which is a part of recommendations of IEC 61499-4-1. The standard also recommends best practices for a secure-by-design approach, including proven secure components and design patterns. Secure links are considered secure design patterns since they have been formally defined, thoroughly implemented and tested.

MSDL design phase recommends establishing design specifications and the specification of cryptographic design requirements. The design specifications are central to secure links through specifying requirements from the repository and implementing function blocks networks using simple annotations over function block data links. Moreover, the annotations endorse the requirements from the repository linking them to address the specification of standard cryptographic primitives as a part of design specifications. However, the secure links do not cater to a vital threat modelling activity

of MSDL. Threat modelling is a complex process and usually part of the design phase. However, it must be applied to the entire development life-cycle, not only in the design phase (Køien, 2020). Therefore, the researcher believes that threat modelling can be carried out separately from IEC 61499 application design and fed into secure links specifications as design constraints. The output of threat modelling can be deduced to use with secure link specifications. For example, a threat such as a man-in-the-middle attack can be mitigated using appropriate security methods or a combination of cryptographic algorithms specified as part of a secure link.

The design phase of CryptSDLC contain practices such as "prove security" and "deploy tool". It recommends that the design tools be built using formal methods and deployment of these tools to fulfil security requirements. As discussed in Chapter 5, secure links contain formally specified components such as annotations, S-Lib security library and FBNs. The structure of annotations that includes security requirement id, secure link id, and an FBN implementation of the security requirement provides a precise way to link requirements and the implementation phase. Thus, the secure link compiler partially supports complying with CryptSDLC's design phase to prepare for the deployment of the function blocks. At the same time, the S-Lib security library finally provides the means of deployment of function blocks on various devices.

Secure links consequently support the traditional SDLC design phase that translates the requirements into the secure-by-design paradigm and subsequently providing means to trace the security requirements between the implementation and requirements phase.

7.2.3 IEC 61499 Application Implementation

The primary artefacts of the implementation of ITSIS are secure links compiler and security library as defined and discussed in Chapter 5 and illustrated in *IEC 61499 Application Implementation* stage in Figure 7.1. The compiler transforms a secure link

into FBNs meant to be deployed on different devices in ICS. At the same time, the S-Lib provides a set of function blocks networks containing function blocks that implement cryptographic transforms.

Practice 4 of the IEC 62443-4-1 recommends secure implementation. A guideline of the standard deals with the traceability of the implementation with the design phase. Using the secure links and the S-Lib ensures the traceability of requirements between the design and implementation phases, e.g. a secure link identifier and an FBN in the annotation form a traceable link.

The implementation phase of MSDL recommends the use of approved tools during the secure implementation. The ITSIS implementation phase artefacts are based on the IEC 61499 standard for ICS application development. The standard provides a set of well-defined function block abstractions. It is rapidly gaining acceptability amongst academia, and the practitioners as a de-facto standard for ICS distributed application development (Lyu & Brennan, 2020; Cabadini et al., 2019). It is also well supported by extensive integrated development environments such as open-source 4diac IDE accompanying 4diac FORTE runtime and the propriety NxtStudio. Such environments also provide the possibility of inherent static code analysis and unit testing techniques. Therefore, the tool support and the rigorous semantics of IEC 61499 support the MSDL recommendation of using approved tools in the secure implementation phase.

CryptSDLC is also similar to MSDL secure implementation phase such that it demands the formally defined tools for implementation and deployment for cryptographic application development. S-Lib complies with this practice, as discussed previously. Furthermore, the S-Lib used in conjunction with secure links and the repository is also helpful in identifying the capability of the service provided by a component. Such phenomenon is compliant with the CryptSDLC's architectural property called "extract capabilities".

The major practices of SDLC's implementation phase include implementation and

deployment of the component or system. The IEC 61499 provides the semantics of the deployment of an ICS application across multiple devices. The secure links compiler also provides the convenience of deploying an FBN specified by the secure link across multiple devices using the IEC 61499 distribution model.

7.2.4 Traceability via TORUS — Verification and Validation

ITSIS provides the traceability support using TORUS splices discussed in Chapter 5 and 6. Figure 7.1 illustrate's TORUS's span across *Specification* and *IEC 61499 Application Design* stages since it is used by the requirements repository and the secure links. Requirements traceability is an integral part of the verification and validation (V & V) process since it can link various test types to the base security requirements. The types of testing include but not limited to, are unit testing, integration, system testing.

Tool support for testing is prevalent in popular IEC 61499 development environments and run-times such as 4diac IDE and 4diac FORTE runtime. For example, it supports a unit testing framework in the form of *FBTester*. In addition, manual integration testing methods can also be developed for function block and their networks using the IEC 61499 compliant environments.

ITSIS contains a chain of tools for the verification and validation of security requirements supported by the testing frameworks in the development environments and TORUS. Support for V & V of the implementation is achieved by running tests on function blocks that implement security primitives. Each function block under testing is a part of an FBN implementing a security method for a particular security goal. Secure links contain these networks as a part of the specification. Testing an FBN, therefore, implies V & V of a secure link. A TORUS splice also contains the secure link and requirement identifiers to maintain the end-to-end traceability as discussed in Chapter 6. Thus, TORUS helps to verify and validate a security requirement by linking tests to

the requirements. Figure 7.2 illustrates an example of a two-way process for achieving requirements verification and validation using the ITSIS artefacts.

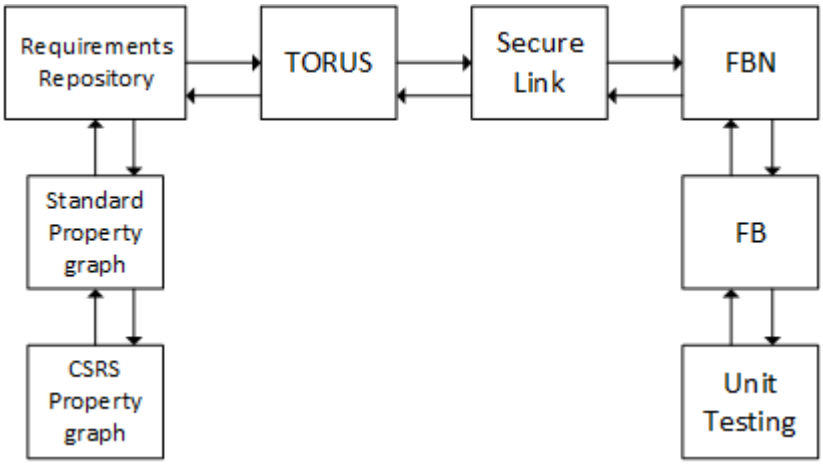

Figure 7.2: ITSIS verification and validation approach.

Practice 5 of the IEC 62443-4-1 provides guidelines for the verification and validation phase. ITSIS provides the ability of security requirements testing that is the foremost recommendation of the standard. On the other hand, it also recommends various testing types such as threat mitigation, penetration and vulnerability testing. However, the research considers such testings as independent procedures to be carried after the component is deployed over ICS (Yi & Kim, 2021).

The main focus of MSDL is on dynamic analysis and fuzz testing. ITSIS does not attend such kinds of testings predominantly. However, there is a possibility of using dynamic analysis approaches such as one used in (B. Dowdeswell and R. Sinha and S. G. MacDonell, 2020). The research discusses a technique for real-time testing of the IEC 61499 function block application by proposing a fault diagnosis engine.

CryptSDLC does not mention a verification and validation phase specifically. Therefore the researcher considers it as a partial match for the ITSIS for this phase. Nevertheless, being an umbrella methodology, SDLC is fully conformed by the proposed methodology that supports a subset of testing techniques recommended by SDLC.

The above analysis of ITSIS with the referenced methodologies shows that it is

compatible with major phases for ICS security application development. It allows the stakeholders to remain relevant with universal system development approaches to produce state-of-the-art certifiable security applications despite using novel and advanced techniques in an individual phase.

7.3 Results Synopsis

This section summarises some of the significant results relative to the application of ITSIS artefacts through this research. The research aims to re-iterate and emphasise the most effective elements of the proposed methodology in the following:

7.3.1 System Complexity

The utility of secure links can be determined by comparing the IMCS implementation with secure links in Figure 5.4 and the automatic compilation of secure links into FBNs implementing security mechanisms in Figure 5.6. This research quantitatively measures the efficacy of secure links by the use of well-established metrics.

Over the years, several metrics have been devised for measuring software quality at the design and implementation level, e.g. structural complexity, program length or reusability, maintainability. However, IEC 61499 applications are highly modular. FBs consist of several components such as data and event input/output interfaces, algorithms and Execution Control Charts (ECC) containing state machine and associated algorithms. Therefore, usual metrics can not be applied straightforwardly on an IEC 61499 application. Hence, in order to enable the measurement of the design and program complexity of such an application, (Zhabelova & Vyatkin, 2015) modifies McCabe's complexity and Halstead's software metrics to suit FB-based applications.

Measuring the complexity of IEC 61499 applications requires adapted measures that provide more accurate results than generic measures like program length. Specifically,

we calculate the following three measures, adapted for IEC 61499 and presented in (Zhabelova & Vyatkin, 2015), to characterize the complexity of a function block i:

- Structural complexity $S(i) = f_{out}^2(i)$ where f_{out} is the number of outgoing connections or afferent coupling.
- Data complexity $DC(i) = [NI + NO]/[f_{out}^2(i) + 1]$ where NI and NO are the numbers of data inputs and outputs, respectively.
- System Complexity $C(i) = S(i) + DC(i)$.

Results show a slight increase in system complexity when IMCS and BHS are implemented using secure links compared to implementation without secure links. For example, system complexity for IMCS rises to 49.6 from 37.6 with secure links, which is an expected overhead.

7.3.2 Scalability

Secure links are applied on an IEC 61499 Baggage Handling System (BHS) application to demonstrate the effectiveness of secure links at a larger scale. BHS handles the merger of bags travelling on connected conveyor belts. Each conveyor belt has a set of photoeyes that signal to the adjacent belts according to the traffic ahead. The BHS application two main types of Composite Function Blocks (CFB), i.e. `TwoConveyor` that controls two connected belts and `ThreeConveyor` that manages a set of three conveyor belts. In the current BHS application, there are four `TwoConveyor` function blocks that send two critical parameters such as outgoing *Bag id* and *Bag length*, to the `ThreeConveyor` function block. All five function blocks are deployed to physically separated devices. Different combinations of secure links are applied on the function block links that send *Bag id* and *Bag length* to the `ThreeConveyor` function block. Subsequently, the design and program complexities of BHS application pre and post

compilation of secure links were calculated. The original $C(i)$ of the BHS application with secure link annotations is 233. However, post-compilation $C(i)$ increases due to the surge of additional security-related function blocks in the application, as can be observed in Figure 5.8 (b).

The results show that secure links scale well regarding system complexity when the number of secure links are increased. Moreover, a rise in complexity can be seen when (more) security mechanisms are added manually.

7.3.3 Maintainability of IEC 61499 Applications

To demonstrate the effectiveness of secure links for program level complexities, we use Halstead (M_H) and McCabe's (M_M) complexity metrics from (Zhabelova & Vyatkin, 2015) that are adjusted for IEC 61499 function blocks. M_H contains multiple metrics that measure different aspects of program complexity. For example, measure of program length is given by $N = N_1 + N_2$ where N is total number of operators and operands. Program vocabulary is given by $n = n_1 + n_2$ where n total number of distinct operators and operands. Estimated length metric can be calculated by $\hat{N} = n_1 log_2 n_1 + n_2 log_2 n_2$. Purity Ratio is $PR = \frac{\hat{N}}{N}$. Program volume is given by $V = N log_2 n$.

Program difficulty D is computed as

$$D(fbn) = \frac{\sum_{i=1}^{n} D(fb_i)}{n}; D(fb) = \sum_{i=1}^{n} D(alg) + D(cf)$$

$D(fbn)$ and $D(fb)$ are the program difficulty of an application and a constituent function block, respectively. Similarly, McCabe's cyclomatic complexity metric M_M is calculated as:

$$M_M(fbn) = \frac{\sum_{i=1}^{n} M_M(fb_i)}{n}; M_M(fb) = \sum_{i=1}^{n} V(alg) + V(cf)$$

$M_M(fbn)$ and $M_M(fb)$ represent the cyclomatic complexity of an application and

a constituent function block, respectively. $M_M(fb)$ calculations use the value V which is the sum of the cyclomatic complexity of algorithms $V(alg)$ and the cyclomatic algorithms of the control flow $V(cf)$ in a function block. $M_M(fbn)$ is simply the average of all $M_M(fb)$ values in the network.

The application MI in that Figure 5.10 shows that the maintainability of the application stays the same with n amount of secure links annotations. However, the actual maintainability of the application drastically decreases with each secure link that gets compiled into an FBN. The BHS was also tested on multiple security levels provided by IEC 62443-4-2 (CR 3.4). SHA can achieve security level 1 by providing an authentication mechanism. Level 2 for the requirement CR 3.4 can be achieved by implementing integrity and authentication mechanism such as HMAC. After trying multiple combinations, it was observed that lower security levels affect MI to a lesser extent. It is due to a fewer number of operations performed by SHA as compared to HMAC or AES. However, it also depends on the implementation and the choice of mechanisms selected for a particular security level. Section 5.7 of Chapter 5 has more insights into the alleviation of maintainability and program difficulty of IEC 61499 applications using secure links.

7.3.4 Requirements Traceability via TORUS

A comprehensive traceability matrix is obtained by applying and evaluating the security requirements repository on a wind turbine case study. It shows that the TORUS can help bridge security requirements in the IEC 62443 security standard and their IEC 61499 implementations.

Requirements Traceability Matrix (RTM) essentially provides a map of relationships between requirements and the product artefacts such as design, implementation and tests. Table 6.4 shows resulting security RTM for the wind turbine system that can be

readily generated by expanding a TORUS splice from the containing requirement and secure link identifier. It can be done by following the steps from the proposed method in Section 6.9 i.e. generate the subgraph by combining CSRS and IEC 62443-4-2 graph through Cypher queries based on a requirement identifier. Each row of the table represents the result of a TORUS splice expansion.

The overall results show that ITSIS artefacts provide improved novel techniques for ICS development. It has the potential to enhance the ability of the developers and designers to produce maintainable and scalable security applications for safety-critical environments. Moreover, it supports requirements management and traceability to suit multiple stakeholders' needs in the highly dynamic process of security certifications.

7.4 Implications on Security Goals

This section discusses this research's work to achieve primary security goals for IEC 61499 applications. Overall, the research followed a bottom-up approach to fill the gaps in distributed ICS application domains. The first step of the approach was to break down the problem area according to the security goals such as confidentiality, integrity, and availability. Therefore, each iteration of the adopted design science research methodology improved the knowledge base and the researcher's understating of the challenges of a specific security goal. The following sections discuss implications and the limitations of the by-products of each iteration in chronological order.

7.4.1 Confidentiality

The distribution model of IEC 61499 makes confidentiality of the data as the most desired feature. Distribution of application function blocks on multiple devices is core to the concept of IEC 61499. The deployment model is intuitive and intrinsic, which can covertly induce security issues if the designers do not consider threat mitigation solutions

(Homay & de Sousa, 2016; Staggs et al., 2017). For example, two function blocks of an IEC 61499 application may get deployed on two different devices physically apart from each other. In such scenarios, an attacker can gain an advantage in the absence of confidential data communication between the two devices.

The contributions in Chapter 3 aims to solve the data communication issues in IEC 61499 function block applications. The proposed solution provides the specification and implementation of encryption function blocks based on the variants of the Advanced Encryption Standard (AES). Therefore, it creates a secure tunnel between the distributed function blocks. The results have shown that confidentiality can be achieved using AES in resource-constrained ICS devices by meeting the latency threshold for smart grid protection functions. The properties of the proposed security layer and their implications are discussed below.

7.4.1.1 Abstract Function Block Interface

The design of the security layer is such that the function blocks only acts as abstractions. Although the experiments demonstrate the applicability of viable encryption in IEC 61449 applications using AES and its variants, the function block interfaces allow for the implementation of cyphers other than the AES. The function block's Execution Control Chart (ECC) can contain the implementation of other similar AES finalists such as MARS, RC6, Serpent, and Twofish. Modern run-times like 4diac FORTE facilitate the implementation of a function block ECC in C/C++ programming languages. Therefore, the encryption algorithms may be implemented in low-level languages to boost the throughput of encrypted communication. The function blocks' data input and output interfaces make it possible to use them as black boxes. For example, an encryption function block (`Encrypt`) has three main data inputs such as plain text (`pt`), key size (`ksize`), and an expanded key (`expkey`). Such interfaces are familiar to the range of audience types, i.e. from a layperson to cryptographers. Therefore, it does not require

a designer or developer to be an expert in cryptography to use these function blocks. Similarly, encryption function block has a single data out (`ct`) that can be connected to the input of `Decrypt` function block.

Key expansion (`KeyExp`) function block is an independent function block. Key expansion is an expensive process in terms of computational resources. Therefore, it is important to isolate it from the actual encryption/decryption process. The advantages of such a design can be seen in situations when the one key is meant to be used for multiple function blocks. Although it is not a recommended scheme, a key can be expanded to be used with more than one function block in such a scenario. In the absence of an autonomous key expansion function block, every instance of an (`Encrypt`) block needs to expand the key, thus increasing the latency and inefficient use of computing power of already resource-constrained ICS devices.

7.4.1.2 Data-in-Transit Encryption

The solution in Chapter 3 is based on a publisher/subscriber model. The publisher sends the cypher text, and the subscriber receives the cypher text to decrypt it back into the plain text. Using such a model reduces the complexity of creating new communication layers for the data-in-transit.

The secure distribution of an encryption key is one of the key issues in secure data communications. The solution provides (`DHInitiator`) function block that is based on the Diffie-Hellman key exchange algorithm. Thus, it solves the secure transfer of the key between the two devices before using it for encryption/decryption. Moreover, it is also an independent procedure that can be used to distribute keys asynchronously, i.e. the key distribution and the use of key can happen at different times.

Latency is also one of the major issues for secure communications. It is not always feasible to use traditional cyphers in resource-constrained ICS devices (Mosteiro-Sanchez, Barcelo, Astorga & Urbieta, 2020). Therefore, the usage of lightweight

encryption algorithms makes a good use case in such environments. Chapter 5 shows the application of traditional and lightweight methods in the IEC 61499 security application. The encryption/decryption function blocks in Figure 5.6 are model extension of the proposed security layer. Furthermore, the latency implications of using LibHydrogen (*LibHydrogen - Cryptographic library for constrained environments*, n.d.) as lightweight and traditional TLS is shown in Table 5.3. Traditional encryption approaches are usually more secure, and they can be used in scenarios where an ICS device needs to send the critical data across the zones in ICS environments. For example, a master PLC can send critical monitoring parameters to the SCADA server. In such cases, high latency may be tolerated. On the other hand, real-time operations, e.g. communication between slave PLCs to control the actuators in relatively confined environments, require minimum latency. Therefore, there is a use case to use lightweight cryptographic algorithms in such scenarios.

7.4.1.3 Data-at-Rest Encryption

IEC 62443-4-2 specifies the requirements to secure the local data residing on the ICS component. The use-cases may include the secure local storage of configuration of a device or storage of other critical parameters such as encryption key or boot parameters. The primary focus of the security layer discussed in Chapter 3 is to provide confidentiality for data-in-transit. However, the encryption/decryption function blocks may be used independently of the publisher/subscriber model to encrypt the data and store it on the device using file writing Service Interface Function Block (SIFB). Such function blocks are available in IEC 61499 compliant IDEs like 4diac IDE.

7.4.1.4 Limitations

7.4.1.4.1 Key Distribution

The key exchange (`DHInitiator`) function block is implemented as SIFB that distributes the key. However, this key distribution method's current implementation and design are at the proof-of-concept stage that is not scalable for multiple devices. Key distribution is a complicated problem where the complexity increases with the additional devices in the group. Moreover, Diffie-Hellman is a resource-intensive algorithm that is not well-suited to resource-constrained ICS devices. A lightweight key distribution model such as presented in (Harn, Hsu & Xia, 2021) may be used in future to cater for the key distribution constraints in the current IEC 61499 security layer.

7.4.1.4.2 Asymmetric Key Cryptography

Asymmetric key encryption algorithms are extremely resource-intensive relative to the ICS devices with modest computing power. The specifications and design of the proposed security layer incline towards the use of symmetric key cryptography. For example, interface of (`Encrypt`) and (`Decrypt`) function blocks do not deal with provision of public and private keys. Therefore, the possibilities of using asymmetric key techniques are not fully explored in the IEC 61499 domain. It provides an open field for future research, especially in the area of lightweight asymmetric key encryption.

7.4.2 Availability

One of the foremost concerns regarding ICS is the maximum availability of the services. It is especially true for highly safety-critical applications deployed in nuclear reactors, manufacturing systems, aeronautical systems. The fourth industrial revolution is connecting more and more ICS to the internet. Attackers can deploy advanced methods of sabotaging the system by attacking its availability using sophisticated denial-of-service

(DoS) attacks.

The contribution of this research regarding the availability of IEC 61449 based ICS is discussed in Chapter 4. The solutions list down some of the attacks and the scenarios that can impact the availability of the ICS. It implements Snort (*Snort - Network Intrusion Detection and Prevention System*, n.d.) — an open-source Intrusion Detection and Prevention System (IDPS) — in the form of IEC 61499 service interface function blocks. The experiments were carried out using Wago 750-8206 PFC200 PLCs using embedded real-time Linux controlling two cylinders. Successful DOS attacks were carried out in order to test the IDPS implementation. The results show that the snort IDPS based on SIFBs could detect the DOS attacks in real-time. The possible future implications and its limitations for the availability solution are presented below.

7.4.2.1 Machine Learning IDPS

The static rules or signature-based detection systems are not able to confront the evolving attack methods. Despite the availability of advanced protection techniques, attackers are also using advanced techniques such as distributed DOS or sophisticated malware attacks to reduce the availability of the services. To protect against such attacks, security administrators need to update the rules and signatures for the possible attacks constantly. However, these may not be sufficient measures to protect the safety-critical ICS against “zero-day” attacks where an attacker can exploit the vulnerabilities before they are fixed. Regarding availability-related attacks on ICS, a new attack pattern might bring down the system for a short period before security administrators can analyze the attacks and provide the fixes to counter the new attack patterns. The downtime caused by such a scenario may become unacceptable especially for real-time ICS performing safety-critical operations.

Active security protection, thus becomes imperative in order to automate monitoring of the attack patterns and apply mitigation measures in real-time. A possible approach

to applying such automation is to use anomaly-based IDPS using Machine Learning (ML) algorithms. The IEC 61499-based IDPS solution proposed in Chapter 4 can be modified to use ML techniques along with *snort*. One such solution is provided in (Shah et al., 2018) where the authors use ML plugin along with the *snort*. However, the applicability of such solutions is limited by the availability of computational power with resource-constrained devices in ICS.

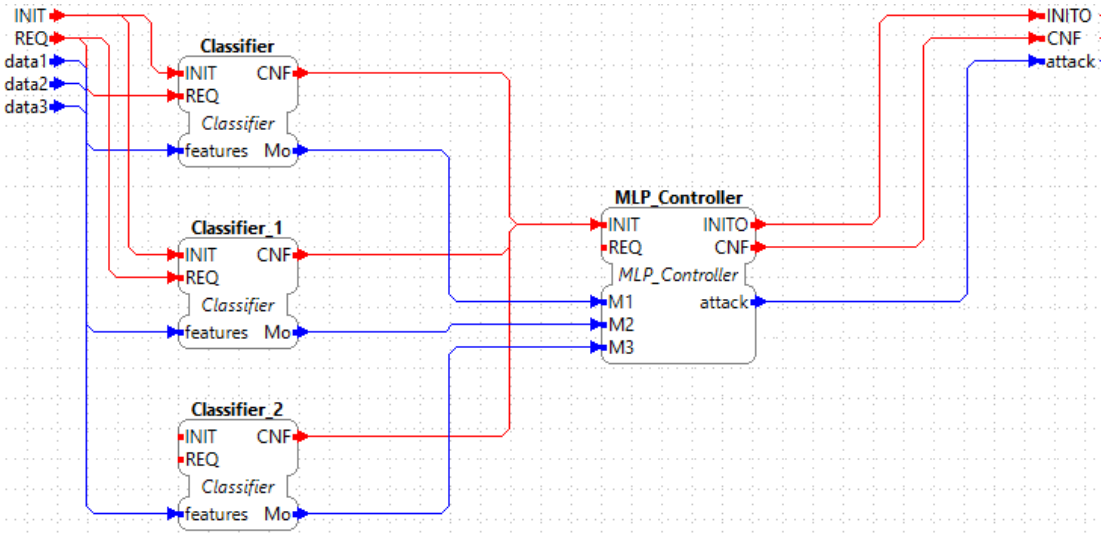

Figure 7.3: An ML-based IDPS proposal for IEC 61499 applications

ML-based solutions for resource-constrained devices have been proposed recently. A scalable Intrusion Detection System (IDS) based on ML approach for IoT devices is proposed in (Rahman et al., 2020). It builds the model with a high detection accuracy of 97.80% using the distributed approach by consuming 73.52s of CPU time. Such a solution can be deployed for distributed IEC 61499 applications since it distributes the classification of attacks on multiple nodes in the network. A possible solution is distributing the classification over multiple function blocks distributed over multiple nodes in a function block network. A single collector function block can be used to gather the data from nodes to decide whether the data corresponds to the attack, as shown in Figure 7.3. However, the ML-based solution uses AWID dataset specific to

impersonation attacks. For its implementation in IEC 61499 applications, a dataset-specific to ICS specific DoS/DDoS attacks is required to efficiently implement these techniques to detect the attacks with the similar (98.7%) or preferably lower accuracy of false positive/negative alarms.

Another suitable IDPS for IEC 61499 application based on ML technique is presented in (Amouri, Alaparthy & Morgera, 2020). The authors present a two-staged ML-based IDS for mobile IoT. The first stage is collecting data using dedicated sniffers and sending the correctly classified instances to a supernode in the network. The supernode uses a linear regression process to differentiate between rogue and normal nodes. The proposed IDS is successfully tested against DoS attacks which makes it compatible with our approach in Chapter 4. Therefore, a similar to Figure 7.3 IEC 61499 IDPS can be developed to realise this concept in ICS environments because of its suitability to resource-constrained devices. However, a limitation to this approach is a high number of false positives, i.e. between 1.3% to 12%.

7.4.2.2 Lightweight IDPS

Lightweight IDPS solutions are viable for the resource-constrained devices in ICS. Traditional IDPS generally have a high memory and computation footprint due to real-time computations while consistently matching the incoming requests against a large set of rules. Therefore, it is highly desirable to deploy independent lightweight solutions in safety-critical infrastructures to provide reasonable protection to a device while consuming minimum resources.

A lightweight IDPS for ICS is presented in (Jin, Valizadeh & van Dijk, 2018). The solution is an interesting work in the context of this research since it shows the applicability of lightweight IDPS on PLS devices. The authors use a forward secure logging technique on a Host IDS (HIDS) called “snapshotter”, deployed on each PLC in the network. The snapshotter agent constantly keeps a log of the security-related

events on a PLC. The logs are periodically sent to a trusted server that analyses the logs and their integrity and raises alarms if it finds discrepancies.

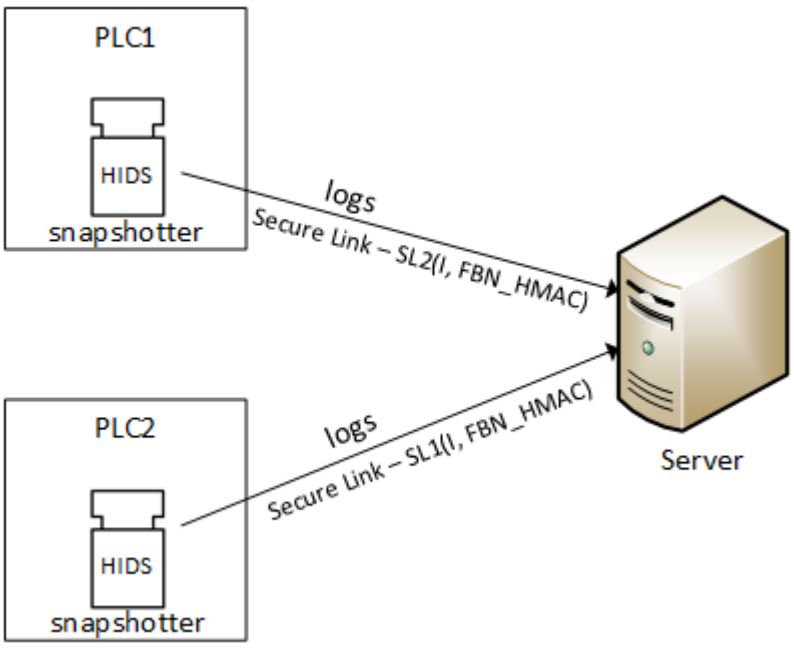

Figure 7.4: A conceptual model of IEC 61499 HIDS based on snapshotter

The method employed by the snappshotter seems to be a viable solution for IEC 61499 applications since it relieves the PLCs of the responsibility to detect the attacks since the trusted server in the network performs the resource-intensive tasks. Therefore, a lightweight HIDS can be implemented as a service interface function block that can log the event on the PLC devices and periodically send it to an external server. The IEC 61499 based HIDS can take advantage of the secure links to send the logs securely over the network. A conceptual model for such an approach is shown in Figure 7.4.

7.4.2.3 Limitations

Snort IDPS proposed in Chapter 4 is not always suitable for ICS applications. The primary purpose of implementing a proof-of-concept snort IDPS in IEC 61499 applications is to determine the feasibility of IDPS in order to protect against availability

attacks. Although the solution is successfully implemented and tested, *Snort* has a high memory footprint. At the same time, it does not perform well in a high-speed network that could restrict its capability to detect and prevent attacks in real-time systems efficiently (Shah & Issac, 2018). Therefore, it is recommended to use lightweight IDPS solutions as discussed in Section 7.4.2.2.

Moreover, it is also not always advisable to deploy IDPS in confined environments with resource-constrained devices. It is because the IDPSs are generally considered as the last line of defence in a network. However, even the most sophisticated IDPS may also fail to detect an attack, especially zero-day vulnerabilities that are attacked for the first time (Ylmaz et al., 2018). An IDPS is usually configured at the network boundary, as shown in Figure 4.4, to help deploy it on a dedicated resource-rich machine that can block the attackers from penetrating the internal ICS network. It can be assumed that an IEC 61499 IDPS deployed on a resource-constrained device will not be effective against such sophisticated attacks if an attacker can infiltrate into ICS internal network nevertheless.

Devices controlling the physical operations in ICS have an essential task to respond to real-time events. They are dedicated to real-time operations. Therefore, it is not always feasible to add function blocks performing resource-intensive intrusion detection and the function blocks that implement the core operations for that particular device. As a result, the deployment of devices dedicated to containing IEC 61499 SIFB IDPS may prove counterproductive in terms of project cost and network complexity.

7.4.3 Integrity

Communications within the ICS domain require data communication integrity between different layers, e.g. field devices → control devices → SCADA. In IEC 61499 distributed applications, it is desirable to ensure the integrity of the critical data transferred

between two function blocks deployed on multiple devices. Attacks such as man-in-the-middle, session hijacking may break the trust between the devices that may result in unintended operations from the compromised device.

Chapter 5 shows the implementation of IEC 61499 function blocks providing data integrity. The secure links concept makes use of such function blocks in order to communicate the critical parameters within the controller devices network and also to external devices such as HMI using open connections defined in Definition 1.

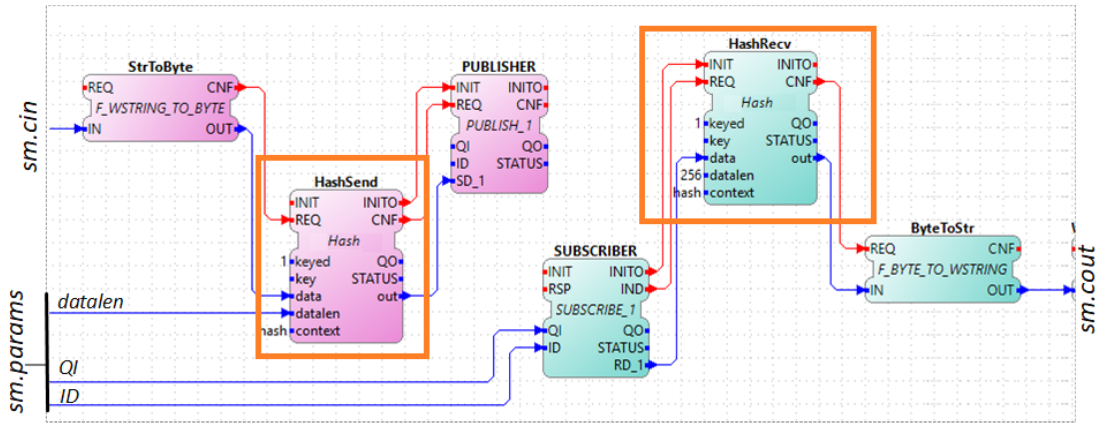

Figure 7.5: IEC 61499 keyed-hash mechanism providing data integrity (Copied from Figure 5.5)

Figure 7.5 shows an IEC 614999 implementation of keyed-hashing or HMAC function block network (highlighted in orange) based on publisher/subscriber model. Function block `HashSend` is used to calculate the hash of the input data and output it as a string of bytes. The publisher function is used to send the data over the network. The receiver's subscriber block receives the hashed data and passes it onto `HashRecv` function block that uses a pre-established symmetric key to ensure the integrity of the hash. The implications of using such a security mechanism in IEC 61499 are discussed below.

7.4.3.1 Abstract Function Block Interface

The interface of `hash` function block allows it to behave like an abstraction independent of hash algorithm implementation. The ECC may contain implementation of the hash based on the selection of a developer respective to security requirements. The `key` data input is an optional parameter that allows the function block to behave in multiple ways. For example, the function block provides a simple hashing facility if `keyed` data input is set to true, and `key` is kept to null. Otherwise, it operates like an HMAC mechanism. The `data` and `out` interfaces are byte types that allows the user of this function block to pass or receive data independent of non-primitive data types. Moreover, the `hash` is a basic function block that can take advantage of ECC for flexible language implementations of the hash function, unlike service interface function blocks that are dependent on the underlying platforms and run-times.

7.4.3.2 Open-ended Secure Communication

The `hash` function block can be used as a one-way communication block. Its usage as an open-ended transmitter is shown in Definition 1. Its subsequent use in the secure links is illustrated in IMCS case study in Figure 5.4 and a detailed implementation is further shown in Figure 5.6. In a scenario where the recipient of the data is an external application other than a function block, the hashed data can be transmitted over the network using a publisher. The implementations shown in Chapter 5 cover the data transmission use-cases. However, data can similarly be received from an external application using a subscriber in order to subsequently process it using `hash` function block. For example, a PLC device can transmit or receive the data to and from an HMI device in SCADA using `hash`'s HMAC transformation to ensure data integrity and authentication. The symmetric key can be independently distributed to both entities using the key distribution method discussed in Section 7.4.1.2.

7.4.3.3 Lightweight Hash Functions

Similar to other security goals, lightweight cryptography can also be beneficial to protect the data integrity in ICS resource-constrained devices. As discussed previously, a significant advantage of using an abstract function block interface is the ability to implement various hash algorithms. The researcher has discussed and compared the lightweight and traditional approaches in Chapter 5 in detail. The implementations of data integrity mechanisms in IEC 61499 include TLS-based implementation using traditional hash algorithms and a lightweight keyed-hash function provided by LibHydrogen (*LibHydrogen - Cryptographic library for constrained environments*, n.d.) library. Table 5.3 shows an improvement while using lightweight integrity and authentication (1.87ms) as compared to TLS (3.03ms).

7.4.3.4 Limitations and Future Works

The research provides a limited implementation and comparison of using traditional cryptography primitives for data integrity. Only the feasibility of TLS in IEC 61499 applications are discussed and shown. Although TLS is recommended to secure the data communication in IT and ICS environments (Mosteiro-Sanchez et al., 2020) because of the availability of its robust implementations, TLS is inherently not well-suited for resource-constrained devices due to its security-performance tradeoff (Hogan et al., 2018). Therefore, non-TLS implementations and comparisons need to be performed in the future in order to substantiate the recommendations of avoiding traditional cryptography for IEC 61499 applications. For example, comparisons can be obtained by implementing hash functions recommended by ISO/IEC 10118-3 using a security mechanism illustrated in Figure 7.5. Modern hashing algorithms like SHA-3 family functions are well-suited to provide parallelism for hash functions (Kelsey, Chang & Perlner, 2016). Therefore, it will be an interesting endeavour to benchmark their

performance in comparison to lightweight cryptography.

The LibHydrogen library used for lightweight implementation of a hash function is not specifically designed for ICS. Although the use of LibHydrogen has been helpful to reduce data hashing latency in IEC 61499 applications, domain-specific algorithms need to be tested and bench-marked in the future to provide a comprehensive account of the feasibility of using lightweight primitives in IEC 61499 applications. The researcher suggests a non-exhaustive list of state-of-the-art lightweight hash functions from ICS and related domains in Table 7.2 that provide an underlying implementation to `hash` function block.

Table 7.2: Lightweight hash functions suitable for IEC 61499 implementation

Algorithm	**Domain**	**Research**
LightBC	IoT	(Pohrmen & Saha, 2021)
Chaskey	Microcontrollers	(Mouha et al., 2014)
Quark	Generic	(Aumasson, Henzen, Meier & Naya-Plasencia, 2013)
AEchain	IoT, Embedded Systems, Blockchain	(S. Khan, Lee & Hwang, 2021)

7.5 S-Lib: Security Function Block Library

In this section, the researcher discusses the security library consisting of IEC 61499 function blocks to help achieve security goals. The IEC 61499 function blocks enabling the implementation of cryptography primitives in Section 7.4 are assembled in the form of library called *S-Lib*. The concept of the security library for IEC 61499 is first discussed in Chapter 3. The concept is advanced in Chapter 4 by providing the implementation of an IEC 61499 IDPS. It is further consolidated and formally defined in Chapter 5 as an accessory to the secure-by-design approach of Secure Links for more practical purposes. However, the researcher proposes it as a standalone component for providing secure applications for IEC 61499-based ICS.

7.5.0.1 Usage in Secure Links

Definition 2 in Section 5.5.2 describes the security library as a finite set of security mechanisms. These are essentially Function Block Networks (FBN) providing the secure communication of the data processed by underlying primitive function blocks. A security mechanism has defined inputs and outputs with underlying implementations of cryptography functions and algorithms. The design of the mechanism allows the containing function blocks to be mapped on two separate devices, which is essential when the same IEC 61499 application is mapped on the transmitter and recipient devices. Moreover, a security mechanism may provide open-ended communications, i.e. it may act as a transmitter or a receiver only if the target or initiator is an external application. The secure link compiler discussed in Section 5.6 provides the ability to translate a secure link and deploy the associated mechanisms on multiple devices. Furthermore, a security mechanism may have one more utility function block to assist the implementation. For Example, `StrToByte` and `ByteToStr` functions blocks in Figure 7.5 provide the data conversion facility to the main `hash` cryptography primitive function block.

Multiple security mechanisms are implemented and tested in order to evaluate the concept of secure links. Table 5.1 shows the design complexities of security mechanisms providing the implementations for key exchange, Authenticated encryption with associated data (AEAD), and Hash/HMAC cryptography methods. These mechanisms further contain primitive function blocks for cryptography services. Table 5.2 further shows the program complexity and maintainability index of cryptographic function blocks supporting the implementation of security mechanisms. Two primitives are implemented in this research, i.e. lightweight algorithms from LibHydgrogen and the traditional algorithms supported by TLS.

The 4diac IDE and 4diac FORTE run-time platforms are used to implement the

cryptographic primitive function blocks. The 4diac FORTE run-time provides basic function blocks that allow the ECC to be implemented in IEC 61131 programming languages (Ramanathan, 2014). However, externally imported implementations that are not based on IEC 61131 programming languages need Service Interface Function Blocks (SIFB). Since LibHydrogen and TLS are implemented in C programming language, therefore, the primitives listed in Table 5.2 are implemented as SIFBs in 4diac FORTE run-time. The run-time allows the C methods from these libraries to be implemented as C++ wrapper classes.

The IEC 61499 implementations of security mechanisms act as FBNs, and cryptographic primitives are implemented as function blocks, providing a platform to consolidate the concept of a novel IEC 61499 security function block library. The design of the library and its potential, implications, future uses, along with its limitations, are discussed in the following.

7.5.1 S-Lib Architectural View

Figure 7.6 illustrates a logical view of S-Lib components. The architecture diagram shows the interactions between S-Lib's internal components and peripherals, such as secure links compiler and generic IEC 61499 utility function blocks.

The cryptographic core is the pivotal component of the library. It is essentially a collection of function blocks providing the implementations of cryptographic algorithms. It contains two types of implementations:

1. *Basic function blocks* that implement a cryptography transform within its ECC using one of the IEC 61131 programming languages. One of the widely used languages is Structured Text (ST). Such function blocks provide the flexibility and independence of the implementation without the dependence on an external cryptography library. One such example of this kind of function block can be seen

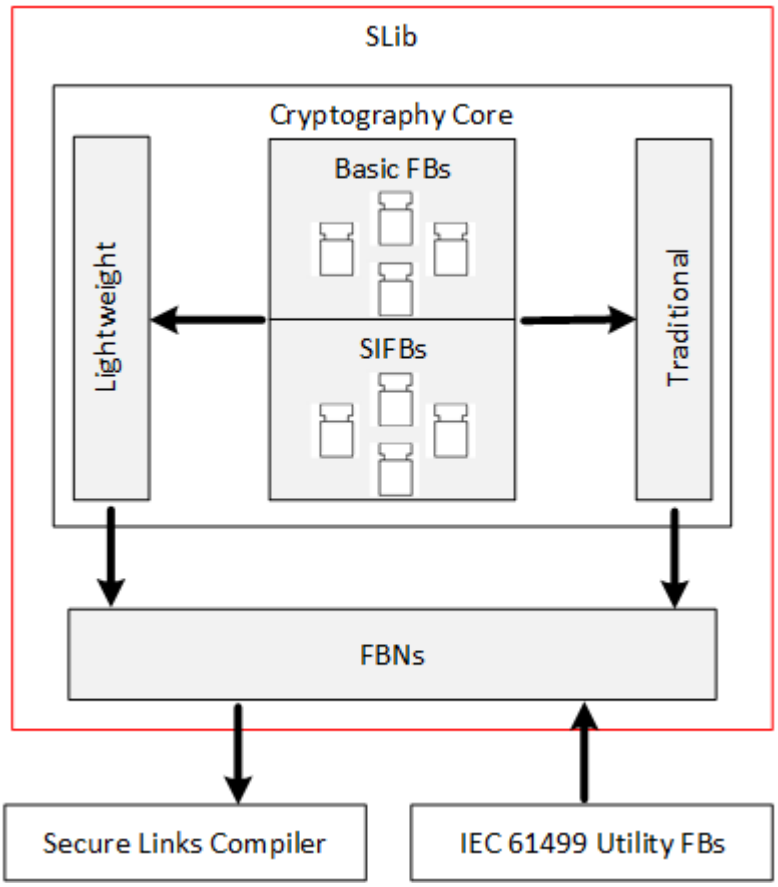

Figure 7.6: IEC 61499 security library architecture

in Figure 3.9 where `AESEncrypt` and `AESKeyExp` function blocks implement the respective algorithm using ST language.

2. *SIFBs* carry out the algorithm implementations that are exported from the third-party cryptographic libraries. Use of such external libraries is discussed in Section 7.5.0.1. Performance is another reason to use such types of function blocks. Security and performance are mutually exclusive with tradeoffs due to computationally intensive cryptographic operations. High-level languages such as ST are not always suitable for resource-constrained devices due to their underlying execution architecture. Real-time applications require minimum latency in order to perform critical tasks. Lower level languages such as C/C++ provide a much-desired performance boost for cryptographic transforms due to their proximity to

the machine hardware. Therefore, resource-intensive cryptographic algorithms can be implemented in a lower-level language using SIFBs when performance and latency are foremost concerns. The function blocks listed in Table 5.2 are the prime examples of such IEC 61499 implementations.

The core of the security library is further divided into lightweight and traditional implementations of cryptographic primitives. It can be considered only as a logical partition. The basic function blocks and SIFBs collection contain both type of implementations for cryptography method where possible. Thus, the library user can select either a traditional or lightweight algorithm depending on the security requirements.

FBNs allow the practical use of core cryptographic function blocks. Therefore, the FBN component of the library uses the cryptographic primitives from the core of the library to form the security mechanisms. These networks also require utility function blocks that handle and prepare the data produced by core function blocks types for further uses. For example, the `PUBLISHER` function block in Figure 7.5 transmits the hash produced by `Hash` block over the network. Although primitive function blocks, if implemented as SIFBs, may also provide such functionality, it is not recommended due to the separation-of-concerns design principle. Moreover, secure links compiler process the FBNs to transform them for the distributed deployment. The deployment devices are chosen based on a secure link's endpoints.

The researcher suggests the *S-Lib* security library as an independent component although it has been developed as a part of the Secure Link Development Method (SLDM). Figure 7.7 shows *S-Lib* (highlighted) in the scheme of SLDM. It can be seen that the *S-Lib* is used to provide implementations of secure mechanisms (FBNs in the context of *S-Lib*) to the secure link compiler, which transforms into an extended network of function blocks suitable for the distributed deployment. The compiler only acts as an external tool to help realise a secure link. However, the distribution of an FBN can be

done manually by the developer using the mechanisms provided by IDEs and run-times without the reliance on tools like secure links compiler.

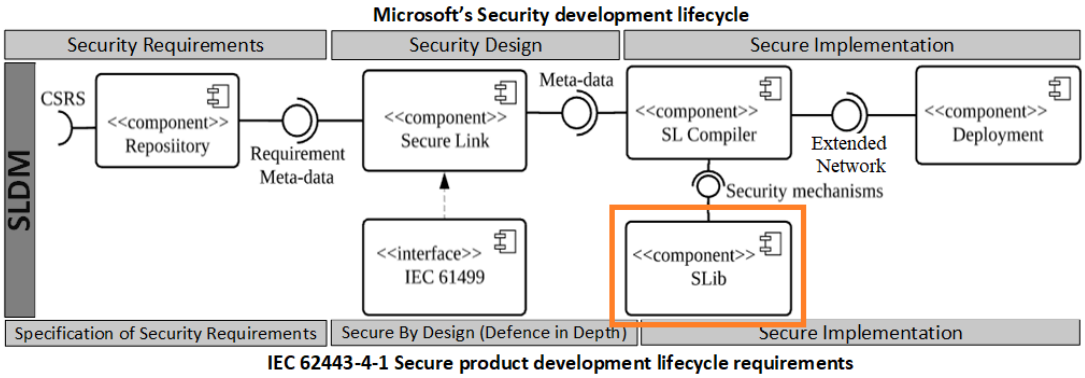

Figure 7.7: An overview of the SLDM (copied from Figure 5.2)

7.5.2 S-Lib: An Open Source IEC 61499 Crypto Library

This section proposes *S-Lib* as an open-source crypto library and discuss its implications for the broader community of ICS developers using the IEC 61449 development model in particular.

7.5.2.1 Integration with Open Source IDEs and Runtimes

IEC 61499 provides an interoperable XML format to represent function block types. Basic function blocks containing ECC implemented in IEC 61131 supported languages can be directly imported into the IDEs that are compatible with IEC 61499 standard. Therefore, *S-Lib* cryptographic primitives using such function block can take advantage of this interoperability to provide platform-independent cryptographic services.

This research has mainly used 4diac IDE and 4diac FORTE runtime for the proof-of-concepts. Both the tools are an IEC 61499 compatible open-source development initiative. The 4diac IDE is an Eclipse-based IDE developed in Java programming language while 4diac FORTE is the IEC 61499 developed in C++ using object-oriented approach. 4diac FORTE in particular, can be deployed on different POSIX compatible

operating systems. The runtime transforms the XML specification of function blocks into compiled C++ classes during the execution. Therefore, *S-Lib* function blocks are also implemented as C++ classes especially the SIFBs. Using such runtime also makes *S-Lib* capable of executing across different platforms and operating systems. Since 4diac FORTE is an open-source project, it also influences *S-Lib* to become an open-source library.

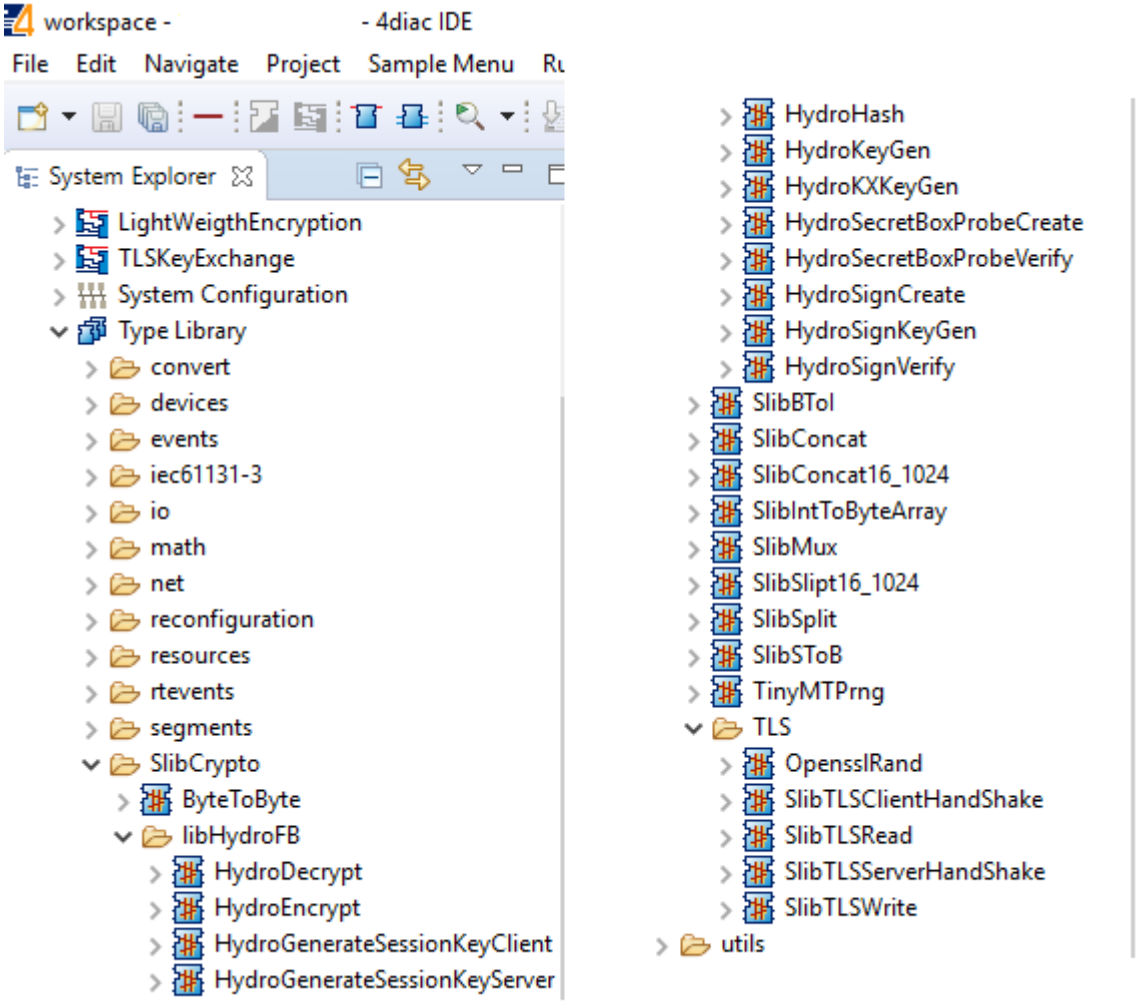

Figure 7.8: Current S-Lib implementation in 4diac IDE and 4diac FORTE

Figure 7.8 shows the screen capture of the current implementation of *S-Lib* achieved during the course of this research. The *S-Lib* cryptographic function blocks are included in the type library of 4diac IDE and 4diac FORTE. They are implemented as SIFBs

since they use external libraries such as LibHydrogen and TLS. The 4diac FORTE runtime contains their C++ implementation.

7.5.2.2 Community Collaboration

Community contribution can propel *S-Lib* for its advanced use in IEC 61499 applications. The concept of *S-Lib* is novel to the best of the researcher's knowledge; however, it is in its prototype stage. It can emerge as a de-facto cryptography library for IEC 61499 through the wider collaboration of security researchers and developers. The current footprint of the library is not very extensive for its more practical uses. The current implementation contains a small number of cryptography services perceived as prototypes to carry out this research. Cryptography is a broad and active research area with frequent new contributions deprecating the older techniques. Similar is the case with the ICS domain due to the emergence of industry 4.0 security challenges. Therefore, it is not always possible for a small group of researchers and developers to maintain and sustain a cryptographic library for an emerging ICS IEC 61499 development standard. Community collaboration ensures that *S-Lib* is kept to the state-of-the-art with the implementations of diversified cryptographic primitives. Moreover, the current architecture of *S-Lib* is limited to the awareness of the researcher and the few experts. The evaluation is limited to the case studies used in this research. Therefore, community-wide collaboration has the potential to improve the architecture and design of the *S-Lib* to make it more robust and maintainable by bringing in the solutions developed and tested by the community.

7.5.2.3 Scalability

7.5.2.3.1 Load Scalability

Scalability of *S-Lib* can also be achieved through community collaboration. The distributed approach of IEC 61499 means that the security mechanisms in the *S-Lib* need to be able to scale well in the case of large-scale distribution of function blocks. The current implementation of lightweight cryptographic primitives can scale with the increasing number of nodes through the use of secure links that have shown scalability properties (See Section 5.7). However, as an example, the key exchange mechanisms in *S-Lib* are fundamental, e.g. lightweight key exchange mechanism implemented in Figure 5.6 is restricted to the key distribution for up to two devices. Moreover, key distribution is a significant challenge for security applications when multiple nodes are involved, as discussed in Section 7.4.1.4.1. Therefore, key distribution in *S-Lib* needs to be scalable to match the potential challenges for IEC 61499 distribution models. An abundant amount of key distribution approaches have been proposed for ICS, and related fields (K. T. Nguyen, Laurent & Oualha, 2015). Furthermore, key distribution is a vast area of research that requires specific expertise. Community contribution, therefore, may help *S-Lib* to scale and evolve by providing additional implementations to such mechanisms suitable for IEC 61499.

7.5.2.3.2 Functional Scalability

The current architecture of *S-Lib* allows adding further cryptographic primitives without disrupting its existing core. The cryptographic services provided by the function blocks are invoked by the FBNs. An FBN can pick and choose different function blocks from the core and arrange them with utility function blocks to provide security mechanisms. A rich set of core function blocks provides flexibility in implementing different security mechanisms according to the security and performance goals. Current

S-Lib implementation contains only a few well-known lightweight and traditional primitives restricted by external LibHydrogen and TLS libraries. They may not offer flexibility against varying performance goals. *S-Lib* is expressly obliged to implement standard algorithms in order to be entirely compatible with the ITSIS proposed in this research. Such algorithms include SHA1, SHA2 and SHA3 family of a hash function for data integrity and authentication. For confidentiality, a minimum of block cyphers such as recommended by IEC 18033-3 (ISO/IEC18033-3:2010, 2010) are required to be implemented. The maintenance and sustenance of such a functionally scalable library is a laborious and infeasible task for a limited number of researchers and developers. Therefore, an open-source *S-Lib* is greatly helped by the contributors by adding core function blocks and security mechanisms providing modern cryptographic primitives.

7.6 ITSIS Implications

7.6.1 ICS Security Certification

Security standard certified products ensure that the product has met the security requirements and has been subject to rigorous testing. Many government regulations worldwide require installing certified ICS products in safety and security-critical ICS infrastructure (Poehlmann et al., 2021). Certification of a product is generally a process requiring extensive collaboration between a vendor (product developer) and a certification authority. There are certification programs (e.g. ISASecure and Achilles) based on security standards that perform security testing and conformance of requirements. Certified ICS components provide more confidence to all stakeholders and boost the product's commercial value because of vigorous security assurance. However, securing ICS should not impede its most critical safety attributes. Ongoing CertMILS project aims to defuse the effects of security certification on safety certification processes

using a compositions security certification approach (Schulz, Griest, Golatowski & Timmermann, 2018). However, a possible constraint of security standards or guidelines, e.g. in NIST 800-82, is the lack of design guidelines as stated in (Drias et al., 2015), which can be detrimental for requirements to design traceability.

A security standard defines a particular set of requirements that a vendor must implement. A certifier's job is to rigorously test that the security requirements have been implemented according to the specified target security level. The independent nature of vendors and certifiers often adds complexity to the requirements specification life-cycle, leading to confusion around implementing the requirements (Fomin et al., 2008). Consequently, it increases the time-to-market of the product because of multiple iterations of the process. A very recent study (Constante et al., 2021) tries to solve this issue with an approach to integrate CI/CD pipelines to IEC 62443-4-1 standard practices. However, being an umbrella approach, it lacks the details by providing an abstract framework.

Moreover, a security standard, such as IEC 62443, is a sizeable document that also needs precise interpretations. The proposed process of using the repository in the certification scheme provides a method to overcome the complexity involved in requirement elicitation and traceability for IEC 62443 security standard. It provides a visual snapshot of individual product security requirements and the requirements from IEC 62443. In addition, Labelled Property Graph (LPG) based requirements specifications help to extract requirements than deriving security requirements from large textual documents, especially when working with multiple projects.

Figure 7.9 proposes an ITSIS utilisation process for ICS security certification schemes. The security requirements repository is a central component that vendors and the certification lab can access to store or retrieve LPGs containing security requirements in a structured form. The repository stores the graphs persistently, making sure that the vendors and the certifiers can access them asynchronously.

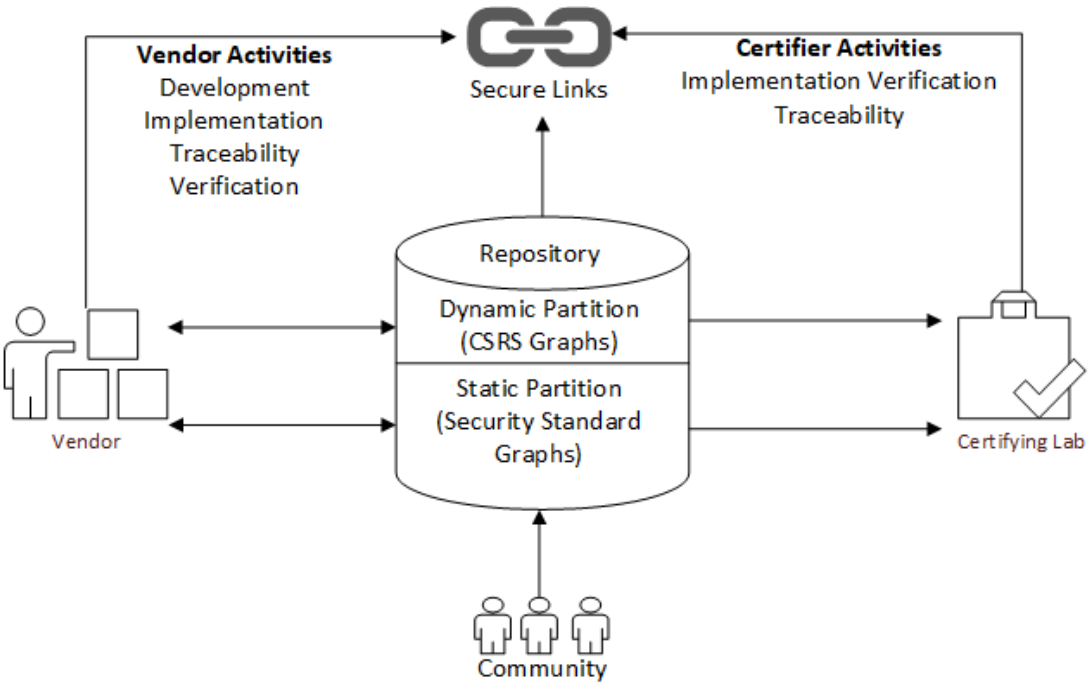

Figure 7.9: ITSIS utilisation in certification process

The static partition of the repository acts as a public space that can be updated and maintained by vendors and the community. Therefore, the public portion of the repository needs to be hosted over an accessible location from where the vendors and certifiers can clone their repository instances. The ability to create separate repository instances helps in protecting the private partition. The vendors can make their dynamic (private) part of the repository not sync with the public instance of the repository.

ITSIS security certification utilisation process presents different views to the stakeholders with a different set of activities. These views are discussed in the following.

7.6.1.1 Vendors

A vendor is an entity that develops a component for the ICS in compliance with an ICS specific security standard. The certification process is expensive and time-consuming. The vendors of such products are usually large organisations having sufficient resources regarding man-power and financial costs (Fomin et al., 2008; Lotz, 2020).

A vendor interacts with the repository to store or access the security requirements graphs. They must make a one-time effort to create a security standard graph if it is already not available in the public portion of the repository. The vendor must create a new CSRS property graph for each project or modify an existing CSRS graph from their collection of CSRS graphs if appropriate. Using the existing graphs from the repository reduces the time for future requirements elicitation and specification.

The LPGs are used with secure links to use the security requirements at the design stage. The vendor uses secure links to support secure-by-design development, implementation and traceability through the use of TORUS. The vendor shares all the relevant documents related to the product with the certifier for compliance validation. In the ITSIS context, these documents include CSRS and IEC 62443-4-2 LPGs, IEC 61499 application and the associated secure link specifications. Graphs are shared through the repository by synchronising the repository instances. The instances can be individual instances of the vendor and the certifier or the central repository.

7.6.1.2 Certification Lab

Certification labs are the intermediary entities between certification authorities and the vendors. A certification authority can be a government organisation that regulates the use of ICS in critical infrastructure. They designate the certification process to a third-party certification lab responsible for validating the compliance of vendor-provided product artefacts against a particular security standard. For example, the National Institute of Standards and Technology (NIST) offers the FIPS 140-2 certification scheme called cryptographic module validation program. However, it designates the testing of products aspiring for the certification to external third party labs worldwide.

With ITSIS, the certification lab interacts with the security requirements repository by accessing the CSRS and security standard graphs. It uses these graphs for the requirements verification and validation to check for standard compliance. This means

that the graphs used by the vendor must be synchronised with the certifiers instance. Therefore, the vendor and the certifier must ensure that their repository instances are synchronised, especially the dynamic partition.

The certifier uses the secure links shared by the vendor. The certifier can use the secure links with the LPGs from the repository to trace the CSRS security requirement with IEC 62443-4-2 requirements and its implementation in IEC 61449. In addition, the certifier generates Requirements Traceability Matrix (RTM) for verification and validation of requirements. The matrix provides a map that the certifier can examine to verify the compliance of a target security level of a CSRS and the associated IEC 62443-4-2 requirement and link them with the implementing function blocks. In essence, the certifier uses the repository and the secure links similarly to the vendor but only to verify the IEC 61499 implementations against the security standard's requirements for compliance.

7.6.1.3 Community

Requirement extraction from security standards is a laborious task because of a large set of requirements. In addition, the extension of IEC 62443-4-2 proposed in Chapter 6 makes it more challenging to create LPGs for all the Foundation Requirements (FRs) in the standard. It is not always in the interest of an individual vendor to create, update and maintain such graphs because their primary concern is creating CSRS graphs for their products.

The open-source community can help in maintaining a central repository of security standards graphs. It can be achieved using graph database tools like Neo4j (Lal, 2015) that can publish and access a graph database in a public space. Such a phenomenon can help the repository to be kept state-of-the-art to support the vendors in an efficient requirements elicitation and extraction process. It is particularly true for the static partition of the repository. However, vendors and the community can also boost

the repository's dynamic part by contributing CSRS graphs that can serve as useful templates for the security requirements for similar ICS projects.

7.6.2 An Open-Source Compatible Method

The open-source development approach contributes to trusted products and systems. More than any other desired quality of a system, security demands trust, especially for ICS deployed in critical infrastructure. Vendor provided security guarantees means that the users trust the security implementation and verification. However, it is not always possible for a vendor to fulfil a user's trust because of zero-day security attacks and human errors in verification and validation. Moreover, a vendor may not have the ability to patch a system promptly due to resource limitations.

The open-source development model alleviates the resource limitation issue. A vibrant open-source community allows the rapid patching of the vulnerabilities of the system in case of zero-day attacks. Furthermore, the accessibility of design and implementation means that the security may be validated and verified by the user or other interested stakeholders in a transparent and timely manner. Therefore, it enables the trust against security guarantees provided by the vendor.

ITSIS enables the developers and users of ICS security applications to take advantage of open-source development. The inherent objective of the ITSIS method is to provide security guarantees for an ICS component in terms of security standard requirements compliance, an intuitive secure-by-design approach, and availability and implementation of trusted cryptography mechanisms and primitives. These guarantees are provided through the use of novel artefacts that have the potential to be open-source. These include an open-source trusted security requirements repository and an IEC 61449 security function block library taking advantage of community contributions. Such artefacts and their arrangements make ITSIS highly compatible with the open-source

development approach that has the primary objective of active community participation and development.

The security requirements repository adopts the open-source development approach through community contribution. This approach is discussed in detail in Section 7.6.1.3. An active security requirements repository project enables the repository users to remain updated regarding the latest security requirements and their implementation techniques regarding security mechanisms and algorithms. It also allows the vendors to seek and share the knowledge base regarding CSRS graph templates. An example scenario is the modification of the IEC 62443-4-2 component requirement or a requirement enhancement. The community can pick up the change in the document, and the relevant graph for that particular FR can be updated promptly.

An active *SLib* project also conforms to the open-source development approach. The implications of *SLib* as an open-source community project are discussed previously in Section 7.5.2. Prompt addition of modern cryptographic primitives implementation ensures that the security requirement implementations specified in the IEC 62443-4-2 extension remain synchronised with IEC 61499 implementations available in the library. It helps the vendors in prompt implementation of the latest security requirements if the security standard is modified.

On the other hand, ITSIS also provides flexibility to develop proprietary ICS components. Security requirements can be cloned into separate instances as discussed in Section 7.6.1. A vendor can choose to keep their repository instance private by not contributing back to the repository in CSRS graphs. In addition, they can choose to modify the IEC 62443-4-2 graphs and its extended guidelines of security machines and cryptographic primitives while keeping their repository instance private. The instance can be shared with the certification lab under a no disclosure license agreement.

Similarly, proprietary cryptographic implementations can be added to *SLib*. A vendor can choose to keep the clone of the library private. For example, an indigenous

cryptographic primitive imported from a third party under a proprietary agreement can be implemented for IEC 61499 security applications and used with *SLib*. However, the advantage of this approach in a security certification process is debatable because such processes assert the use of standard primitives. Nevertheless, *SLib* is proposed as a standalone ITSIS component providing IEC 61499 implementation of cryptographic primitives supporting. Therefore, its usage to develop a proprietary ICS component is not unconventional.

7.7 Conclusions

Secure-by-design approaches can significantly reduce the effort required to build certifiably secure ICS applications. However, the security certified ICS software requires a common approach that integrates the security requirements engineering with secure-by-design techniques to provide comprehensive traceability of the security requirements.

This research contributes to solving this issue by proposing a novel multi-partitioned security requirements repository model and a novel secure-by-design technique of "Secure Links" along with IEC 61499 function block implementations to fulfil security goals in ICS applications. Case study experiments for these techniques show an improvement in IEC 61499 ICS application's system and design complexities. Moreover, secure links also produce a maintainable and scalable ICS application. The application of the security requirements repository integrated with secure links and TORUS results in a traceability matrix that can link the security standard requirements in the repository to IEC 61499 design and function block implementations of security services. This thesis discusses the ITSIS that integrates the techniques and the artefacts proposed in this research while being compatible with traditional and security application development methodologies and frameworks. In addition, the researcher discusses the implications of ITSIS on the ICS security certification process. A security function block library

(S-Lib) consisting of cryptographic primitive implementations also helps ITSIS become scalable for industry use. The researcher proposes S-Lib as an open-source initiative to help achieve functional scalability.

7.7.1 Limitations and Future Works

This section discusses some of the limitations and potential future improvements for the ITSIS approach proposed in this research.

7.7.1.1 IEC 61499 Bound Design and Development

IEC 61499 is an emerging standard for ICS distributed application developments. It provides an intuitive object-oriented approach for developing ICS application supporting features like portability and interoperability. However, there are some open issues regarding its industry-wide adoption. A state-of-the-art literature review (Lyu & Brennan, 2020) focusing on the applicability of IEC 61499 identifies its major challenges in industrial practices. These include *1)* Industrial concerns, *2)* Technical issues, and *3)* Societal aspects. Across all these barriers, the key issue is the industry's reluctance to deviate from legacy ICS development approaches and rely on an older standard like IEC 61131-3. The causes for such hesitation in adopting IEC 61499 are the cost to upgrade the legacy systems, lack of proven redesign methods, practitioners' expertise, and the lack of educational aspects around IEC 61499 regarding course designs and industrial training.

The design and implementation phase of ITSIS is very much reliant on IEC 61449 standard that is the right choice, with a view to the future. While the use of the modern IEC 61499 standard is a promising solution to some of the challenges mentioned above; it also acts as a limiting factor for ITSIS's industrial adoption for the same reasons. The secure links development methodology (Chapter 5) is a major design approach

that relies on IEC 61449. It provides secure-by-design abstractions specific to the connections between the data endpoints of IEC 61499 function blocks. Moreover, *SLib* security library that assists secure link implementation is also an IEC 61499 specific artefact. These approaches further play a significant role in security requirements traceability, thus making the requirement verification and validation process subject to the semantics of IEC 61449.

In addition, some of the issues faced by the researcher during the development of proof-of-concept IEC 61499 applications were also closely related to the issues mentioned in (Lyu & Brennan, 2020). As IEC 61499 is still in its early stages of adoption in industry and academia, some of the challenges faced by the researcher were:

1. Lack of off-the-shelf platforms and devices that are compatible with IEC 61499. For example, standard-compliant PLCs such as Wago PLCs used to implement the case study implementations in Chapter 4 and 5, are expensive as compared to the legacy PLCs. They are also not readily available in the market.

2. Lack of freely available compliant runtimes and IDEs is another issue that restricts the developers to very few options. For example, at the inception of this research, very few active and viable IEC 61449 compliant IDEs and runtimes providing portability, interoperability, and configurability were available. Therefore, the researcher chose 4diac IDE and 4diac FORTE runtime that were freely available having relatively better community support.

3. In the researcher's experience, there is an overall lack of available community support regarding IEC 61499, e.g. support forums and extensive tutorials.

Future Works

The issues mentioned above may prove to be significant barriers for the ITSIS's industry-wide use, at least in the near future. A short-term approach to mitigate such

problems is introducing backward compatibility for legacy devices for the usage of ITSIS in the design and implementation phase. It can be achieved by exploring the development of IEC 61131 compatible secure links or a similar secure-by-design approach. IEC 61131 is an obvious choice since it is considered as a predecessor for IEC 61499 (Thramboulidis, 2013) and they have a high degree of implementation reusability between them (Wenger & Zoitl, 2012). Moreover, extensive and robust implementation of secure links plugin will also remove hindrance regarding technical issues faced by the industry by providing robust design tools. However, as the standard evolves with widespread use in the industry, ITSIS's dependence on IEC 61499 will become a less significant issue in the researcher's opinion.

7.7.1.2 Current Limitations on Scope and Scalability

There are two components in ITSIS that limit its scalability regarding its practical use in ICS distributed security applications. These interrelated factors are discussed below:

LPG security requirements repository proposed in Chapter 6 provide the implementation guidelines regarding security requirements in IEC 62443-4-2 in the form of LPG nodes. These guidelines include standard security mechanisms and their associated cryptographic primitives. The current collection of these guidelines is not extensive, i.e. it specifies limited numbers of standard cryptographic algorithms and methods on a limited set of security requirements in the IEC 62443-4-2 standard. The LPG graphs of the standard needs to comprehensive for the repository to be used in industrial-scale ICS projects.

***SLib* security library's** limitations are discussed in Section 7.5.2.3. The current implementation of *SLib* also restricts the applicability of ITSIS in industrial-scale ICS projects due to the lack of standard implementations of cryptographic methods and primitives. Moreover, *SLib* complements IEC 62443-4-2 LPG graphs since it provides the implementations for the security requirements. The nonconcurrence of

these components is counterproductive, therefore severely limiting the application of ITSIS. In essence, the functional scalability of ITSIS depends on a comprehensive function block library providing cryptographic solutions.

Future Works

Consequently, future works include providing an extensive collection of IEC 61499 function blocks implementation of standard cryptographic primitives and function block networks, using these function blocks to form security mechanisms. Similarly, future works for IEC 62443-4-2 include extending the requirements structure of all of its foundation and component requirements by providing an extensive set of security mechanisms and primitives. Such an exercise requires comprehensive research on prevailing security mechanisms in the cryptography domain and the related security standards that recommend their implementation in ICS.

Moreover, the current security LPG repository is limited to the IEC 62443 standard. Other ICS specific security standards such as listed in Table 2.3 can be explored to be used with ITSIS. It will also allow industry-specific ICS such as smart grids, manufacturing, and gas and oil industries to take advantage of the ITSIS approach.

7.7.1.3 Industrial Evaluation

One of the limitations of the use of ITSIS is its lack of industrial evaluation. The evaluation approach used for ITSIS in this research is based on case studies developed to show the feasibility of each concept within the methodology. Although the results regarding security application maintainability, design and program complexities, and traceability are promising, they are not a valid substitute for industrial evaluation.

Future Works

In future, the researcher aims to collaborate with the industry and carry out projects that solve real-world problems using the ITSIS. Future works for industrial evaluations of ITSIS are needed for its two aspects:

1. A detailed study needs to be carried out in order to ascertain the usefulness of ITSIS in ICS security certification process discussed in Section 7.6.1. It can be

achieved by collaborating with industry vendors that are involved in certifying their products using certification programs such as IEC 62443 specific *ISASecure* (SECURE, 2013) or *FIPS 140-2 — Cryptographic Module Validation Program* (Vassilev, Feldman & Witte, 2014).

2. The industrial application of ITSIS in a real-world, large-scale ICS project needs to be accomplished to compare and validate the results of this research.

References

62443-1-1, I. (2009). 62443-1-1 Industrial communication networks - Network and system security - Part 1-1: Terminology, concepts and models.

62443-4-1, I. (2018). IEC 62443-4-1: 2018: Security for industrial automation and control systems-Part 4–1: Secure product development lifecycle requirements.

Aceto, L., Ingólfsdóttir, A., Larsen, K. G. & Srba, J. (2007). *Reactive systems: modelling, specification and verification*. cambridge university press.

Ahsan, M., Motla, Y. H. & Azeem, M. W. (n.d.). An ontology-based approach for handling the issues in requirement engineering. *Pakistan Academy of Sciences*, 187.

Alur, R. & Dill, D. L. (1994). A theory of timed automata. *Theoretical computer science*, *126*(2), 183–235.

Alvarez, M. L., Sarachaga, I., Burgos, A., Estévez, E. & Marcos, M. (2016). A methodological approach to model-driven design and development of automation systems. *IEEE Transactions on Automation Science and Engineering*, *15*(1), 67–79.

Alves, T., Das, R. & Morris, T. (2018). Embedding encryption and machine learning intrusion prevention systems on programmable logic controllers. *IEEE Embedded Systems Letters*, *10*(3), 99–102.

Alves, T. & Morris, T. (2018). Openplc: An iec 61,131–3 compliant open source industrial controller for cyber security research. *Computers & Security*, *78*, 364–379.

Alves, T., Morris, T. & Yoo, S.-M. (2017). Securing scada applications using openplc with end-to-end encryption. In *Proceedings of the 3rd annual industrial control system security workshop* (pp. 1–6).

Amouri, A., Alaparthy, V. T. & Morgera, S. D. (2020). A machine learning based intrusion detection system for mobile internet of things. *Sensors*, *20*(2), 461.

Andreeva, O., Gordeychik, S., Gritsai, G., Kochetova, O., Potseluevskaya, E., Sidorov, S. I. & Timorin, A. A. (2016). Industrial control systems vulnerabilities statistics. *Kaspersky Lab, Report*.

Ani, U. P. D., Watson, J. M., Green, B., Craggs, B. & Nurse, J. R. (2020). Design considerations for building credible security testbeds: Perspectives from industrial control system use cases. *Journal of Cyber Security Technology*, 1–49.

Ardis, M. A. (1997). Formal methods for telecommunication system requirements: A survey of standardized languages. *Annals of Software Engineering*, *3*(1),

157–187.

Ashibani, Y. & Mahmoud, Q. H. (2017). Cyber physical systems security: Analysis, challenges and solutions. *Computers & Security*, *68*, 81–97.

Aumasson, J.-P., Henzen, L., Meier, W. & Naya-Plasencia, M. (2013). Quark: A lightweight hash. *Journal of cryptology*, *26*(2), 313–339.

B. Dowdeswell and R. Sinha and S. G. MacDonell. (2020). *Diagnosable-by-design model-driven development for iec 61499 industrial cyber-physical systems.* doi: 10.1109/IECON43393.2020.9254620

Babin, G. & Lustman, F. (2001). Application of formal methods to scenario-based requirements engineering. *International Journal of Computers and Applications*, *23*(3), 141–151.

Baier, C. & Katoen, J.-P. (2008). *Principles of model checking*. MIT press.

Baldini, G., Skarmeta, A., Fourneret, E., Neisse, R., Legeard, B. & Le Gall, F. (2016). Security certification and labelling in internet of things. In *2016 ieee 3rd world forum on internet of things (wf-iot)* (pp. 627–632).

Barnat, J., Brim, L., Havel, V., Havlíček, J., Kriho, J., Lenčo, M., … Weiser, J. (2013). Divine 3.0–an explicit-state model checker for multithreaded c & c++ programs. In *International conference on computer aided verification* (pp. 863–868).

Beckers, K. (2015). Relating iso 27001 to the conceptual framework for security requirements engineering methods. In *Pattern and security requirements* (pp. 85–108). Springer.

Beckers, K., Faßbender, S., Heisel, M. & Schmidt, H. (2012). Using security requirements engineering approaches to support iso 27001 information security management systems development and documentation. In *2012 seventh international conference on availability, reliability and security* (pp. 242–248).

Behrmann, G., David, A., Larsen, K. G., Håkansson, J., Pettersson, P., Yi, W. & Hendriks, M. (2006). Uppaal 4.0.

Berry, G. (2016). Formally unifying modeling and design for embedded systems-a personal view. In *International symposium on leveraging applications of formal methods* (pp. 134–149).

Bicaku, A., Zsilak, M., Theiler, P., Tauber, M. & Delsing, J. (2021). Security standard compliance verification in system of systems. *IEEE Systems Journal*.

Bjørner, D. & Havelund, K. (2014). 40 years of formal methods. In *International symposium on formal methods* (pp. 42–61).

Black, G. & Vyatkin, V. (2009). Intelligent component-based automation of baggage handling systems with IEC 61499. *IEEE Transactions on Automation Science and Engineering*, *7*(2), 337–351.

Borg, M., de la Vara, J. L. & Wnuk, K. (2016). Practitioners' perspectives on change impact analysis for safety-critical software–a preliminary analysis. In *International conference on computer safety, reliability, and security* (pp. 346–358).

Bourque, P., Dupuis, R., Abran, A., Moore, J. W. & Tripp, L. (1999). The guide to the software engineering body of knowledge. *IEEE software*, *16*(6), 35–44.

Bozga, M., Daws, C., Maler, O., Olivero, A., Tripakis, S. & Yovine, S. (1998). Kronos: A model-checking tool for real-time systems. In *International symposium on*

formal techniques in real-time and fault-tolerant systems (pp. 298–302).

Briciu, C.-V. & Filip, I. (2014). The challenge of safety and security in automotive systems. In *2014 ieee 9th ieee international symposium on applied computational intelligence and informatics (saci)* (pp. 177–181).

Bruel, J.-M., Ebersold, S., Galinier, F., Mazzara, M., Naumchev, A. & Meyer, B. (2021). The role of formalism in system requirements. *ACM Computing Surveys (CSUR)*, *54*(5), 1–36.

Burns, A., Dobbing, B. & Vardanega, T. (2004). Guide for the use of the ada ravenscar profile in high integrity systems. *ACM SIGAda Ada Letters*, *24*(2), 1–74.

Cabadini, F., Montalbano, G., Kollegger, G., Mayer, H., Vytakin, V., Soldatos, J., … Cavadini, F. (2019). Iec-61499 distributed automation for the next generation of manufacturing systems. In *The digital shopfloor: Industrial automation in the industry 4.0 era–performance analysis and applications* (pp. 103–127). River.

Calder, A. (2013). *Iso27001/iso27002: A pocket guide*. IT Governance Publishing.

Campos, J. (2010). Guest editorial special section on formal methods in manufacturing. *IEEE Transactions on Industrial Informatics*, *6*(2), 125–126.

Canetti, R. & Krawczyk, H. (2001). Analysis of key-exchange protocols and their use for building secure channels. In *International conference on the theory and applications of cryptographic techniques* (pp. 453–474).

Chapman, J. P., Ofner, S. & Pauksztelo, P. (2016). Key factors in industrial control system security. In *2016 ieee 41st conference on local computer networks (lcn)* (pp. 551–554).

Chawuthai, R. & Takeda, H. (2015). rsim: Simplifying an rdf graph at the visualization tier for non-expert users. In *International semantic web conference (posters & demos)*.

Che, X. & Maag, S. (2014). Testing protocols in internet of things by a formal passive technique. *Science China Information Sciences*, *57*(3), 1–13.

Cheminod, M., Bertolotti, I. C., Durante, L., Sisto, R. & Valenzano, A. (2006). On the use of automatic tools for the formal analysis of ieee 802.11 key-exchange protocols. In *2006 ieee international workshop on factory communication systems* (pp. 273–282).

Cimatti, A., Clarke, E., Giunchiglia, F. & Roveri, M. (1999). Nusmv: A new symbolic model verifier. In *International conference on computer aided verification* (pp. 495–499).

Commission, I. E. et al. (2016). *Iec 62443-1-1, industrial communication network–network and system security. part 1-1: Terminology, concepts and models.*

Constante, F. M., Soares, R., Pinto-Albuquerque, M., Méndez, D. & Beckers, K. (2021). Integration of security standards in devops pipelines: An industry case study. *arXiv preprint arXiv:2105.13024*.

Control, Process and Requirements, Security and Bond, Digital. (2006). *Field Device Protection Profile For SCADA Systems In Medium Robustness Environments.* `https://www.enisa.europa.eu/publications/annex-iv/at_download/fullReport`. (Accessed: 2021-01-29)

Corbin, J. & Strauss, A. (2014). *Basics of qualitative research: Techniques and procedures for developing grounded theory*. Sage publications.

Cornière, A., Fortineau, V., Paviot, T. & Lamouri, S. (2016). Requirements verification method for system engineering based on a rdf logic view. In *Service orientation in holonic and multi-agent manufacturing* (pp. 135–143). Springer.

Council, N. R. et al. (2012). *Terrorism and the electric power delivery system*. National Academies Press.

Culot, G., Fattori, F., Podrecca, M. & Sartor, M. (2019). Addressing industry 4.0 cybersecurity challenges. *IEEE Engineering Management Review*, *47*(3), 79–86.

Czerny, B. J., D'Ambrosio, J. & Debouk, R. (2002). Iso 26262 functional safety draft international standard for road vehicles: Background, status, and overview. *Origins*, *9*(1.2004), 2003.

Davis, J. A., Clark, M., Cofer, D., Fifarek, A., Hinchman, J., Hoffman, J., ... Wagner, L. (2013). Study on the barriers to the industrial adoption of formal methods. In *International workshop on formal methods for industrial critical systems* (pp. 63–77).

Den Braber, F., Hogganvik, I., Lund, M. S., Stølen, K. & Vraalsen, F. (2007). Model-based security analysis in seven steps—a guided tour to the coras method. *BT Technology Journal*, *25*(1), 101–117.

Deng, P., Ren, G., Yuan, W., Chen, F. & Hua, Q. (2015). An integrated framework of formal methods for interaction behaviors among industrial equipments. *Microprocessors and Microsystems*, *39*(8), 1296–1304.

Derhamy, H., Eliasson, J., Delsing, J. & Priller, P. (2015). A survey of commercial frameworks for the internet of things. In *2015 ieee 20th conference on emerging technologies & factory automation (etfa)* (pp. 1–8).

Dhirani, L. L., Armstrong, E. & Newe, T. (2021). Industrial iot, cyber threats, and standards landscape: Evaluation and roadmap. *Sensors*, *21*(11), 3901.

Diffie, W. & Hellman, M. (1976). New directions in cryptography. *IEEE transactions on Information Theory*, *22*(6), 644–654.

Donenfeld, J. A. (2017). Wireguard: Next generation kernel network tunnel. In *Ndss*.

Drias, Z., Serhrouchni, A. & Vogel, O. (2015). Analysis of cyber security for industrial control systems. In *2015 international conference on cyber security of smart cities, industrial control system and communications (ssic)* (pp. 1–8).

Drivers & Controls. (2019). *Schneider and Rockwell join ISA in cyber-security drive*. Retrieved 2019-07-30, from `https://drivesncontrols.com/news/fullstory.php/aid/6064/Schneider_and_Rockwell_join_ISA_in_cyber-security_drive.html`

Dunlap, S., Butts, J., Lopez, J., Rice, M. & Mullins, B. (2016). Using timing-based side channels for anomaly detection in industrial control systems. *International Journal of Critical Infrastructure Protection*, *15*, 12–26.

Easterbrook, S., Lutz, R., Covington, R., Kelly, J., Ampo, Y. & Hamilton, D. (1998). Experiences using lightweight formal methods for requirements modeling. *IEEE Transactions on Software Engineering*, *24*(1), 4–14.

Ebenezer, J. & Murty, S. S. (2015). Deployment of wireless sensor network for

radiation monitoring. In *2015 international conference on computing and network communications (coconet)* (pp. 27–32).

Eclipse 4diac™ - 4diac IDE and 4diac FORTE runtime. (2021). Retrieved from `https://www.eclipse.org/4diac/index.php`

Ehrlich, M., Gergeleit, M., Trsek, H. & Lukas, G. (2020). Towards automated security evaluation within the industrial reference architecture. In *2020 25th ieee international conference on emerging technologies and factory automation (etfa)* (Vol. 1, pp. 1644–1651).

Ehrlich, M., Wisniewski, L., Trsek, H. & Jasperneite, J. (2018). Modelling and automatic mapping of cyber security requirements for industrial applications: Survey, problem exposition, and research focus. In *2018 14th ieee international workshop on factory communication systems (wfcs)* (pp. 1–9).

Eiza, M. H., Randles, M., Johnson, P., Shone, N., Pang, J. & Bhih, A. (2015). Rail internet of things: An architectural platform and assured requirements model. In *2015 ieee international conference on computer and information technology; ubiquitous computing and communications; dependable, autonomic and secure computing; pervasive intelligence and computing* (pp. 364–370).

Elahi, G. & Yu, E. (2007). A goal oriented approach for modeling and analyzing security trade-offs. In *International conference on conceptual modeling* (pp. 375–390).

El Houssaïni, S. E. G., Maskani, I. & Boutahar, J. (2021). A security requirement engineering case study: Challenges and lessons learned. In *Intelligent computing* (pp. 761–783). Springer.

Evans, P. C. & Annunziata, M. (2012). Industrial internet: Pushing the boundaries. *General Electric Reports*, 488–508.

Fabian, B., Gürses, S., Heisel, M., Santen, T. & Schmidt, H. (2010). A comparison of security requirements engineering methods. *Requirements engineering*, *15*(1), 7–40.

Faily, S., Iacob, C., Ali, R. & Ki-Aries, D. (2020). Identifying implicit vulnerabilities through personas as goal models. In *Computer security* (pp. 185–202). Springer.

Fay, A., Vogel-Heuser, B., Frank, T., Eckert, K., Hadlich, T. & Diedrich, C. (2015). Enhancing a model-based engineering approach for distributed manufacturing automation systems with characteristics and design patterns. *Journal of Systems and Software*, *101*, 221–235.

Feiler, P. H., Gluch, D. P. & Hudak, J. J. (2006). *The architecture analysis & design language (aadl): An introduction* (Tech. Rep.). Carnegie-Mellon Univ Pittsburgh PA Software Engineering Inst.

Felizardo, K. R., Nakagawa, E. Y., Feitosa, D., Minghim, R. & Maldonado, J. C. (2010). An approach based on visual text mining to support categorization and classification in the systematic mapping. In *14th international conference on evaluation and assessment in software engineering (ease)* (pp. 1–10).

Fenz, S., Plieschnegger, S. & Hobel, H. (2016). Mapping information security standard iso 27002 to an ontological structure. *Information & Computer Security*.

Fernández, M., Kantarcioglu, M. & Thuraisingham, B. (2016). A framework for secure

data collection and management for internet of things. In *Proceedings of the 2nd annual industrial control system security workshop* (pp. 30–37).

Fisher, M. (2011). *An introduction to practical formal methods using temporal logic* (Vol. 82). Wiley Online Library.

Fitzgerald, J., Bicarregui, J., Larsen, P. G. & Woodcock, J. (2013). Industrial deployment of formal methods: Trends and challenges. In *Industrial deployment of system engineering methods* (pp. 123–143). Springer.

Fockel, M., Merschjohann, S., Fazal-Baqaie, M., Förder, T., Hausmann, S. & Waldeck, B. (2019). Designing and integrating iec 62443 compliant threat analysis. In *European conference on software process improvement* (pp. 57–69).

Fomin, V. V., Vries, H. & Barlette, Y. (2008). Iso/iec 27001 information systems security management standard: exploring the reasons for low adoption. In *Euromot 2008 conference, nice, france.*

Forsberg, K. & Mooz, H. (1991). The relationship of system engineering to the project cycle. In *Incose international symposium* (Vol. 1, pp. 57–65).

Franceschini, F., Maisano, D. & Mastrogiacomo, L. (2016). Empirical analysis and classification of database errors in scopus and web of science. *Journal of Informetrics*, *10*(4), 933–953.

Francis, N., Green, A., Guagliardo, P., Libkin, L., Lindaaker, T., Marsault, V., ... Taylor, A. (2018). Cypher: An evolving query language for property graphs. In *Proceedings of the 2018 international conference on management of data* (pp. 1433–1445).

Fruth, J. & Nett, E. (2014). Uniform approach of risk communication in distributed it environments combining safety and security aspects. In *International conference on computer safety, reliability, and security* (pp. 289–300).

Fuchs, A., Gürgens, S. & Rudolph, C. (2010). A formal notion of trust–enabling reasoning about security properties. In *Ifip international conference on trust management* (pp. 200–215).

Fürst, S., Mössinger, J., Bunzel, S., Weber, T., Kirschke-Biller, F., Heitkämper, P., ... Lange, K. (2009). Autosar–a worldwide standard is on the road. In *14th international vdi congress electronic systems for vehicles, baden-baden* (Vol. 62, p. 5).

García-Ferreira, I., Laorden, C., Santos, I. & Bringas, P. G. (2014). A survey on static analysis and model checking. In *International joint conference soco'14-cisis'14-iceute'14* (pp. 443–452).

Garlan, D. (2000). *Software architecture: a roadmap. the future of software engineering, a. finkekstein.* ACM Press.

Ghaleb, A., Zhioua, S. & Almulhem, A. (2018). On plc network security. *International Journal of Critical Infrastructure Protection*, *22*, 62–69.

Ghosh, S. & Sampalli, S. (2019). A survey of security in scada networks: Current issues and future challenges. *IEEE Access*, *7*, 135812–135831.

Giannakopoulou, D., Pressburger, T., Mavridou, A. & Schumann, J. (2020). Generation of formal requirements from structured natural language. In *International working conference on requirements engineering: Foundation for software quality* (pp.

19–35).

Giannakopoulou, D., Pressburger, T., Mavridou, A. & Schumann, J. (2021). Automated formalization of structured natural language requirements. *Information and Software Technology*, 106590.

GICSP, E. H., Assante, M. & Conway, T. (2014). An abbreviated history of automation & industrial controls systems and cybersecurity.

Graham, J., Hieb, J. & Naber, J. (2016). Improving cybersecurity for industrial control systems. In *2016 ieee 25th international symposium on industrial electronics (isie)* (pp. 618–623).

Green, B., Lee, A., Antrobus, R., Roedig, U., Hutchison, D. & Rashid, A. (2017). Pains, gains and plcs: ten lessons from building an industrial control systems testbed for security research. In *10th {USENIX} workshop on cyber security experimentation and test ({CSET} 17)*.

Gubbi, J., Buyya, R., Marusic, S. & Palaniswami, M. (2013). Internet of things (iot): A vision, architectural elements, and future directions. *Future generation computer systems*, *29*(7), 1645–1660.

Gunter, D. G., Medoff, M. D. & OBrien, P. C. (2018). *Implementing IEC 62443: a pragmatic approach to cybersecurity*. Exida.

Hailesellasie, M. & Hasan, S. R. (2018). Intrusion detection in plc-based industrial control systems using formal verification approach in conjunction with graphs. *Journal of Hardware and Systems Security*, *2*(1), 1–14.

Hansch, G., Schneider, P. & Brost, G. S. (2019). Deriving impact-driven security requirements and monitoring measures for industrial iot. In *Proceedings of the 5th on cyber-physical system security workshop* (pp. 37–45).

Hansch, G., Schneider, P., Fischer, K. & Böttinger, K. (2019). A unified architecture for industrial iot security requirements in open platform communications. In *2019 24th ieee international conference on emerging technologies and factory automation (etfa)* (pp. 325–332).

Harn, L., Hsu, C. & Xia, Z. (2021). Lightweight and flexible key distribution schemes for secure group communications. *Wireless Networks*, *27*(1), 129–136.

Harrison, R., Vera, D. & Ahmad, B. (2016). Engineering methods and tools for cyber–physical automation systems. *Proceedings of the IEEE*, *104*(5), 973–985.

Helmerich, A., Koch, N., Mandel, L., Braun, P., Dornbusch, P., Gruler, A., ... others (2005). Study of worldwide trends and r&d programmes in embedded systems in view of maximising the impact of a technology platform in the area. *Final Report for the European Comission, Brussels, Belgium*.

Henrie, M. (2013). Cyber security risk management in the scada critical infrastructure environment. *Engineering Management Journal*, *25*(2), 38–45.

Heron, S. (2009). Advanced encryption standard (aes). *Network Security*, *2009*(12), 8–12.

Hierons, R. M., Bogdanov, K., Bowen, J. P., Cleaveland, R., Derrick, J., Dick, J., ... others (2009). Using formal specifications to support testing. *ACM Computing Surveys (CSUR)*, *41*(2), 1–76.

Hinchey, M. G., Rash, J. L., Rouff, C. A. & Gračanin, D. (2006). Achieving dependability in sensor networks through automated requirements-based programming. *Computer Communications*, *29*(2), 246–256.

Hirsch, M., Missal, D. & Hanisch, H.-M. (2008). Design and verification of distributed industrial manufacturing control systems. In *2008 34th annual conference of ieee industrial electronics* (pp. 152–157).

Hofmann, M. & Klinkenberg, R. (2016). *Rapidminer: Data mining use cases and business analytics applications*. CRC Press.

Hogan, M., Piccarreta, B., Group, I. I. C. S. W. et al. (2018). *Interagency report on status of international cybersecurity standardization for the internet of things (iot)* (Tech. Rep.). National Institute of Standards and Technology.

Holzmann, G. J. (1997). The model checker spin. *IEEE Transactions on software engineering*, *23*(5), 279–295.

Homay, A. & de Sousa, M. (2016). Multi-cast authentication framework for distributed control systems based on iec 61499. In *2016 ieee 21st international conference on emerging technologies and factory automation (etfa)* (pp. 1–4).

Homay, A., Martins, A. P., de Sousa, M. & Kashefi, F. (2016). Message security for automation and control applications based on iec61131-3. In *2016 future technologies conference (ftc)* (pp. 991–997).

Hopcroft, J. E., Motwani, R. & Ullman, J. D. (2001). Introduction to automata theory, languages, and computation. *Acm Sigact News*, *32*(1), 60–65.

Houmb, S. H., Islam, S., Knauss, E., Jürjens, J. & Schneider, K. (2010). Eliciting security requirements and tracing them to design: an integration of common criteria, heuristics, and umlsec. *Requirements Engineering*, *15*(1), 63–93.

Howard, M. & Lipner, S. (2006). *The security development lifecycle* (Vol. 8). Microsoft Press Redmond.

hping - active network security tool. (n.d.). Retrieved Accessed: 2019-01-23, from `http://www.hping.org`

Huth, M. & Ryan, M. (2004). *Logic in computer science: Modelling and reasoning about systems*. Cambridge university press.

IEC 62443 Conformance Certification. (2019). Retrieved from `https://www.isasecure.org/en-US/End-Users/ISASecure-Certified-Components`

ISA/IEC. (2018). IEC 62443-4-2:2019 Security for industrial automation and control systems - Part 4-2: Technical security requirements for IACS components.

Ishikawa, F., Suleiman, B., Yamamoto, K. & Honiden, S. (2009). Physical interaction in pervasive computing: formal modeling, analysis and verification. In *Proceedings of the 2009 international conference on pervasive services* (pp. 133–140).

ISO/IEC 15408-1:2009 Criteria, C. (2009). Information technology — Security techniques — Evaluation criteria for IT security .

ISO/IEC18033-3:2010. (2010). Information Technology, Security techniques, Encryption algorithms (part 3: Block ciphers), 18033-3:2010.

Iyengar, S. S. & Lepper, M. R. (2000). When choice is demotivating: Can one desire

too much of a good thing? *Journal of personality and social psychology*, *79*(6), 995.

Jabangwe, R. & Nguyen-Duc, A. (2020). Siot framework: Towards an approach for early identification of security requirements for internet-of-things applications. *e-Informatica Software Engineering Journal*, *14*(1).

Jaffe, M. S., Leveson, N. G., Heimdahl, M. & Melhart, B. (1990). Software requirements analysis for real-time process-control systems.

Jalali, S. & Wohlin, C. (2012). Systematic literature studies: database searches vs. backward snowballing. In *Proceedings of the 2012 acm-ieee international symposium on empirical software engineering and measurement* (pp. 29–38).

Jin, C., Valizadeh, S. & van Dijk, M. (2018). Snapshotter: Lightweight intrusion detection and prevention system for industrial control systems. In *2018 ieee industrial cyber-physical systems (icps)* (pp. 824–829).

Johnson, R. E. (2010). Survey of scada security challenges and potential attack vectors. In *2010 international conference for internet technology and secured transactions* (pp. 1–5).

Karatas, E. K., Iyidir, B. & Birtürk, A. (2014). Ontology-based software requirements reuse: Case study in fire control software product line domain. In *2014 ieee international conference on data mining workshop* (pp. 832–839).

Katyara, S., Shah, M. A., Zardari, S., Chowdhry, B. S. & Kumar, W. (2017). Wsn based smart control and remote field monitoring of pakistan's irrigation system using scada applications. *Wireless Personal Communications*, *95*(2), 491–504.

Kaur, A., Gulati, S. & Singh, S. (2012). A comparative study of two formal specification languages: Z-notation & b-method. In *Proceedings of the second international conference on computational science, engineering and information technology* (pp. 524–531).

Kavallieratos, G., Katsikas, S. & Gkioulos, V. (2020). Safesec tropos: Joint security and safety requirements elicitation. *Computer Standards & Interfaces*, *70*, 103429.

Kelsey, J., Chang, S.-j. & Perlner, R. (2016). Sha-3 derived functions: cshake, kmac, tuplehash, and parallelhash. *NIST special publication*, *800*, 185.

Keshav, S. (2007). How to read a paper. *ACM SIGCOMM Computer Communication Review*, *37*(3), 83–84.

Khan, M. T., Serpanos, D. & Shrobe, H. (2017). Armet: Behavior-based secure and resilient industrial control systems. *Proceedings of the IEEE*, *106*(1), 129–143.

Khan, S., Lee, W.-K. & Hwang, S. O. (2021). Aechain: A lightweight blockchain for iot applications. *IEEE Consumer Electronics Magazine*.

Kissel, R., Scholl, M., Skolochenko, S. & Li, X. (2006). Nist sp800-88 guidelines for media sanitization. *NIST Spec Publ*, *88*.

Kivelä, T., Golder, M. & Furmans, K. (2018). Towards an approach for assuring machinery safety in the iiot-age. *Logistics Journal: Proceedings*, *2018*(01).

Knorr, K. (2013). Patching our critical infrastructure: Towards an efficient patch and update management for industrial control systems. In *Securing critical infrastructures and critical control systems: Approaches for threat protection* (pp. 190–216). IGI Global.

Knowles, W., Prince, D., Hutchison, D., Disso, J. F. P. & Jones, K. (2015). A survey of cyber security management in industrial control systems. *International journal of critical infrastructure protection*, *9*, 52–80.

Køien, G. M. (2020). A philosophy of security architecture design. *Wireless Personal Communications*, *113*(3), 1615–1639.

Kossak, F. & Mashkoor, A. (2016). How to select the suitable formal method for an industrial application: A survey. In *International conference on abstract state machines, alloy, b, tla, vdm, and z* (pp. 213–228).

Koundinya, A. K., Sharvani, G. & Rao, K. U. (2016). Calibrated security measures for centralized iot applications of smart grids. In *2016 international conference on computation system and information technology for sustainable solutions (csitss)* (pp. 153–157).

Křetínskỳ, J. (2016). Survey of statistical verification of linear unbounded properties: Model checking and distances. In *International symposium on leveraging applications of formal methods* (pp. 27–45).

Kronfuss, E. (2018). *Industrial cyber security standard-iec 62443* (Tech. Rep.).

Krotofil, M. & Gollmann, D. (2013). Industrial control systems security: What is happening? In *2013 11th ieee international conference on industrial informatics (indin)* (pp. 670–675).

Kulik, T., Tran-Jørgensen, P. W. & Boudjadar, J. (2019). Compliance verification of a cyber security standard for cloud-connected scada. In *2019 global iot summit (giots)* (pp. 1–6).

Kusumah, R. I. T. & Andriawan, Y. (2019). Implementation of cryptography module security certification based on sni iso/iec 19790: 2012-security requirements for cryptography module. In *2019 international seminar on intelligent technology and its applications (isitia)* (pp. 216–221).

Lahbib, A., Wakrime, A. A., Laouiti, A., Toumi, K. & Martin, S. (2020). An event-b based approach for formal modelling and verification of smart contracts. In *International conference on advanced information networking and applications* (pp. 1303–1318).

Lal, M. (2015). *Neo4j graph data modeling*. Packt Publishing Ltd.

Langner, R. (2011). Stuxnet: Dissecting a cyberwarfare weapon. *IEEE Security & Privacy*, *9*(3), 49–51.

Lasi, H., Fettke, P., Kemper, H.-G., Feld, T. & Hoffmann, M. (2014). Industry 4.0. *Business & information systems engineering*, *6*(4), 239–242.

Latella, D., Majzik, I. & Massink, M. (1999). Towards a formal operational semantics of uml statechart diagrams. In *International conference on formal methods for open object-based distributed systems* (pp. 331–347).

Lecomte, T. (2009). Applying a formal method in industry: a 15-year trajectory. In *International workshop on formal methods for industrial critical systems* (pp. 26–34).

Lendvay, R. L. (2016). *Shadows of stuxnet: Recommendations for us policy on critical infrastructure cyber defense derived from the stuxnet attack* (Tech. Rep.). NAVAL POSTGRADUATE SCHOOL MONTEREY CA MONTEREY United States.

Leuschel, M. & Butler, M. (2003). Prob: A model checker for b. In *International symposium of formal methods europe* (pp. 855–874).

Leveson, N. G., Heimdahl, M. P. & Reese, J. D. (1999). Designing specification languages for process control systems: Lessons learned and steps to the future? In *Software engineering—esec/fse'99* (pp. 127–146).

Libhydrogen - cryptographic library for constrained environments. (n.d.). Retrieved Accessed: 2020-03-24, from `https://github.com/jedisct1/libhydrogen/wiki`

Linhares, M. V., de Oliveira, R. S., Farines, J.-M. & Vernadat, F. (2007). Introducing the modeling and verification process in sysml. In *2007 ieee conference on emerging technologies and factory automation (efta 2007)* (pp. 344–351).

Litherland, P., Orr, R. & Piggin, R. (2016). Cyber security of operational technology: understanding differences and achieving balance between nuclear safety and nuclear security.

Liu, L., Kong, W., Ando, T., Yatsu, H. & Fukuda, A. (2013). A survey of acceleration techniques for smt-based bounded model checking. In *2013 international conference on computer sciences and applications* (pp. 554–559).

Liu, S., Liu, X. P. & El Saddik, A. (2013). Denial-of-service (dos) attacks on load frequency control in smart grids. In *2013 ieee pes innovative smart grid technologies conference (isgt)* (pp. 1–6).

Lodderstedt, T., Basin, D. & Doser, J. (2002). Secureuml: A uml-based modeling language for model-driven security. In *International conference on the unified modeling language* (pp. 426–441).

Loruenser, T., Pöhls, H. C., Sell, L. & Laenger, T. (2018). Cryptsdlc: Embedding cryptographic engineering into secure software development lifecycle. In *Proceedings of the 13th international conference on availability, reliability and security* (pp. 1–9).

Lotz, V. (2020). Cybersecurity certification for agile and dynamic software systems–a process-based approach. In *2020 ieee european symposium on security and privacy workshops (euros&pw)* (pp. 85–88).

Lu, T., Guo, X., Li, Y., Peng, Y., Zhang, X., Xie, F. & Gao, Y. (2014). Cyberphysical security for industrial control systems based on wireless sensor networks. *International Journal of Distributed Sensor Networks*, *10*(6), 438350.

Lu, X., Wang, W. & Ma, J. (2013). An empirical study of communication infrastructures towards the smart grid: Design, implementation, and evaluation. *IEEE Transactions on Smart Grid*, *4*(1), 170–183.

Lyu, G. & Brennan, R. W. (2020). Towards iec 61499-based distributed intelligent automation: A literature review. *IEEE Transactions on Industrial Informatics*, *17*(4), 2295–2306.

Mahnke, W., Leitner, S.-H. & Damm, M. (2009). *Opc unified architecture*. Springer Science & Business Media.

Manifavas, C., Hatzivasilis, G., Fysarakis, K. & Papaefstathiou, Y. (2016). A survey of lightweight stream ciphers for embedded systems. *Security and Communication Networks*, *9*(10), 1226–1246.

Marali, M. & Sudarsan, S. D. (2018). Graceful reincarnation of legacy industrial control systems. In *2018 ieee 3rd international conference on computing, communication and security (icccs)* (pp. 224–229).

Martín-Liras, L., Prada, M. A., Fuertes, J. J., Morán, A., Alonso, S. & Domínguez, M. (2017). Comparative analysis of the security of configuration protocols for industrial control devices. *International Journal of Critical Infrastructure Protection*, *19*, 4–15.

Martins, L. E. G. & Gorschek, T. (2016). Requirements engineering for safety-critical systems: A systematic literature review. *Information and software technology*, *75*, 71–89.

Maskani, I., Boutahar, J. & El Houssaïni, S. E. G. (2016). Analysis of security requirements engineering: towards a comprehensive approach. *Int. J. Adv. Comput. Sci. Appl*, *7*(11), 38–45.

Matheu, S. N., Hernández-Ramos, J. L., Skarmeta, A. F. & Baldini, G. (2020). A survey of cybersecurity certification for the internet of things. *ACM Computing Surveys (CSUR)*, *53*(6), 1–36.

McIntosh, H. V. (2009). *One dimensional cellular automata*. Luniver Press.

Mead, N. R. & Stehney, T. (2005). Security quality requirements engineering (square) methodology. *ACM SIGSOFT Software Engineering Notes*, *30*(4), 1–7.

Mellado, D., Fernández-Medina, E. & Piattini, M. (2007). A common criteria based security requirements engineering process for the development of secure information systems. *Computer standards & interfaces*, *29*(2), 244–253.

Meseguer, J. (2012). Twenty years of rewriting logic. *The Journal of Logic and Algebraic Programming*, *81*(7-8), 721–781.

Mohamed, M. A., Challenger, M. & Kardas, G. (2020). Applications of model-driven engineering in cyber-physical systems: A systematic mapping study. *Journal of computer languages*, *59*, 100972.

Morimoto, S. & Cheng, J. (2007). A security specification library with a schemaless database. In *International conference on computational science* (pp. 890–893).

Morimoto, S., Horie, D. & Cheng, J. (2006). A security requirement management database based on iso/iec 15408. In *International conference on computational science and its applications* (pp. 1–10).

Mosteiro-Sanchez, A., Barcelo, M., Astorga, J. & Urbieta, A. (2020). Securing iiot using defence-in-depth: towards an end-to-end secure industry 4.0. *Journal of Manufacturing Systems*, *57*, 367–378.

Mosterman, P. J. & Zander, J. (2016). Cyber-physical systems challenges: a needs analysis for collaborating embedded software systems. *Software & Systems Modeling*, *15*(1), 5–16.

Mouha, N., Mennink, B., Van Herrewege, A., Watanabe, D., Preneel, B. & Verbauwhede, I. (2014). Chaskey: an efficient mac algorithm for 32-bit microcontrollers. In *International conference on selected areas in cryptography* (pp. 306–323).

Mouratidis, H., Argyropoulos, N. & Shei, S. (2016). Security requirements engineering for cloud computing: The secure tropos approach. In *Domain-specific conceptual modeling* (pp. 357–380). Springer.

Mouratisis, H. (2010). Secure by design: Considering security from the early stages of the information systems development. In *Handbook of electronic security and digital forensics* (pp. 115–132). World Scientific.

Müller, T., Walz, A., Kiefer, M., Doran, H. D. & Sikora, A. (2018). Challenges and prospects of communication security in real-time ethernet automation systems. In *2018 14th ieee international workshop on factory communication systems (wfcs)* (pp. 1–9).

Mussmann, A., Brunner, M. & Breu, R. (2020). Mapping the state of security standards mappings. In *Wirtschaftsinformatik (zentrale tracks)* (pp. 1309–1324).

Needham, M. & Hodler, A. E. (2019). *Graph algorithms: Practical examples in apache spark and neo4j*. O'Reilly Media.

Nguyen, K. T., Laurent, M. & Oualha, N. (2015). Survey on secure communication protocols for the internet of things. *Ad Hoc Networks*, *32*, 17–31.

Nguyen, P. H., Ali, S. & Yue, T. (2017). Model-based security engineering for cyber-physical systems: A systematic mapping study. *Information and Software Technology*, *83*, 116–135.

Nicholas, K., Bhatti, Z. E. & Roop, P. S. (2012). Model-driven development of industrial embedded systems: Challenges faced and lessons learnt. In *Proceedings of 2012 ieee 17th international conference on emerging technologies & factory automation (etfa 2012)* (pp. 1–4).

NIST, F. (2016). FIPS 140-2, Security requirements for cryptographic modules. *National Institute of Standards and Technology*.

Nourian, A. & Madnick, S. (2015). A systems theoretic approach to the security threats in cyber physical systems applied to stuxnet. *IEEE Transactions on Dependable and Secure Computing*, *15*(1), 2–13.

Nunamaker Jr, J. F., Chen, M. & Purdin, T. D. (1990). Systems development in information systems research. *Journal of management information systems*, *7*(3), 89–106.

Nyberg, K. (1996). Generalized feistel networks. In *International conference on the theory and application of cryptology and information security* (pp. 91–104).

packeth. (n.d.). Retrieved Accessed: 2019-01-23, from `http://packeth.sourceforge.net/packeth/Home.html`

Pal, S., Hitchens, M., Rabehaja, T. & Mukhopadhyay, S. (2020). Security requirements for the internet of things: A systematic approach. *Sensors*, *20*(20), 5897.

Palomares, C., Quer, C. & Franch, X. (2017). Requirements reuse and requirement patterns: a state of the practice survey. *Empirical Software Engineering*, *22*(6), 2719–2762.

Pandey, R. (2010). Architectural description languages (adls) vs uml: a review. *ACM SIGSOFT Software Engineering Notes*, *35*(3), 1–5.

Parvez, B., Ali, J., Ahmed, U. & Farhan, M. (2015). Framework for implementation of aga 12 for secured scada operation in oil and gas industry. In *2015 2nd international conference on computing for sustainable global development (indiacom)* (pp. 1281–1284).

Pawgasame, W. (2016). A survey in adaptive hybrid wireless sensor network for

military operations. In *2016 second asian conference on defence technology (acdt)* (pp. 78–83).

Peffers, K., Tuunanen, T., Rothenberger, M. A. & Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of management information systems*, *24*(3), 45–77.

Peltola, J., Christensen, J., Sierla, S. & Koskinen, K. (2007). A migration path to iec 61499 for the batch process industry. In *2007 5th ieee international conference on industrial informatics* (Vol. 2, pp. 811–816).

Petersen, K., Feldt, R., Mujtaba, S. & Mattsson, M. (2008). Systematic mapping studies in software engineering. In *12th international conference on evaluation and assessment in software engineering (ease) 12* (pp. 1–10).

Petersen, K., Vakkalanka, S. & Kuzniarz, L. (2015). Guidelines for conducting systematic mapping studies in software engineering: An update. *Information and Software Technology*, *64*, 1–18.

Poehlmann, N., Caramancion, K. M., Tatar, I., Li, Y., Barati, M. & Merz, T. (2021). The organizational cybersecurity success factors: An exhaustive literature review. *Advances in Security, Networks, and Internet of Things*, 377–395.

Pohrmen, F. H. & Saha, G. (2021). Lightbc: A lightweight hash-based blockchain for the secured internet of things. In *International conference on innovative computing and communications* (pp. 811–819).

Pokornỳ, J. (2015). Graph databases: their power and limitations. In *Ifip international conference on computer information systems and industrial management* (pp. 58–69).

Pokornỳ, J. (2016). Conceptual and database modelling of graph databases. In *Proceedings of the 20th international database engineering & applications symposium* (pp. 370–377).

Potlapally, N. R., Ravi, S., Raghunathan, A., Lee, R. B. & Jha, N. K. (2007). Configuration and extension of embedded processors to optimize ipsec protocol execution. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, *15*(5), 605–609.

Preston, R., Duck, M., Finnegan, C. & Munoz, R. (2019). Integrating cyber security requirements into a power management system. In *2019 ieee petroleum and chemical industry committee conference (pcic)* (pp. 131–138).

Rahman, M. A., Asyhari, A. T., Leong, L., Satrya, G., Tao, M. H. & Zolkipli, M. (2020). Scalable machine learning-based intrusion detection system for iot-enabled smart cities. *Sustainable Cities and Society*, *61*, 102324.

Ramadan, Q., Salnitriy, M., Strüber, D., Jürjens, J. & Giorgini, P. (2017). From secure business process modeling to design-level security verification. In *2017 acm/ieee 20th international conference on model driven engineering languages and systems (models)* (pp. 123–133).

Ramanathan, R. (2014). The iec 61131-3 programming languages features for industrial control systems. In *2014 world automation congress (wac)* (pp. 598–603).

Rogowski, D. (2018). Identification of information technology security issues specific to industrial control systems. In *International conference on dependability and*

complex systems (pp. 400–408).

Roscoe, B. (1998). The theory and practice of concurrency.

Rosenstatter, T. & Olovsson, T. (2018). Open problems when mapping automotive security levels to system requirements. In *Vehits* (pp. 251–260).

Runde, S., Fay, A. & Wutzke, W.-O. (2009). Knowledge-based requirement-engineering of building automation systems by means of semantic web technologies. In *2009 7th ieee international conference on industrial informatics* (pp. 267–272).

Sakarovitch, J. (2009). *Elements of automata theory*. Cambridge University Press.

Sarkar, P. (2000). A brief history of cellular automata. *Acm computing surveys (csur)*, *32*(1), 80–107.

Schedel, R. (2008). Darpa urban challenge 2007. *ATZ worldwide*, *110*(1), 10–12.

Schlegel, R., Obermeier, S. & Schneider, J. (2017). A security evaluation of iec 62351. *Journal of Information Security and Applications*, *34*, 197–204.

Schmitt, C. & Liggesmeyer, P. (2015). Getting grip on security requirements elicitation by structuring and reusing security requirements sources. *Complex Systems Informatics and Modeling Quarterly*(3), 15–34.

Schneider, K. (2007). Generating fast feedback in requirements elicitation. In *International working conference on requirements engineering: Foundation for software quality* (pp. 160–174).

Schulz, T., Griest, C., Golatowski, F. & Timmermann, D. (2018). Strategy for security certification of high assurance industrial automation and control systems. In *2018 ieee 13th international symposium on industrial embedded systems (sies)* (pp. 1–4).

SECURE, I. (2013). Establishment of isasecure japanese scheme and publication of isasecure embedded device security assurance certification program specifications in japan. *Last accessed July*, *29*.

Sepúlveda, S., Cravero, A. & Cachero, C. (2016). Requirements modeling languages for software product lines: A systematic literature review. *Information and Software Technology*, *69*, 16–36.

Shah, S. A. R. & Issac, B. (2018). Performance comparison of intrusion detection systems and application of machine learning to snort system. *Future Generation Computer Systems*, *80*, 157–170.

Shah, S. A. R., Issac, B. & Jacob, S. M. (2018). Intelligent intrusion detection system through combined and optimized machine learning. *International Journal of Computational Intelligence and Applications*, *17*(02), 1850007.

Sharma, C. (2020). Flux: From sql to gql query translation tool. In *2020 35th ieee/acm international conference on automated software engineering (ase)* (pp. 1379–1381).

Sharma, C. & Sinha, R. (2019). A schema-first formalism for labeled property graph databases: Enabling structured data loading and analytics. In *Proceedings of the 6th ieee/acm international conference on big data computing, applications and technologies* (pp. 71–80).

Sharma, C., Sinha, R. & Johnson, K. (2021). Practical and comprehensive formalisms

for modeling contemporary graph query languages. *Information Systems, Elsevier*, 101816. doi: https://doi.org/10.1016/j.is.2021.101816

Sharma, C., Sinha, R. & Leitao, P. (2019). Iaselect: Finding best-fit agent practices in industrial cps using graph databases. In *2019 ieee 17th international conference on industrial informatics (indin)* (Vol. 1, pp. 1558–1563).

Sharma, S. & Pandey, S. K. (2014). Requirements elicitation: Issues and challenges. In *2014 international conference on computing for sustainable global development (indiacom)* (pp. 151–155).

Singh, D., Tripathi, G. & Jara, A. J. (2014). A survey of internet-of-things: Future vision, architecture, challenges and services. In *2014 ieee world forum on internet of things (wf-iot)* (pp. 287–292).

Singh, N. K., Aït-Ameur, Y., Pantel, M., Dieumegard, A. & Jenn, E. (2016). Stepwise formal modeling and verification of self-adaptive systems with event-b. the automatic rover protection case study. In *2016 21st international conference on engineering of complex computer systems (iceccs)* (pp. 43–52).

Sinha, R., Dowdeswell, B., Zhabelova, G. & Vyatkin, V. (2018). TORUS: Scalable Requirements Traceability for Large-Scale Cyber-Physical Systems. *ACM Transactions on Cyber-Physical Systems*, *3*(2), 15.

Smith, G. (2012). *The object-z specification language* (Vol. 1). Springer Science & Business Media.

Snook, C. & Butler, M. (2006). Uml-b: Formal modeling and design aided by uml. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, *15*(1), 92–122.

Snort - network intrusion detection and prevention system. (n.d.). Retrieved Accessed: 2019-01-23, from `https://www.snort.org/`

Sommerville, I. (2011). Software engineering 9th edition. *ISBN-10*, *137035152*, 18.

Souag, A., Mazo, R., Salinesi, C. & Comyn-Wattiau, I. (2016). Reusable knowledge in security requirements engineering: a systematic mapping study. *Requirements Engineering*, *21*(2), 251–283.

Srinivas, J., Das, A. K. & Kumar, N. (2019). Government regulations in cyber security: Framework, standards and recommendations. *Future Generation Computer Systems*, *92*, 178–188.

Staggs, J., Ferlemann, D. & Shenoi, S. (2017). Wind farm security: attack surface, targets, scenarios and mitigation. *International Journal of Critical Infrastructure Protection*, *17*, 3–14.

Stouffer, K. A., Falco, J. A. & Scarfone, K. A. (2011). *Sp 800-82. guide to industrial control systems (ics) security: Supervisory control and data acquisition (scada) systems, distributed control systems (dcs), and other control system configurations such as programmable logic controllers (plc).* National Institute of Standards & Technology.

Tanveer, A., Sinha, R. & Kuo, M. M. (2020). Secure links: Secure-by-design communications in iec 61499 industrial control applications. *IEEE Transactions on Industrial Informatics*.

Thramboulidis, K. (2013). Iec 61499 vs. 61131: A comparison based on misperceptions.

arXiv preprint arXiv:1303.4761.

Toval, A., Nicolás, J., Moros, B. & García, F. (2002). Requirements reuse for improving information systems security: a practitioner's approach. *Requirements Engineering*, *6*(4), 205–219.

Tuma, K., Sandberg, C., Thorsson, U., Widman, M., Herpel, T. & Scandariato, R. (2021). Finding security threats that matter: Two industrial case studies. *Journal of Systems and Software*, *179*, 111003.

Turan, M. S., McKay, K. A., Çalık, Ç., Chang, D. & Bassham, L. (2019). Status report on the first round of the nist lightweight cryptography standardization process. *National Institute of Standards and Technology, Gaithersburg, MD, NIST Interagency/Internal Rep.(NISTIR)*.

Uschold, M., Gruninger, M. et al. (1996). Ontologies: Principles, methods and applications. *TECHNICAL REPORT-UNIVERSITY OF EDINBURGH ARTIFICIAL INTELLIGENCE APPLICATIONS INSTITUTE AIAI TR*.

Van der Aalst, W. & Stahl, C. (2011). *Modeling business processes: a petri net-oriented approach*. MIT press.

Varela-Vaca, Á. J., Rosado, D. G., Sánchez, L. E., Gómez-López, M. T., Gasca, R. M. & Fernández-Medina, E. (2020). Definition and verification of security configurations of cyber-physical systems. In *Computer security* (pp. 135–155). Springer.

Vassilev, A. T., Feldman, L. & Witte, G. A. (2014). Cryptographic module validation program (cmvp).

Veichtlbauer, A., Ortmayer, M. & Heistracher, T. (2017). Opc ua integration for field devices. In *2017 ieee 15th international conference on industrial informatics (indin)* (pp. 419–424).

Vistbakka, I. & Troubitsyna, E. (2021). Deriving implicit security requirements in safety-explicit formal development of control systems. In *Implicit and explicit semantics integration in proof-based developments of discrete systems* (pp. 109–130). Springer.

Vyatkin, V. (2009). The iec 61499 standard and its semantics. *IEEE Industrial Electronics Magazine*, *3*(4), 40–48.

Vyatkin, V., Hanisch, H.-M., Pang, C. & Yang, C.-H. (2008). Closed-loop modeling in future automation system engineering and validation. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *39*(1), 17–28.

Vyatkin, V. & of America, I. S. (2007). Iec 61499 function blocks for embedded and distributed control systems design.

Wagner, M. (2016). The hard truth about hardware in cyber-security: it's more important. *Network Security*, *2016*(12), 16–19.

Wang, J. (2007). Petri nets for dynamic event-driven system modeling. *Handbook of Dynamic System Modeling*, *1*.

Wang, R., Wang, S., Tang, C., Zhao, X., Hu, W., Tan, M. & Gao, B. (2016). Hybrid wireless sensor network for rescue site monitoring after earthquake. *Journal of Applied Remote Sensing*, *10*(3), 036020.

Wang, W. & Lu, Z. (2013). Cyber security in the smart grid: Survey and challenges.

Computer networks, *57*(5), 1344–1371.

Wang, W., Niu, N., Alenazi, M. & Da Xu, L. (2018). In-place traceability for automated production systems: A survey of plc and sysml tools. *IEEE Transactions on Industrial Informatics*, *15*(6), 3155–3162.

Wang, Z., Chen, C.-H., Zheng, P., Li, X. & Khoo, L. P. (2019). A novel data-driven graph-based requirement elicitation framework in the smart product-service system context. *Advanced engineering informatics*, *42*, 100983.

Wenger, M. & Zoitl, A. (2012). Re-use of iec 61131-3 structured text for iec 61499. In *2012 ieee international conference on industrial technology* (pp. 78–83).

Weyns, D., Iftikhar, M. U., De La Iglesia, D. G. & Ahmad, T. (2012). A survey of formal methods in self-adaptive systems. In *Proceedings of the fifth international c* conference on computer science and software engineering* (pp. 67–79).

Wieringa, R., Maiden, N., Mead, N. & Rolland, C. (2006). Requirements engineering paper classification and evaluation criteria: a proposal and a discussion. *Requirements engineering*, *11*(1), 102–107.

Yang, Y., Lu, J., Choo, K.-K. R. & Liu, J. K. (2015). On lightweight security enforcement in cyber-physical systems. In *Lightweight cryptography for security and privacy* (pp. 97–112).

Yap, C. N. & Holcombe, M. (1997). Using graphical icons to build z specifications. *Proceedings of the 2nd BCS-FACS Northern Formal Methods 2*, 1–15.

Yi, C.-G. & Kim, Y.-G. (2021). Security testing for naval ship combat system software. *IEEE Access*, *9*, 66839–66851.

Yılmaz, E. N. & Gönen, S. (2018). Attack detection/prevention system against cyber attack in industrial control systems. *Computers & Security*, *77*, 94–105.

Ylmaz, E. N., Ciylan, B., Gönen, S., Sindiren, E. & Karacayılmaz, G. (2018). Cyber security in industrial control systems: Analysis of dos attacks against plcs and the insider effect. In *2018 6th international istanbul smart grids and cities congress and fair (icsg)* (pp. 81–85).

Yu, X. & Xue, Y. (2016). Smart grids: A cyber–physical systems perspective. *Proceedings of the IEEE*, *104*(5), 1058–1070.

Zahid, F., Tanveer, A., Kuo, M. M. & Sinha, R. (2021). A systematic mapping of semi-formal and formal methods in requirements engineering of industrial cyber-physical systems. *Journal of Intelligent Manufacturing*, 1–36.

Zhabelova, G., Patil, S., Yang, C.-w. & Vyatkin, V. (2013). Smart grid applications with iec 61499 reference architecture. In *2013 11th ieee international conference on industrial informatics (indin)* (pp. 458–463).

Zhabelova, G. & Vyatkin, V. (2015). Towards software metrics for evaluating quality of iec 61499 automation software. In *2015 ieee 20th conference on emerging technologies & factory automation (etfa)* (pp. 1–8).

Zhabelova, G., Yang, C.-w., Vyatkin, V., Etherden, N. & Christoffersson, L. (2016). Open architecture for cost effective protection and control of power distribution networks. In *2016 ieee international conference on smart grid communications (smartgridcomm)* (pp. 729–735).

Zhang, J. & Cheng, B. H. (2006). Using temporal logic to specify adaptive program

semantics. *Journal of Systems and Software*, *79*(10), 1361–1369.

Zhou, C., Hu, B., Shi, Y., Tian, Y.-C., Li, X. & Zhao, Y. (2020). A unified architectural approach for cyberattack-resilient industrial control systems. *Proceedings of the IEEE*.

Appendix A

Systematic Mapping Study—Primary Studies

The following is the list of primary studies selected during the systematic mapping study presented in Section 2.2.

S[1] Feo-Arenis, S., Westphal, B., Dietsch, D., Muñiz, M., Andisha, S., & Podelski, A. (2016). Ready for testing: ensuring conformance to industrial standards through formal verification. Formal Aspects of Computing, 28(3), 499527. article. http://doi.org/10.1007/s00165-016-0365-3

S[2] Mateescu, F. J. L. (2016). Formal modelling and verification of GALS systems using GRL and CADP. Formal Aspects of Computing, 0(0).

S[3] Sinha, R., Pang, C., Martínez, G. S., & Vyatkin, V. (2016). Automatic test case generation from requirements for industrial cyber-physical systems. At - Automatisierungstechnik, 64(3), 216230. article. http://doi.org/10.1515/auto-2015-0075

S[4] Wu, D., & Schnieder, E. (2016). Scenario-based system design with colored Petri nets: an application to train control systems. Software and Systems Modeling, 123. article. http://doi.org/10.1007/s10270-016-0517-1

S[5] Zafar, N. A. (2016). Formal specification and analysis of take-off procedure

using VDM-SL. Complex Adaptive Systems Modeling, 4(1).

S[6] Ahmad, E., Dong, Y., Larson, B., Lü, J., Tang, T., & Zhan, N. (2015). Behavior modeling and verification of movement authority scenario of Chinese Train Control System using AADL. Science China Information Sciences. article.

S[7] Besnard, L., Bouakaz, A., Gautier, T., Le Guernic, P., Ma, Y., Talpin, J.-P., & Yu, H. (2015). Timed behavioural modelling and affine scheduling of embedded software architectures in the AADL using Polychrony. Science of Computer Programming, 106, 5477. article. http://doi.org/10.1016/j.scico.2014.05.014

S[8] Bouskela, D., Nguyen, T., & Jardin, A. (2015). Towards a rigorous approach for verifying cyber-physical systems against requirements. In Electrical Power and Energy Conference (EPEC), 2015 IEEE (pp. 250255).

S[9] Deng, P., Ren, G., Yuan, W., Chen, F., & Hua, Q. (2015). An integrated framework of formal methods for interaction behaviors among industrial equipments. Microprocessors and Microsystems. article. http://doi.org/10.1016/j.micpro.2015.07.015

S[10] Dordowsky, F. (2015). An experimental Study using ACSL and Frama-C to formulate and verify Low-Level Requirements from a DO-178C compliant avionics project. In Electronic Proceedings in Theoretical Computer Science, EPTCS (Vol. 187, pp. 2841). conference. http://doi.org/10.4204/EPTCS.187.3

S[11] Baouya, A., Bennouar, D., Mohamed, O. A., & Ouchani, S. (2015). A quantitative verification framework of SysML activity diagrams under time constraints. Expert Systems with Applications, 42(21), 74937510.

S[12] Ebeid, E., Fummi, F., & Quaglia, D. (2015). Model-Driven Design of Network Aspects of Distributed Embedded Systems. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 34(4), 603614.

S[13] Eiza, M. H., Randles, M., Johnson, P., Shone, N., Pang, J., & Bhih, A. (2015). Rail Internet of Things: An architectural platform and assured requirements model. In Proceedings - 15th IEEE International Conference on Computer and Information

Technology, CIT 2015

S[14] Geetha Lekshmy, V., & Bhaskar, J. (2015). Programming smart environments using p-calculus. In Procedia Computer Science (Vol. 46, pp. 884891). conference. http://doi.org/10.1016/j.procs.2015.02.158

S[15] Greenyer, J., & Haase, M. (2015). Evaluating a Formal Scenario-Based Method for the Requirements Analysis in Automotive Software Engineering Categories and Subject Descriptors. 10th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering BERGAMO, ITALY, August 30 September 4, 10021005.

S[16] Hamadou, S., Mullins, J., Chareton, C., & Gherbi, A. (2015). Specifying Avionic Embedded Systems by Denotations of the Time-Triggered Constraint-Based Calculus. In Proceedings - 2015 IEEE 16th International Conference on Information Reuse and Integration, IRI 2015 (pp. 303310). conference.

S[17] Hissam, S. A., Chaki, S., & Moreno, G. A. (2015). High assurance for distributed cyber physical systems. In ACM ECSAW 15 Proceedings of the 2015 European Conference on Software Architecture Workshops (Vol. 0711Sept). conference. http://doi.org/10.1145/2797433.2797439

S[18] Ikram, A., Anjum, A., Hill, R., Antonopoulos, N., Liu, L., & Sotiriadis, S. (2015). Approaching the Internet of things (IoT): A modelling, analysis and abstraction framework. Concurrency Computation, 27(8), 19661984. article. http://doi.org/10.1002/cpe.3131

S[19] Kim, J. H., Larsen, K. G., Nielsen, B., Mikucionis, M., & Olsen, P. (2015). Formal analysis and testing of real-time automotive systems using UPPAAL tools. In Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) (Vol. 9128, pp. 4761). article. http://doi.org/10.1007/978-3-319-19458-5_4

S[20] Liu, W., & Li, M. (2015). Requirements Planning with Event Calculus

for Runtime Self-Adaptive System. 2015 IEEE 39th Annual Computer Software and Applications Conference, 7782. http://doi.org/10.1109/COMPSAC.2015.18

S[21] Mahmud, N., Seceleanu, C., & Ljungkrantz, O. (2015). ReSA: An ontology-based requirement specification language tailored to automotive systems. In 10th IEEE International Symposium on Industrial Embedded Systems (SIES) (pp. 110). inproceedings. http://doi.org/10.1109/SIES.2015.7185035

S[22] Marko, N., Leitner, A., Herbst, B., & Wallner, A. (2015). Combining Xtext and OSLC for Integrated Model-Based Requirements Engineering. In Proceedings - 41st Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2015 (pp. 143150). conference. http://doi.org/10.1109/SEAA.2015.11

S[23] Mazzini, S., Favaro, J., & Baracchi, L. (2015). A Model-Based Approach Across the IoT Lifecycle for Scalable and Distributed Smart Applications. In IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC (Vol. 2015Octob, pp. 149154). conference. http://doi.org/10.1109/ITSC.2015.33

S[24] Mosterman, Pieter J Zander, J. (2015). Cyber-physical systems challenges : a needs analysis for collaborating embedded software systems. Software & Systems Modeling, 15(1), 516. http://doi.org/10.1007/s10270-015-0469-x

S[25] Ainhauser, R. F. B. T. B. (2014). Automatically Generated Safety Mechanisms from Semi-Formal Software Safety Requirements. In Computer Safety, Reliability, and Security (Vol. 0). inproceedings. http://doi.org/10.1007/978-3-319-10506-2_19

S[26] Arts, T., Dorigatti, M., & Tonetta, S. (2014). Making implicit safety requirements explicit: An AUTOSAR safety case. SAFECOMP 2014 Proceedings of the 33rd International Conference on Computer Safety, Reliability, and Security - Volume 8666, 8666 LNCS, 8192. article. http://doi.org/10.1007/978-3-319-10506-2_6

S[27] Ayed, R. Ben, Collart-Dutilleul, S., Bon, P., Idani, A., & Ledru, Y. (2014). B formal validation of ERTMS/ETCS railway operating rules. In Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and

Lecture Notes in Bioinformatics) 4th International Conference, ABZ 2014, Toulouse, France, June 2-6, 2014. Proceedings (Vol. 8477 LNCS, pp. 124129). inproceedings. http://doi.org/10.1007/978-3-662-43652-3_10

S[28] Blackburn, M., & Denno, P. (2014). Virtual design and verification of cyber-physical systems: Industrial process plant design. In Procedia Computer Science (Vol. 28, pp. 883890). conference. http://doi.org/10.1016/i.procs.2014.03.006

S[29] Che, X., & Maag, S. (2014). Testing protocols in Internet of Things by a formal passive technique. Science China Information Sciences, 57(3), 113. article. http://doi.org/10.1007/s11432-014-5068-x

S[30] Daoud, H., Tanougast, C., Belarbi, M., & Heil, M. (2014). Formal specification and verification of wireless networked self-organized Systems on Chip. In Control, Decision and Information Technologies (CoDIT), 2014 International Conference on (pp. 730735). inproceedings. http://doi.org/10.1109/CoDIT.2014.6996987

S[31] Filipovikj, P., Nyberg, M., & Rodriguez-Navas, G. (2014). Reassessing the pattern-based approach for formalizing requirements in the automotive domain. 2014 IEEE 22nd International Requirements Engineering Conference, RE 2014 - Proceedings, 444450. http://doi.org/10.1109/RE.2014.6912296

S[32] Fockel, M., & Holtmann, J. (2014). A requirements engineering methodology combining models and controlled natural language. In Model-Driven Requirements Engineering Workshop (MoDRE), 2014 IEEE 4th International (pp. 6776). inproceedings. http://doi.org/10.1109/MoDRE.2014.6890827

S[33] Koch, T., Holtmann, J., & Deantoni, J. (2014). Generating EAST-ADL event chains from scenario-based requirements specifications. 8th European Conference, ECSA 2014, Vienna, Austria, August 25-29, 2014. Proceedings, 8627 LNCS, 146153. article. http://doi.org/10.1007/978-3-319-09970-5_14

S[34] Nuzzo, P., Finn, J. B., Iannopollo, A., & Sangiovanni-Vincentelli, A. L. (2014). Contract-based design of control protocols for safety-critical cyber-physical

systems. In 2014 Design, Automation Test in Europe Conference Exhibition (DATE) (pp. 14). inproceedings. http://doi.org/10.7873/DATE.2014.072

S[35] Sampaio, G. C. C. R. C. (2014). A Formal Model for Natural-Language Timed Requirements of Reactive Systems. In 16th International Conference on Formal Engineering Methods, ICFEM 2014, Luxembourg, Luxembourg, November 3-5, 2014. Proceedings (Vol. 0). inproceedings. http://doi.org/10.1007/978-3-319-11737-9_4

S[36] Sena Marques, M. R., Siegert, E., & Brisolara, L. (2014). Integrating UML, MARTE and sysml to improve requirements specification and traceability in the embedded domain. In Proceedings - 2014 12th IEEE International Conference on Industrial Informatics, INDIN 2014 (pp. 176181).

S[37] Suryadevara, J., Sapienza, G., Seceleanu, C., Seceleanu, T., Ellevseth, S.-E., & Pettersson, P. (2014). Wind Turbine System: An Industrial Case Study in Formal Modeling and Verification. In LNCS: Formal Techniques for Safety-Critical Systems (Vol. 0, pp. 229245). inproceedings. http://doi.org/10.1007/978-3-319-05416-2_15

S[38] Voisin, R. A. (2014). Formal Implementation of Data Validation for Railway Safety-Related Systems with OVADO. In Software Engineering and Formal Methods (Vol. 0). inproceedings. http://doi.org/10.1007/978-3-319-05032-4_17

S[39] Westman, J., & Nyberg, M. (2014). Specifying and Structuring Requirements on Cyber-Physical Systems using Contracts, 135.

S[40] Zhang, L. (2014). Modeling large scale complex cyber physical control systems based on system of systems engineering approach. In ICAC 2014 - Proceedings of the 20th International Conference on Automation and Computing: Future Automation, Computing and Manufacturing (pp. 5560). conference.

S[41] Zhao, Y., & Rozier, K. Y. (2014). Formal specification and verification of a coordination protocol for an automated air traffic control system. Science of Computer Programming, 96(P3), 337353. http://doi.org/10.1016/j.scico.2014.04.002

S[42] Zhou, J. (2014). Requirements development and management of embedded

real-time systems. In 2014 IEEE 22nd International Requirements Engineering Conference (RE) (pp. 479484). inproceedings. http://doi.org/10.1109/RE.2014.6912302

S[43] Carvalhaes, M. F. A., d. S. Neto, O. C., Ferreira, J. O., da Rocha, A. F., & d. A. Barbosa, T. M. G. (2013). Including affectivity requirements in embedded systems. In Systems Conference (SysCon), 2013 IEEE International (pp. 229234). inproceedings. http://doi.org/10.1109/SysCon.2013.6549887

S[44] Chen, D., Feng, L., Qureshi, T. N., Lönn, H., & Hagl, F. (2013). An architectural approach to the analysis, verification and validation of software intensive embedded systems. Computing, 95(8), 649688. article. http://doi.org/10.1007/s00607-013-0314-4

S[45] Hazra, A., Ghosh, P., Vadlamudi, S. G., Chakrabarti, P. P., & Dasgupta, P. (2013). Formal methods for early analysis of functional reliability in component-based embedded applications. IEEE Embedded Systems Letters, 5(1), 811. article. http://doi.org/10.1109/LES.2013.2239605

S[46] Mangeruca, L., Ferrante, O., & Ferrari, A. (2013). Formalization and completeness of evolving requirements using Contracts. In 2013 8th IEEE International Symposium on Industrial Embedded Systems (SIES) (pp. 120129). inproceedings. http://doi.org/10.1109/SIES.2013.6601484

S[47] Meseguer, J. (2013). Taming distributed system complexity through formal patterns. Science of Computer Programming (2014), 7253 LNCS, 12. article.

S[48] Ordinez, L., Alimenti, O., Rinland, E., Gomez, M., & Marchetti, J. (2013). Modeling and specifying requirements for cyber-physical systems. IEEE Latin America Transactions, 11(1), 625632. article. http://doi.org/10.1109/TLA.2013.6502874

S[49] Zhang, L. (2013). Aspect-oriented modeling for railway control systems. In 2013 IEEE International Conference on Information and Automation, ICIA 2013 (pp. 236241). conference. http://doi.org/10.1109/ICInfA.2013.6720302

S[50] Zhang, L. (2013). Specifying and modeling automotive cyber physical systems. In Proceedings - 16th IEEE International Conference on Computational Science and Engineering, CSE 2013 (pp. 603610).

S[51] Zhang, L. (2013). Requirement Specification for Transportation Cyber Physical Systems. In Green Computing and Communications (GreenCom), 2013 IEEE and Internet of Things (iThings/CPSCom), IEEE International Conference on and IEEE Cyber, Physical and Social Computing (pp. 14861491). inproceedings. http://doi.org/10.1109/GreenCom-iThings-CPSCom.2013.262

S[52] Zhang, L. (2013). Modeling railway cyber physical systems based on AADL. In Automation and Computing (ICAC), 2013 19th International Conference on (pp. 16). inproceedings.

S[53] Zhang, L. (2013). Requirement Analysis Method for Vehicular Cyber Physical Systems. 2013 IEEE 10th International Conference on High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing, 20962103. inproceedings.

S[54] Barnat, J., Beran, J., Brim, L., Kratochvíla, T., & Rockai, P. (2012). Tool chain to support automated formal verification of avionics simulink designs. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 17th International Workshop, FMICS 2012, Paris, France, August 27-28, 2012. Proceedings, 7437 LNCS, 7892. article. http://doi.org/10.1007/978-3-642-32469-7_6

S[55] Hazra, A., Ghosh, P., & Dasgupta, P. (2012). Reliability annotations to formal specifications of context-sensitive safety properties in embedded systems. In Forum on Specification and Design Languages (pp. 3643). conference.

S[56] Kim, M., Stehr, M.-O., Talcott, C., Dutt, N., & Venkatasubramanian, N. (2012). xTune: A Formal Methodology for Cross-Layer Tuning of Mobile Embedded Systems. ACM Transactions on Embedded Computing Systems, 11(4), 123.

http://doi.org/10.1145/2362336.2362340

S[57] Lijie, C., Tao, T., Xianqiong, Z., & Schnieder, E. (2012). Verification of the safety communication protocol in train control system using colored Petri net. Reliability Engineering and System Safety, 100, 818. http://doi.org/10.1016/j.ress.2011.12.010

S[58] Nicholas, K., Bhatti, Z. E., & Roop, P. S. (2012). Model-driven development of industrial embedded systems: Challenges faced and lessons learnt. Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies & Factory Automation (ETFA 2012), 14. conference. http://doi.org/10.1109/ETFA.2012.6489690

S[59] Ölveczky, J. M. C. (2012). Formalization and Correctness of the PALS Architectural Pattern for Distributed Real-Time Systems. In Theoretical Computer Science (Vol. 0). inproceedings. http://doi.org/10.1007/978-3-642-16901-4_21

S[60] Kinder, A. E. H. P. (2011). A formal approach for the construction and verification of railway control systems. Formal Aspects of Computing, 23(2). article. http://doi.org/10.1007/s00165-009-0143-6

S[61] Krüger, I., Farcas, C., Farcas, E., & Menarini, M. (2011). Requirements Modeling for Embedded Realtime Systems. In International Dagstuhl Workshop, Dagstuhl Castle, Germany, November 4-9, 2007. Revised Selected Papers (Vol. 6100, pp. 155199). article. http://doi.org/10.1007/978-3-642-16277-0_7

S[62] Schulz, Dominik Dietrich, L. S. (2011). Formalizing and Operationalizing Industrial Standards. In FASE11/ETAPS11 Proceedings of the 14th international conference on Fundamental approaches to software engineering: part of the joint European conferences on theory and practice of software (Vol. 0). inproceedings. http://doi.org/10.1007/978-3-642-19811-3_7

S[63] Siegl, S., Hielscher, K. S., German, R., & Berger, C. (2011). Formal specification and systematic model-driven testing of embedded automotive systems. In 2011 Design, Automation Test in Europe (pp. 16).

S[64] Wang, R., Song, X., Zhu, J., & Gu, M. (2011). Formal modeling and

synthesis of programmable logic controllers. Computers in Industry, 62(1), 2331. http://doi.org/10.1016/j.compind.2010.05.015

S[65] Zhang, L. (2011). Formal methods for aspect-oriented specification of cyber physical systems. Communications in Computer and Information Science (LNCS), International Conference, CSEE 2011, Wuhan, China, August 21-22, 2011. Proceedings, Part II, 215 CCIS(PART 2), 316322. article. http://doi.org/10.1007/978-3-642-23324-1_51

S[66] Cancila, D., Passerone, R., Vardanega, T., & Panunzio, M. (2010). Toward correctness in the specification and handling of non-functional attributes of high-integrity real-time embedded systems. IEEE Transactions on Industrial Informatics, 6(2), 181194. article. http://doi.org/10.1109/TII.2010.2043741

S[67] Coronato, A., & De Pietro, G. (2010). Formal specification of wireless and pervasive healthcare applications. ACM Transactions on Embedded Computing Systems, 10(1). article. http://doi.org/10.1145/1814539.1814551

S[68] Fuchs, A., Gürgens, S., Weber, D., Bodenstedt, C., & Ruland, C. (2010). Formalization of Smart Metering requirements. In ACM International Conference Proceeding Series. conference. http://doi.org/10.1145/1868433.1868439

S[69] Jaichandran, R. (2010). Specification Verification and Validation of Wireless Sensor Network Model for Environment Monitoring. Proceedings of the 1st International Conference on Wireless Technologies for Humanitarian Relief, 437440.

S[70] Ölveczky, P. C., & Meseguer, J. (2010). Specification and Verification of Distributed Embedded Systems: A Traffic Intersection Product Family. Electronic Proceedings in Theoretical Computer Science, 36, 137157. http://doi.org/10.4204/EPTCS.36.8

S[71] Whittle, J., Sawyer, P., Bencomo, N., Cheng, B. H. C., & Bruel, J.-M. (2010). RELAX: A language to address uncertainty in self-adaptive systems requirement. Requirements Engineering, 15(2), 177196.

S[72] Yang, N., Yu, H., Sun, H., & Qian, Z. (2010). Modeling UML sequence

diagrams using extended Petri nets. In 2010 International Conference on Information Science and Applications, ICISA 2010.

S[73] Andrade, E., Maciel, P., Callou, G., & Nogueira, B. (2009). A methodology for mapping SysML activity diagram to time petri net for requirement validation of embedded real-time systems with energy constraints. Proceedings of the 3rd International Conference on Digital Society, ICDS 2009, 266271. http://doi.org/10.1109/ICDS.2009.19

S[74] du BousquetMasahide NakamuraBen YanHiroshi Igaki, L. (2009). Using formal methods to increase confidence in a home network system implementation: a case study. Innovations in Systems and Software Engineering, 5(3). article. http://doi.org/10.1007/s11334-009-0092-5

S[75] Ishikawa, F., Suleiman, B., Yamamoto, K., & Honiden, S. (2009). Physical interaction in pervasive computing: Formal modeling, analysis and verification. In ICPS09 - Proceedings of the 2009 International Conference on Pervasive Services and Co-located Workshops (pp. 133140). conference.

S[76] Wu, J., & Yang, S. (2009). Process Algebra Approach to Verifying Safety Specification of Hybrid Embedded Systems. In 2009 International Conference on Communication Software and Networks (pp. 129133). conference.

S[77] Zhang, J., Goldsby, H. J., & Cheng, B. H. C. (2009). Modular verification of dynamically adaptive systems. In Proceedings of the 8th ACM International Conference on Aspect-Oriented Software Development, AOSD09 (pp. 161172). conference. http://doi.org/10.1145/1509239.1509262

S[78] Zhang, L. (2009). Aspect-Oriented Formal Specification for Real-Time Systems. In Advances in Computational Science and Engineering (Vol. 0). inproceedings. http://doi.org/10.1007/978-3-642-10238-7_2

S[79] Benvenuti, L., Ferrari, A., Mangeruca, L., Mazzi, E., Passerone, R., & Sofronis, C. (2008). A contract-based formalism for the specification of heterogeneous

systems. Proceedings - 2008 Forum on Specification, Verification and Design Languages, FDL08, 142147. http://doi.org/10.1109/FDL.2008.4641436

S[80] Boulanger, J. L., & Dao, V. Q. (2008). Requirements engineering in a model-based methodology for embedded automotive software. In Research, Innovation and Vision for the Future, 2008. RIVF 2008. IEEE International Conference on (pp. 263268). inproceedings. http://doi.org/10.1109/RIVF.2008.4586365

S[81] Hirsch, M., Missal, D., & Hanisch, H. M. (2008). Design and verification of distributed industrial manufacturing control systems. In Proceedings - 34th Annual Conference of the IEEE Industrial Electronics Society, IECON 2008 (pp. 152157). inproceedings. http://doi.org/10.1109/IECON.2008.4757944

S[82] Hooman, J., Kugler, H., Ober, I., Votintseva, A., & Yushtein, Y. (2008). Supporting UML-based development of embedded systems by formal techniques. Software and Systems Modeling, 7(2), 131155. article. http://doi.org/10.1007/s10270-006-0043-7

S[83] Jan Tretmans, Klaas Wijbrans, M. C. (2008). Software Engineering with Formal Methods: Experiences with the Development of a Storm Surge Barrier Control System. In FM 2008: Formal Methods (Vol. 0).

S[84] Markose, S., Liu, X., & McMillin, B. (2008). A Systematic framework for structured object-oriented security requirements analysis in embedded systems. In Proceedings of The 5th International Conference on Embedded and Ubiquitous Computing, EUC 2008 (Vol. 1, pp. 7581). conference. http://doi.org/10.1109/EUC.2008.92

S[85] Morimoto, S., Shigematsu, S., Goto, Y., & Cheng, J. (2008). Classification, formalization and verification of security functional requirements. ACM SOFSEM08 Proceedings of the 34th Conference on Current Trends in Theory and Practice of Computer Science, 4910 LNCS, 622633. article.

S[86] Nicolay, C. W. G. (2008). NQSL - Formal Language and Tool Support for Network Quality-of-Service Requirements. In Formal Techniques for Networked

and Distributed Systems FORTE 2008, Proceedings of the 28th IFIP WG6.1 International Conference on Formal Techniques for Networked and Distributed Systems (FORTE2008) (Vol. 0). inproceedings. http://doi.org/10.1007/978-3-540-68855-6_14

S[87] Zhang, J. S. D. S. S. T. (2008). Specifying and Verifying Sensor Networks: An Experiment of Formal Methods. In 10th International Conference on Formal Engineering Methods, ICFEM 2008, Kitakyushu-City, Japan, October 27-31, 2008. Proceedings (Vol. 0). inproceedings. http://doi.org/10.1007/978-3-540-88194-0_20

S[88] Chuanhui, L., Mei, R., & Guangquan, Z. (2007). A scenario-based modeling method for component-based embedded software. In 2007 International Conference on Convergence Information Technology, ICCIT 2007 (pp. 343346). conference. http://doi.org/10.1109/ICCIT.2007.4420283

S[89] da CostaLeila Ribeiro, L. M. A. (2007). Formal Specification and Verification of Real-Time Systems using Graph Grammars. Journal of the Brazilian Computer Society, 13(4). article. http://doi.org/10.1007/BF03194256

S[90] del BiancoLuigi LavazzaMarco MauriGiuseppe Occorso, V. (2007). Towards UML-based formal specifications of component-based real-time software. International Journal on Software Tools for Technology Transfer, 9(2). article.

S[91] Lundqvist, M. O. (2007). The TASM Toolset: Specification, Simulation, and Formal Verification of Real-Time Systems. In Computer Aided Verification 19th International Conference, CAV 2007, Berlin, Germany, July 3-7, 2007. Proceedings (Vol. 0). inproceedings. http://doi.org/10.1007/978-3-540-73368-3_15

S[92] Piotrowicz, M., Slusarczyk, K., & Napieraiski, A. (2007). A Coloured PETRI Net Based Solution for the Generalized Railway Crossing Problem. In 2007 14th International Conference on Mixed Design of Integrated Circuits and Systems (pp. 657660). inproceedings. http://doi.org/10.1109/MIXDES.2007.4286245

S[93] Venkatasubramanian, M. K.-O. S. T. D. (2007). A Probabilistic Formal Analysis Approach to Cross Layer Optimization in Distributed Embedded Systems.

In Formal Methods for Open Object-Based Distributed Systems (LNCS), 9th IFIP WG 6.1 International Conference, FMOODS 2007, Paphos, Cyprus, June 6-8, 2007. Proceedings (Vol. 0). inproceedings. http://doi.org/10.1007/978-3-540-72952-5_18

S[94] Wang, R., Song, X., & Gu, M. (2007). Modelling and verification of program logic controllers using timed automata. IET Software, 1(4), 127131. article. http://doi.org/10.1049/iet-sen:20070009

S[95] Andrei, S., & Cheng, A. M. K. (2006). Faster verification of RTL-specified systems via decomposition and constraint extension. In Proceedings - Real-Time Systems Symposium (pp. 6776). conference. http://doi.org/10.1109/RTSS.2006.23

S[96] Hinchey, M. G., Rash, J. L., Rouff, C. A., & Gracanin, D. (2006). Achieving dependability in sensor networks through automated requirements-based programming. Computer Communications, 29(2), 246256. article.

S[97] Malik, L. L. T. L. L. A. (2006). Formal Service-Oriented Development of Fault Tolerant Communicating Systems. In TUCS Technical Reports (Vol. 0). inproceedings. http://doi.org/10.1007/11916246_14

S[98] Norström, A. W. A. (2006). Decreasing Maintenance Costs by Introducing Formal Analysis of Real-Time Behavior in Industrial Settings. In ISoLA04 Proceedings of the First international conference on Leveraging Applications of Formal Methods (Vol. 0). inproceedings. http://doi.org/10.1007/11925040_9

S[99] Radojevic, I., Salcic, Z., & Roop, P. S. (2006). Modeling Embedded Systems: From SystemC and Esterel to DFCharts. IEEE Design Test of Computers, 23(5), 348358. article. http://doi.org/10.1109/MDT.2006.130

S[100] Apvrille, L., Li, L., & Roudier, Y. (2016). Model-Driven Engineering for Designing Safe and Secure Embedded Systems. In Proceedings - 2016 Architecture-Centric Virtual Integration, ACVI 2016 (pp. 47). http://doi.org/10.1109/ACVI.2016.6

S[101] Arnold, A., Baleani, M., Ferrari, A., Marazza, M., Senni, V., Legay, A., Etzien, C. (2016). An Application of SMC to Continuous Validation of Heterogeneous

Systems. In Proceedings of the 9th EAI International Conference on Simulation Tools and Techniques (pp. 7685). ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering). Retrieved from http://dl.acm.org/citation.cfmid=3021426.3021438

S[102] Cabodi, G., Camurati, P., Finocchiaro, S. F., Loiacono, C., Savarese, F., & Vendraminetto, D. (2016). Secure embedded architectures: Taint properties verification. In 2016 International Conference on Development and Application Systems (DAS) (pp. 150157). http://doi.org/10.1109/DAAS.2016.7492565

S[103] Coletta, A., & Armando, A. (2016). Security monitoring for industrial control systems. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 9588, 4862. http://doi.org/10.1007/978-3-319-40385-4_4

S[104] Darvas, D., Majzik, I., & Vinuela, E. B. (2016). Conformance checking for programmable logic controller programs and specifications. In 2016 11th IEEE International Symposium on Industrial Embedded Systems, SIES 2016 - Proceedings. http://doi.org/10.1109/SIES.2016.7509409

S[105] Fockel, M. (2016). ASIL tailoring on functional safety requirements. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 9923 LNCS, 298310.

S[106] Harland, J., Blech, J. O., Peake, I., & Trodd, L. (2016). Formal behavioural models to facilitate distributed development and commissioning in industrial automation. In ENASE 2016 - Proceedings of the 11th International Conference on Evaluation of Novel Software Approaches to Software Engineering (pp. 363369).

S[107] He, X., & Fu, Y. (2016). Modeling and analyzing security patterns using high level petri nets. In Proceedings of the International Conference on Software Engineering and Knowledge Engineering, SEKE (Vol. 2016Janua, pp. 623627). Retrieved from

S[108] Johnsen, A., Lundqvist, K., Hanninen, K., Pettersson, P., & Torelm, M. (2016). AQAF: An architecture quality assurance framework for systems modeled in AADL. In Proceedings - 2016 12th International ACM SIGSOFT Conference on Quality of Software Architectures, QoSA 2016 (pp. 3140). http://doi.org/10.1109/QoSA.2016.9

S[109] Kim, J. H., Kang, I., Kang, S., & Boudjadar, A. (2016). A Process Algebraic Approach to Resource-Parameterized Timing Analysis of Automotive Software Architectures. IEEE Transactions on Industrial Informatics, 12(2), 655671. http://doi.org/10.1109/TII.2016.2527624

S[110] Musat, L., Kandl, S., Puschner, P., Hubl, M., Buzo, A., & Pelz, G. (2016). Requirement semi-formalization methodology for SoC design. In ISOCC 2015 - International SoC Design Conference: SoC for Internet of Everything (IoE) (pp. 910). http://doi.org/10.1109/ISOCC.2015.7401686

S[111] Nam, M.-Y., Delange, J., & Feiler, P. (2016). Integrated modeling workflow for security assurance. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 9952 LNCS, 926941. http://doi.org/10.1007/978-3-319-47166-2_64

S[112] Petnga, L., & Austin, M. (2016). An ontological framework for knowledge modeling and decision support in cyber-physical systems. Advanced Engineering Informatics, 30(1), 7794. http://doi.org/10.1016/j.aei.2015.12.003

S[113] Sun, H., Liu, J., Chen, X., & Du, D. (2016). Specifying cyber physical system safety properties with metric temporal spatial logic. In Proceedings - Asia-Pacific Software Engineering Conference, APSEC (Vol. 2016May, pp. 254260). http://doi.org/10.1109/APSEC.2015.58

S[114] Dragomir, I., Ober, I., & Percebois, C. (2017). Contract-based modeling and verification of timed safety requirements within SysML. Software and Systems Modeling, 16(2), 587624.

S[115] Elleuch, M., & Hasan, O. (2017). Formal Probabilistic Analysis of a WSN-Based Monitoring Framework for IoT Applications, 3, 93108.

S[116] Kantaros, Y., & Zavlanos, M. M. (2017). Sampling-based Control Synthesis for Multi-robot Systems Under Global Temporal Specifications. In Proceedings of the 8th International Conference on Cyber-Physical Systems (pp. 313). New York, NY, USA: ACM. https://doi.org/10.1145/3055004.3055027

S[117] Kirov, D., Nuzzo, P., Passerone, R., & Sangiovanni-Vincentelli, A. (2017). ArchEx: An Extensible Framework for the Exploration of Cyber-Physical System Architectures. In Proceedings of the 54th Annual Design Automation Conference 2017 (p. 31:1–31:6). New York, NY, USA: ACM. https://doi.org/10.1145/3061639.3062204

S[118] Singh, N. K., Yamine, A., Pantel, M., Dieumegard, A., & Jenn, E. (2016). Stepwise Formal Modeling and Verification of Self-Adaptive systems with Event-B . The Automatic Rover Protection case study. https://doi.org/10.1109/ICECCS.2016.10

S[119] Xu, G., Yu, W., Griffith, D., Golmie, N., & Moulema, P. (2017). Toward Integrating Distributed Energy Resources and Storage Devices in Smart Grid, 4(1), 192204.

S[120] Seceleanu, C., Johansson, M., Suryadevara, J., Sapienza, G., Seceleanu, T., Ellevseth, S., & Pettersson, P. (2017). Analyzing a wind turbine system : From simulation to formal verification. Science of Computer Programming, 133, 216242. https://doi.org/10.1016/j.scico.2016.09.007

S[121] Borgne, A. Le, & Belloir, N. (2016). Formal Requirements Engineering for Smart Industries Toward a Model-Based Graphical Language, 10281032. https://doi.org/10.1109/UIC-ATC-ScalCom-CBDCom-IoP-SmartWorld.2016.134

S[122] Ruan, C., & Wang, J. (2016). The Timed Abstract State Machine based test requirement auto generation for embedded systems. https://doi.org/10.1109/iThings-GreenCom-CPSCom-SmartData.2016.80

S[123] Marinescu, R., Enoiu, E., Seceleanu, C., & Sundmark, D. (2017). Automatic Test Generation for Energy Consumption of Embedded Systems Modeled in EAST-ADL. In Proceedings - 10th IEEE International Conference on Software Testing, Verification and Validation Workshops, ICSTW 2017 (pp. 6976). Institute of Electrical and Electronics Engineers Inc.

S[124] Dowdeswell, B., Sinha, R., & Haemmerle, E. (2016). TORUS : Tracing Complex Requirements for Large Cyber-Physical Systems.

S[125] Howard, G., Butler, M., Colley, J., & Sassone, V. (2017). Formal analysis of safety and security requirements of critical systems supported by an extended STPA methodology. Proceedings - 2nd IEEE European Symposium on Security and Privacy Workshops, EuroS and PW 2017, 174–180. https://doi.org/10.1109/EuroSPW.2017.68

S[126] Gawanmeh, A., Alwadi, A., & Parvin, S. (2017). Formal Verification of Control Strategies for a Cyber Physical System. Proceedings - IEEE 37th International Conference on Distributed Computing Systems Workshops, ICDCSW 2017, 91–96. https://doi.org/10.1109/ICDCSW.2017.59

S[127] Wang, T., Su, Q., & Chen, T. (2017). Formal Analysis of Security Properties of Cyber-Physical System Based on Timed Automata. Proceedings - 2017 IEEE 2nd International Conference on Data Science in Cyberspace, DSC 2017, 534–540. https://doi.org/10.1109/DSC.2017.44

S[128] Zahra, S., Alam, M., Javaid, Q., Wahid, A., Javaid, N., Malik, S. U. R., & Khan, M. K. (2017). Fog Computing over IoT: A Secure Deployment and Formal Verification. IEEE Access, 5, 27132–27144. https://doi.org/10.1109/ACCESS.2017.2766180

S[129] Dokhanchi, A., Hoxha, B., & Fainekos, G. (2017). Formal requirement debugging for testing and verification of cyber-physical systems. ACM Transactions on Embedded Computing Systems, 17(2). https://doi.org/10.1145/3147451

S[130] Jeannin, J. B., Ghorbal, K., Kouskoulas, Y., Schmidt, A., Gardner, R., Mitsch, S., & Platzer, A. (2017). A formally verified hybrid system for safe advisories

in the next-generation airborne collision avoidance system. International Journal on Software Tools for Technology Transfer, 19(6), 717–741. https://doi.org/10.1007/s10009-016-0434-1

S[131] Zafar, N. A., & Afzaal, H. (2017). Formal model of earthquake disaster mitigation and management system. Complex Adaptive Systems Modeling, 5(1). https://doi.org/10.1186/s40294-017-0049-8

S[132] Drozdov, D., Patil, S., & Vyatkin, V. (2017). Formal modelling of distributed automation CPS with CP-agnostic software. Studies in Computational Intelligence, 694, 35–46. https://doi.org/10.1007/978-3-319-51100-9_4

S[133] Butler, M., Dghaym, D., Fischer, T., Hoang, T. S., Reichl, K., Snook, C., & Tummeltshammer, P. (2017). Formal Modelling Techniques for Efficient Development of Railway Control Products. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 10598 LNCS, 71–86. https://doi.org/10.1007/978-3-319-68499-4_5

S[134] Skruch, P., Długosz, M., & Markiewicz, P. (2017). A formal approach for the verification of control systems in autonomous driving applications. Advances in Intelligent Systems and Computing, 577, 178–189. https://doi.org/10.1007/978-3-319-60699-6_18

S[135] Emzivat, Y., Ibanez-Guzman, J., Illy, H., Martinet, P., & Roux, O. H. (2018). A Formal Approach for the Design of a Dependable Perception System for Autonomous Vehicles. IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC, 2018-Novem, 2452–2459. https://doi.org/10.1109/ITSC.2018.8569903

S[136] Muñoz, C., Narkawicz, A., & Dutle, A. (2018). From formal requirements to highly assured software for unmanned aircraft systems. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 10951 LNCS, 647–652. https://doi.org/10.1007/978-3-319-95582-7_38

S[137] Rizaldi, A., Immler, F., Schürmann, B., & Althoff, M. (2018). A Formally Verified Motion Planner for Autonomous Vehicles. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 11138 LNCS, 75–90. https://doi.org/10.1007/978-3-030-01090-4_5

S[138] Li, W., Miyazawa, A., Ribeiro, P., Cavalcanti, A., Woodcock, J., & Timmis, J. (2018). From Formalised State Machines to Implementations of Robotic Controllers. 517–529. https://doi.org/10.1007/978-3-319-73008-0_36

S[139] González, C. A., Varmazyar, M., Nejati, S., Briand, L. C., & Isasi, Y. (2018). Enabling model testing of cyber-physical systems. Proceedings - 21st ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS 2018, 176–186. https://doi.org/10.1145/3239372.3239409

S[140] Besnard, V., Brun, M., Jouault, F., Teodorov, C., & Dhaussy, P. (2018). Unified LTL verification and embedded execution of UML models. Proceedings - 21st ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS 2018, 112–122. https://doi.org/10.1145/3239372.3239395

S[141] Todorov, V., Boulanger, F., & Taha, S. (2018). Formal verification of automotive embedded software. Proceedings - 2018 ACM/IEEE Conference on Formal Methods in Software Engineering, FormaliSE 2018, 84–87.

S[142] Ovsiannikova, P., Chivilikhin, D., Ulyantsev, V., Stankevich, A., Zakirzyanov, I., Vyatkin, V., & Shalyto, A. (2018). Active Learning of Formal Plant Models for Cyber-Physical Systems. Proceedings - IEEE 16th International Conference on Industrial Informatics, INDIN 2018, 719–724.

S[143] Bouskela, D., & Jardin, A. (2018). ETL: A new temporal language for the verification of cyber-physical systems. 12th Annual IEEE International Systems Conference, SysCon 2018 - Proceedings, 1–8. https://doi.org/10.1109/SYSCON.2018.8369502

S[144] Kahani, N. (2018). AutoModel: A domain-specific language for automatic modeling of real-Time embedded systems. Proceedings - International Conference on

Software Engineering, 515–517. https://doi.org/10.1145/3183440.3190333

S[145] Schlingloff, B. H. (2018). Specification and verification of collaborative transport robots. Proceedings - 2018 4th International Workshop on Emerging Ideas and Trends in the Engineering of Cyber-Physical Systems, EITEC 2018, 3–8. https://doi.org/10.1109/EITEC.2018.00006

S[146] Ali, S. (2018). Formal verification of SysML diagram using case studies of real-time system. Innovations in Systems and Software Engineering, 14(4), 245–262. https://doi.org/10.1007/s11334-018-0318-5

S[147] Akram, W., & Niazi, M. A. (2018). A formal specification framework for smart grid components. Complex Adaptive Systems Modeling, 6(1).

S[148] Jarrar, A., & Balouki, Y. (2018). Formal modeling of a complex adaptive air traffic control system. Complex Adaptive Systems Modeling, 6(1).

S[149] Ge, N., Jenn, E., Breton, N., & Fonteneau, Y. (2018). Integrated formal verification of safety-critical software. International Journal on Software Tools for Technology Transfer, 20(4), 423–440. https://doi.org/10.1007/s10009-017-0475-0

S[150] Hailesellasie, M., & Hasan, S. R. (2018). Intrusion Detection in PLC-Based Industrial Control Systems Using Formal Verification Approach in Conjunction with Graphs. Journal of Hardware and Systems Security, 2(1), 1–14.

S[151] Wang, N., & Chen, X. (2018). A formal model for temporal - Spatial event in internet of vehicles. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 11280 LNCS, 222–234. https://doi.org/10.1007/978-3-030-04648-4_19

S[152] Santos, T., Carvalho, G., & Sampaio, A. (2018). Formal modelling of environment restrictions from natural-language requirements. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 11254 LNCS, 252–270. **S[153]** Bouquerel, M., Kremers, E., van der Kamp, J., Thuy, N., & Jardin, A. (2019). Requirements modelling to help

decision makers to efficiently renovate energy systems of urban districts. Simulation Series, 2019-July.

S[154] Ferrère, T., Nickovic, D., Donzé, A., Ito, H., & Kapinski, J. (2019). Interface-aware signal temporal logic. HSCC 2019 - Proceedings of the 2019 22nd ACM International Conference on Hybrid Systems: Computation and Control, 57–66. https://doi.org/10.1145/3302504.3311800

S[155] Cicirelli, F., Nigro, L., & Pupo, F. (2019). Formal Modelling and Verification of Real-Time Self-Adaptive Systems. Proceedings - 2019 IEEE/ACM 23rd International Symposium on Distributed Simulation and Real Time Applications, DS-RT 2019. https://doi.org/10.1109/DS-RT47707.2019.8958707

S[156] Chen, H., Xie, K., Cui, L., & Pescape, A. (2019). A Formal Methodology for Easing Development and Maintenance of Entity Services in Service Oriented Software-Defined Internet of Things. IEEE Internet of Things Journal, 6(6), 9516–9530. https://doi.org/10.1109/JIOT.2019.2929285

S[157] Jue, W., Song, Y., Wu, X., & Dai, W. (2019). A Semi-Formal Requirement Modeling Pattern for Designing Industrial Cyber-Physical Systems. IECON Proceedings (Industrial Electronics Conference), 2019-Octob, 2883–2888.

S[158] Gomes, R. M., & Baunach, M. (2019). Code Generation from Formal Models for Automatic RTOS Portability. CGO 2019 - Proceedings of the 2019 IEEE/ACM International Symposium on Code Generation and Optimization, 271–272. https://doi.org/10.1109/CGO.2019.8661170

S[159] Ruiz, M. C., Garrido-Hidalgo, C., Gruska, D. P., Olivares, T., Hortelano, D., & Roda-Sanchez, L. (2019). Modeling and evaluation of a power-aware algorithm for IoT Bluetooth low energy devices. Proceedings - 2019 IEEE International Conference on Smart Internet of Things, SmartIoT 2019, 28–35.

S[160] Drozdov, D., Patil, S., Dubinin, V., & Vyatkin, V. (2019). Towards formal ASM semantics of timed control systems for industrial CPS. IEEE International

Conference on Emerging Technologies and Factory Automation, ETFA, 2019-Septe, 1682–1685. https://doi.org/10.1109/ETFA.2019.8869293

S[161] Apvrille, L., & Li, L. W. (2019). Harmonizing Safety, Security and Performance Requirements in Embedded Systems. Proceedings of the 2019 Design, Automation and Test in Europe Conference and Exhibition, DATE 2019, 1631–1636. https://doi.org/10.23919/DATE.2019.8715124

S[162] Anda, A. A., & Amyot, D. (2019). Arithmetic semantics of feature and goal models for adaptive cyber-physical systems. Proceedings of the IEEE International Conference on Requirements Engineering, 2019-Septe, 245–256.

S[163] Quan, J., Chunling, Z., & Siqi, W. (2019). Qualitative analysis for state/event fault trees using formal model checking. Journal of Systems Engineering and Electronics, 30(5), 959–973. https://doi.org/10.21629/JSEE.2019.05.13

S[164] Randriamasy, M., Cabani, A., Chafouk, H., & Fremont, G. (2019). Formally validated of novel tolling service with the ITS-G5. IEEE Access, 7, 41133–41144. https://doi.org/10.1109/ACCESS.2019.2906046

S[165] Krichen, M. (2019). Improving Formal Verification and Testing Techniques for Internet of Things and Smart Cities. Mobile Networks and Applications. https://doi.org/10.1007/s11036-019-01369-6

S[166] Ali, T., Yasin, S., Draz, U., & Ayaz, M. (2019). Towards Formal Modeling of Subnet Based Hotspot Algorithm in Wireless Sensor Networks. Wireless Personal Communications, 107(4), 1573–1606. https://doi.org/10.1007/s11277-019-06346-6

S[167] Souri, A., Rahmani, A. M., Navimipour, N. J., & Rezaei, R. (2019). A symbolic model checking approach in formal verification of distributed systems. Human-Centric Computing and Information Sciences, 9(1). https://doi.org/10.1186/s13673-019-0165-x

S[168] Deshmukh, J. V., & Sankaranarayanan, S. (2019). Formal Techniques for

Verification and Testing of Cyber-Physical Systems. Design Automation of Cyber-Physical Systems, 69–105. https://doi.org/10.1007/978-3-030-13050-3_4

S[169] Kaindl, H., Hoch, R., Rathmair, M., & Luckeneder, C. (2019). Formal Verification of Cyber-physical Feature Coordination with Minimalist Qualitative Models. Communications in Computer and Information Science, 1023, 261–287.

S[170] Keerthi, K., Roy, I., Hazra, A., & Rebeiro, C. (2019). Formal verification for security in IoT devices. Internet of Things, 179–200. https://doi.org/10.1007/978-3-030-02807-7_9

S[171] Rodríguez, A., Kristensen, L. M., & Rutle, A. (2019). Formal Modelling and Incremental Verification of the MQTT IoT Protocol. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 11790 LNCS, 126–145. https://doi.org/10.1007/978-3-662-60651-3_5

S[172] Tueno Fotso, S. J., Laleau, R., Frappier, M., Mammar, A., Thibodeau, F., & Nsangou Mouchili, M. (2019). Assessment of a Formal Requirements Modeling Approach on a Transportation System. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 11852 LNCS, 470–486. https://doi.org/10.1007/978-3-030-32409-4_29

S[173] Huang, L., & Kang, E. Y. (2019). Formal verification of safety & security related timing constraints for a cooperative automotive system. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 11424 LNCS, 210–227. https://doi.org/10.1007/978-3-030-16722-6_12

S[174] Idani, A., Ledru, Y., Ait Wakrime, A., Ben Ayed, R., & Collart-Dutilleul, S. (2019). Incremental Development of a Safety Critical System Combining formal Methods and DSMLs. Application to a Railway System. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes

in Bioinformatics), 11687 LNCS, 93–109. https://doi.org/10.1007/978-3-030-27008-7_6

S[175] Avram, C., Bezerra, K., Radu, D., Machado, J., & Astilean, A. (2019). A formal approach for railroad traffic modelling using timed automata. Lecture Notes in Electrical Engineering, 505, 307–314. https://doi.org/10.1007/978-3-319-91334-6_42

S[176] Zhou, P., Zuo, D., Hou, K., Zhang, Z., & Dong, J. (2020). Improving the Dependability of Self-Adaptive Cyber Physical System with Formal Compositional Contract. IEEE Transactions on Reliability, 69(3), 1130–1146.

S[177] Dorr, T., Sandmann, T., & Becker, J. (2020). A Formal Model for the Automatic Configuration of Access Protection Units in MPSoC-Based Embedded Systems. Proceedings - Euromicro Conference on Digital System Design, DSD 2020, 596–603. https://doi.org/10.1109/DSD51259.2020.00098

S[178] Chouali, S., Boukerche, A., Mostefaoui, A., & Merzoug, M. A. (2020). Formal Verification and Performance Analysis of a New Data Exchange Protocol for Connected Vehicles. IEEE Transactions on Vehicular Technology, 69(12), 15385–15397. https://doi.org/10.1109/TVT.2020.3040817

S[179] Hadad, A. S. A., Ma, C., & Ahmed, A. A. O. (2020). Formal Verification of AADL Models by Event-B. IEEE Access, 8, 72814–72834.

S[180] Huang, L., Liang, Y., Huang, H., Wang, Q., Qian, S., & Zhao, R. (2020). Application of formal method in grid cyber physical systems. 1075–1080.

S[181] Hu, M., Duan, W., Zhang, M., Wei, T., & Chen, M. (2020). Quantitative Timing Analysis for Cyber-Physical Systems Using Uncertainty-Aware Scenario-Based Specifications. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 39(11), 4006–4017. https://doi.org/10.1109/TCAD.2020.3012843

S[182] Wang, X., Yi, G., & Wang, Y. (2020). An Empirical Study of Flight Control System Model Checking Integrated with FMEA. Proceedings - Companion of the 2020 IEEE 20th International Conference on Software Quality, Reliability, and Security,

QRS-C 2020, 478–484.

S[183] Shrestha, S. L. (2020). Automatic Generation of Simulink Models to Find Bugs in a Cyber-Physical System Tool Chain using Deep Learning. Proceedings - 2020 ACM/IEEE 42nd International Conference on Software Engineering: Companion, ICSE-Companion 2020, 110–112. https://doi.org/10.1145/3377812.3382163

S[184] Foster, S., Nemouchi, Y., O'Halloran, C., Stephenson, K., & Tudor, N. (2020). Formal model-based assurance cases in isabelle/SACM an autonomous underwater vehicle case study. Proceedings - 2020 IEEE/ACM 8th International Conference on Formal Methods in Software Engineering, FormaliSE 2020, 11–21. https://doi.org/10.1145/3372020.3391559

S[185] Bechar, R., Tahar Abbes, M., Mezoudj, F., & Bellatreche, L. (2020). On Formal Modeling and Validation of Wireless Sensor Network Protocols. Wireless Personal Communications, 114(4), 2855–2888. https://doi.org/10.1007/s11277-020-07507-8

S[186] Jha, S. B., Babiceanu, R. F., & Seker, R. (2020). Formal modeling of cyber-physical resource scheduling in IIoT cloud environments. Journal of Intelligent Manufacturing, 31(5), 1149–1164. https://doi.org/10.1007/s10845-019-01503-x

S[187] Hansen, D., Leuschel, M., Körner, P., Krings, S., Naulin, T., Nayeri, N., … Skowron, F. (2020). Validation and real-life demonstration of ETCS hybrid level 3 principles using a formal B model. International Journal on Software Tools for Technology Transfer, 22(3), 315–332. https://doi.org/10.1007/s10009-020-00551-6

S[188] Abate, A., Bessa, I., Cordeiro, L., David, C., Kesseli, P., Kroening, D., & Polgreen, E. (2020). Automated formal synthesis of provably safe digital controllers for continuous plants. Acta Informatica, 57(1–2), 223–244. https://doi.org/10.1007/s00236-019-00359-1

S[189] Ahmad, W., Hasan, O., & Tahar, S. (2020). Formal reliability and failure analysis of ethernet based communication networks in a smart grid substation. Formal

Aspects of Computing, 32(1), 71–111. https://doi.org/10.1007/s00165-019-00503-1

S[190] Dupont, G., Aït-Ameur, Y., Pantel, M., & Singh, N. K. (2020). Formally Verified Architecture Patterns of Hybrid Systems Using Proof and Refinement with Event-B. Lecture Notes in Computer Science, 12071 LNCS, 169–185.

S[191] Nyberg, M., Westman, J., & Gurov, D. (2020). Formally Proving Compositionality in Industrial Systems with Informal Specifications. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 12478 LNCS, 348–365. https://doi.org/10.1007/978-3-030-61467-6_22

S[192] Poorhadi, E., Troubitysna, E., & Dan, G. (2020). Formalising the Impact of Security Attacks on IoT Safety. Lecture Notes in Computer Science, 12235 LNCS, 69–81. https://doi.org/10.1007/978-3-030-55583-2_5

S[193] Lahbib, A., Ait Wakrime, A., Laouiti, A., Toumi, K., & Martin, S. (2020). An Event-B Based Approach for Formal Modelling and Verification of Smart Contracts. Advances in Intelligent Systems and Computing, 1151 AISC, 1303–1318. https://doi.org/10.1007/978-3-030-44041-1_111

S[194] Alves, G. V., Dennis, L., & Fisher, M. (2020). Formalisation and implementation of road junction rules on an autonomous vehicle modelled as an agent. Lecture Notes in Computer Science, 12232 LNCS, 217–232. https://doi.org/10.1007/978-3-030-54994-7_16

S[195] Wach, P., & Salado, A. (2020). Model-Based Security Requirements for Cyber-Physical Systems in SysML. Systems Security Symposium, SSS 2020 - Conference Proceedings. https://doi.org/10.1109/SSS47320.2020.9174222

S[196] Rosales, R., & Paulitsch, M. (2021). Composable Finite State Machine-based Modeling for Quality-of-Information-aware Cyber-physical Systems. ACM Transactions on Cyber-Physical Systems, 5(2). https://doi.org/10.1145/3386244

S[197] Dupont, Guillaume and Ait-Ameur, Yamine and Singh, Neeraj Kumar and

Pantel, M. (2021). Event-B Hybridation: A Proof and Refinement-based Framework for Modelling Hybrid Systems. ACM Transactions on Embedded Computing Systems (TECS), 20(4), 1–37.

S[198] Guan, Y., Guo, J., & Li, Q. (2021). Formal Verification of a Hybrid IoT Operating System Model. IEEE Access.

S[199] Menghi, C., Vigano, E., Bianculli, D., & Briand, L. C. (2021). Trace-Checking CPS Properties: Bridging the Cyber-Physical Gap. 847–859.

S[200] Akhtar, S., & Zahoor, E. (2021). Formal Specification and Verification of MQTT Protocol in PlusCal-2. Wireless Personal Communications.

S[201] Fredj, N., Hadj Kacem, Y., & Abid, M. (2021). An event-based approach for formally verifying runtime adaptive real-time systems. Journal of Supercomputing, 77(3), 3110–3143. https://doi.org/10.1007/s11227-020-03386-9