



UniBa

UNIVERSITÀ
DEGLI STUDI
DI BARI
ALDO MORO



UNIVERSITÀ DEGLI STUDI DI BARI ALDO MORO

DEPARTMENT OF COMPUTER SCIENCE

PHD PROGRAMME IN COMPUTER SCIENCE AND MATHEMATICS

XXXVII CYCLE

SCIENTIFIC DISCIPLINARY AREA IINF-05/A

QUANTUM SOFTWARE ENGINEERING FOR SECURITY

Coordinator:

Prof. Francesca MAZZIA

FRANCESCA MAZZIA
27.03.2025 09:23:53
GMT+01:00

Tutor:

Prof. Danilo CAIVANO

Co-Tutor:

Dr. Saverio PASCAZIO

SAVERIO
PASCAZIO
26.03.2025
20:01:27
GMT+02:00



PhD Candidate:

Anibrata PAL

Digitally
signed by
PAL
ANIBRATA
C=IT



Firmato digitalmente da:

CAIVANO DANILO

Firmato il 27/03/2025 06:55

Seriale Certificato: 2034561

Valido dal 27/12/2022 al 27/12/2025

InfoCamere Qualified Electronic Signature CA

FINAL EXAM 2025

FUNDED BY PON RICERCA E INNOVAZIONE 2014-2020 FSE REACT-EU,
AZIONE IV.4 "DOTTORATI E CONTRATTI DI RICERCA SU TEMATICHE
DELL'INNOVAZIONE" (CUP: H99J21010060001)

Dedication

To my father, who would have been proud to see me reach this milestone.

To my mother and my dearest family—my daughter, my greatest strength, and my wife, whose unwavering support and sacrifices made this journey possible.

To my in-laws, for their endless support and belief in me.

Acknowledgements

This thesis would not have been possible without the guidance, support, and encouragement of many individuals, to whom I am deeply grateful.

I extend my sincere gratitude to my advisor, Prof. Danilo Caivano, for his invaluable guidance throughout this journey, and to my co-advisor, Prof. Saverio Pascazio, for his thoughtful feedback and encouragement. A heartfelt thanks to Vita Santa Barletta, whose unwavering mentorship helped me overcome challenges and see possibilities beyond problems.

I am thankful to Mirko De Vincentiis for collaborating on numerous projects, sharing technical insights, and for his enduring friendship. I gratefully acknowledge Vincenzo, Domenico, Sara, Berenice, Sabina, and Imran for their friendship and support throughout this journey.

Special thanks to Prof. Antonio Piccinno, whose support extended far beyond academia—helping me settle in Italy. I am also indebted to Prof. Maria Francesca Costabile for her crucial help throughout visa and the admission process, and to Prof. Maria Teresa Baldassarre, Prof. Michele Scalera, and Prof. Paolo Buono for their guidance throughout my PhD.

I warmly thank Prof. Francesca Mazzia, my PhD coordinator, for her continuous support, and Prof. Manuel Serrano, member of my evaluation committee, for his insightful feedback and subject-matter guidance.

I acknowledge the University of Bari and the Department of Computer Science for providing an excellent academic environment. This research was supported by PON Ricerca e Innovazione 2014–2020 FSE REACT-EU, Azione IV.4 (CUP: H99J21010060001), which I gratefully recognize.

Thanks to SerLab and Ser & P Practices srl for hosting my internship, and to Dr. Giovanni Bruno for his support during that period. I also thank Francesca Pace and Adolorata Brivittello for their invaluable administrative assistance over these three years.

My deepest gratitude goes to my family—my daughter Aadriya and my wife Ahana—for their unconditional love, sacrifices, and constant emotional support.

To all who supported me, in ways big and small—thank you.

Abstract

Quantum Software Engineering (QSE) merges quantum computing with software engineering to tackle problems beyond classical capabilities. As quantum technology continues to develop, there is growing concern about the security and privacy of quantum-hybrid systems, especially due to the challenges involved in merging quantum and classical computing systems and due to their natural weaknesses, which include qubit decoherence and vulnerability to side-channel attacks. This thesis addresses these challenges through a comprehensive evaluation of existing QSE practices, identifying critical gaps in Security by Design (SbD) and Privacy by Design (PbD).

To bridge these gaps, this work pursues three key goals: (i) analyzing existing security and privacy integration models in QSE, (ii) proposing and empirically validating new methodologies for secure quantum software development, and (iii) adopting agile approaches for iterative security and privacy enforcement in QSE.

A systematic literature review reveals a lack of dedicated research on PbD and SbD, underscoring the need for empirical studies in quantum-hybrid programming and quantum-enhanced machine learning. This analysis leads to the QuIPS (Quantum Impact on Privacy and Security) framework, which maps privacy and security integration into the Quantum Development Lifecycle (QDLC), highlighting key phases for embedding security measures.

Building on QuIPS, this thesis introduces and validates multiple quantum-enabled security frameworks across domains like automotive, smart cities, and education. The Quantum Privacy-Oriented Software Development (QPOSD) model embeds security and privacy into Quantum software Development Lifecycle (QDLC), while the Quantum Error (QError) methodology systematically analyzes quantum bugs and vulnerabilities, linking them to security risks. Five case studies empirically assess these frameworks, demonstrating their feasibility and real-world applicability.

Further extending QuIPS, the thesis integrates DevSecOps, MLOps, and Scrum into QSE, ensuring continuous security validation in agile quantum development. This unified framework embeds PbD and SbD principles via proactive risk assessment, automated security testing, and CI/CD pipelines, enhancing quantum software resilience while addressing iterative security needs.

Results confirm that quantum technologies can enhance security and privacy

in diverse domains while maintaining accuracy and achieving exponential time gains. QPOSD facilitates early-stage security integration, while QError expedites vulnerability mitigation in quantum-hybrid systems. However, given the nascent stage of quantum computing, it is premature to generalize these conclusions, necessitating further empirical validation and real-world implementation. It is important to acknowledge that the outcomes of experiments conducted on quantum computing platforms are inherently non-deterministic, influenced by the current state and maturity of quantum technologies. Therefore, ongoing and rigorous analysis of the proposed frameworks and models remains essential to refine and adapt them for future advancements in this rapidly evolving field.

Sommario

Quantum Software Engineering (QSE) unisce l'informatica quantistica con l'ingegneria del software per affrontare problemi che vanno oltre le capacità classiche. Mentre la tecnologia quantistica continua a svilupparsi, cresce la preoccupazione per la sicurezza e la privacy dei sistemi ibridi quantistici, soprattutto a causa delle sfide legate alla fusione dei sistemi informatici quantistici e classici e per le loro naturali debolezze, che includono la decoerenza dei qubit e la vulnerabilità agli attacchi del canale laterale(side-channel). Questa tesi affronta queste sfide attraverso una valutazione completa delle pratiche QSE esistenti, identificando lacune critiche in Security by Design (SbD) e Privacy by Design (PbD).

Per colmare queste lacune, questo lavoro persegue tre obiettivi chiave: (i) analizzare i modelli di integrazione di sicurezza e privacy esistenti in QSE, (ii) proporre e convalidare empiricamente nuove metodologie per lo sviluppo sicuro di software quantistico e (iii) adottare approcci agili per l'applicazione iterativa di sicurezza e privacy in QSE.

Una revisione sistematica della letteratura rivela una mancanza di ricerca dedicata su PbD e SbD, sottolineando la necessità di studi empirici sulla programmazione ibrida quantistica e sull'apprendimento automatico potenziato dai quanti. Questa analisi porta al framework QuIPS (Quantum Impact on Privacy and Security), che mappa l'integrazione di privacy e sicurezza nel Quantum Development Lifecycle (QDLC), evidenziando le fasi chiave per l'incorporamento di misure di sicurezza.

Basandosi su QuIPS, questa tesi introduce e convalida più framework di sicurezza abilitati dai quanti in domini come automotive, smart city e istruzione. Il modello Quantum Privacy-Oriented Software Development (QPOSD) incorpora sicurezza e privacy nel Quantum software Development Lifecycle (QDLC), mentre la metodologia Quantum Error (QError) analizza sistematicamente bug e vulnerabilità quantistiche, collegandoli ai rischi per la sicurezza. Cinque casi di studio valutano empiricamente questi framework, dimostrandone la fattibilità e l'applicabilità nel mondo reale.

Estendendo ulteriormente QuIPS, la tesi integra DevSecOps, MLOps e Scrum in QSE, garantendo una convalida continua della sicurezza nello sviluppo agile quantistico. Questo framework unificato incorpora i principi PbD e SbD tramite valutazione proattiva del rischio, test di sicurezza automatizzati e

pipeline CI/CD, migliorando la resilienza del software quantistico e rispondendo alle esigenze di sicurezza iterative.

I risultati confermano che le tecnologie quantistiche possono migliorare la sicurezza e la privacy in diversi domini, mantenendo l'accuratezza e ottenendo guadagni di tempo esponenziali. QPOSD facilita l'integrazione della sicurezza in fase iniziale, mentre QError accelera la mitigazione della vulnerabilità nei sistemi ibridi quantistici. Tuttavia, data la fase nascente del calcolo quantistico, è prematuro generalizzare queste conclusioni, il che richiede un'ulteriore convalida empirica e un'implementazione nel mondo reale. È importante riconoscere che i risultati degli esperimenti condotti su piattaforme di calcolo quantistico sono intrinsecamente non deterministici, influenzati dallo stato attuale e dalla maturità delle tecnologie quantistiche. Pertanto, resta essenziale un'analisi continua e rigorosa dei quadri e dei modelli proposti per perfezionarli e adattarli ai futuri progressi in questo campo in rapida evoluzione.

Contents

Dedication	ii
Acknowledgements	iii
Abstract	iv
Sommario	vi
List of Figures	x
List of Tables	xiii
1 Introduction	1
1.1 Research Motivation	1
1.2 Objectives	2
1.3 Thesis Outline	4
2 Background and Related Works	5
2.1 Software Engineering: History and Methodologies	5
2.2 Quantum Computing: Quick Primer	8
2.3 Security and Privacy in QSE	14
2.3.1 Quantum Development Lifecycle (QDLC)	15
2.3.2 Talavera Manifesto	16
2.3.3 Integrating Security and Privacy	18
2.4 Empirical Validation in QSE	19
2.5 Agile in QSE	25
3 Frameworks and Models	28
3.1 Security and Privacy in QSE	29
3.1.1 Research Methodology	30
3.1.2 Data Synthesis and Results	39
3.1.3 Answers to the Research Questions	39
3.1.4 Discussion	54
3.2 Empirical Validation in QSE	72
3.2.1 QuIPS Framework	72
3.2.2 Automotive Sector	76
3.2.3 Smart City Sector	84

3.2.4	Education Sector	91
3.2.5	Security and Privacy-Aware Quantum Software Development	93
3.2.5.1	Quantum Privacy Knowledge Base	94
3.2.5.2	Quantum Privacy-Oriented Software Development	101
3.2.6	Quantum Bug and Vulnerability Analysis in Quantum Software Development	105
3.3	Agile in QSE	110
3.3.1	DevSecOps and MLOps enabled Quantum Scrum	111
3.3.2	Scrum for Iterative Quantum Development	113
3.3.3	Unified Framework for Agile Privacy and Security in QSE	116
4	Experiments and Results	120
4.1	Automotive Sector	121
4.1.1	Quantum-based Intrusion Detection System	121
4.1.2	Quantum Powered Developer Support for Automotive Security	124
4.1.3	Quantum AI in Extending Developer Support	127
4.2	Smart City Sector	131
4.2.1	Datasets	132
4.2.2	Data Pre-processing & Experiments	134
4.2.3	Results	143
4.3	Security and Privacy-Aware Quantum Software Development	148
4.3.1	Results and Discussion	148
4.3.2	Limitations/Threats to Validity	151
4.4	Quantum Bug and Vulnerability Analysis in Quantum Software Development	152
4.4.1	Results and Discussion	152
4.4.2	Case Studies	162
5	Conclusion, Limitation, and Future Work	172
5.1	Conclusion	172
5.2	Limitation	174
5.3	Future Work	175
	Bibliography	177
	Appendices	205
A	Quantum and Quantum-Resilient Protocols	206
A.1	Algorithms and Security Aspects	206
A.1.1	Shor's Algorithm	206
A.1.2	Grover's Algorithm	207
A.1.3	Quantum Annealing	208
A.1.4	Quantum Key Distribution (QKD)	208
A.2	Quantum-Resistant Cryptography	211

List of Figures

2.1	Bloch Sphere.	9
2.2	Quantum Circuit for Full Adder	11
2.3	Energy change diagram during Quantum Annealing.	12
2.4	Energy diagram - Best answer.	13
2.5	Quantum Development Lifecycle	15
3.1	Systematic Literature Review flowchart.	37
3.2	Distribution of papers across Years, Venues, and Type of Work (RQ1 and RQ2)	40
3.3	Objectives and Methodologies used (RQ3).	41
3.4	Technologies used (RQ3).	41
3.5	Graphs related to Research Questions RQ4 and RQ5	42
3.6	Relevance to Principles of Privacy by Design in QSE	45
3.7	Adherence to Principles of Privacy by Design in QSE	45
3.8	Distribution of Papers in Privacy by Design	46
3.9	Security by Design in QSE	47
3.10	Adherence to Principles of Security by Design in QSE	47
3.11	Distribution of Papers in Security by Design	48
3.12	QuIPS Framework	75
3.13	IDS pipeline	78
3.14	Proposed Dev-Support methodology.	79
3.15	Proposed example to map the DoS attack using the taxonomy to support the developer.	81
3.16	Proposed methodology to support the developer in the automo- tive field: SeQuADE (Secure Quantum Automotive Development and Engineering).	84
3.17	Hybrid Quantum-Classical Architecture for Smart City Security.	86
3.18	Integrated Smart City Security Dashboard for analysis and in- vestigation.	87
3.19	IDS Pipeline.	88
3.20	ICSEQAI Vision Model to support Developers and Testers during SDLC	92
3.21	Quantum Privacy Knowledge Base Relationship	94
3.22	Recurring bug patterns in Quantum Hybrid Software Develop- ment [1]	96
3.23	Quantum Hybrid Recurring Vulnerabilities/Security Patterns [2]	98

3.24	Quantum Privacy-Oriented Software Development Model	103
3.25	Quantum Privacy Design Strategies	104
3.26	Methodology of Code Commit extraction and Analysis	107
3.27	Scrum Methodology(<i>From scrum.org website</i>)	111
3.28	DevSecOps Framework (From https://www.pagerduty.com/blog/devsecops-ops-guide/)	112
3.29	MLOps Framework (From https://ubuntu.com/blog/what-is-mlops)	113
3.30	Unified Framework	117
4.1	SVC and QBoost Confusion matrix on DoS & Fuzzy datasets. "0" malicious; "1" normal messages. TN-True Negative, FP-False Positive, FN-False Negative, TP-True Positive	124
4.2	Quality of the ML models.	127
4.3	Time Efficiency of ML Classifiers.	128
4.4	The top three sentences most similar to the description derived from the taxonomy. For brevity, CWE and their descriptions have been excluded from the final sentence. Highlighted terms, with yellow and green underlines, indicate similar words.	129
4.5	Sensors in MQTTset Scenario [3].	133
4.6	The figures depict the rules for identifying an HTTP Flood attack categorized as DDoS. Figure (a) displays the main rule, and if any of the conditions outlined in (a) are met, an offense will be triggered. Figure (b) presents the rules for two or more IP Sources. Figures (c) and (d) show the BB rules.	139
4.7	Identification of DDoS attack offense.	140
4.8	XSS attack identification rule.	141
4.9	XSS attack and Offense detection information.	141
4.10	SQL Injection attack identification rule.	141
4.11	Offense generated by SQL Attack.	142
4.12	SQL Injection with destination IPs involved in the attack. . . .	142
4.13	The images depict the rules used to detect a port scan. Figure (a) displays the primary rule, which triggers an offense if any of the rules outlined in (a) are met. Figure (c) illustrates a BB rule.	142
4.14	Figure (a) shows the offense generated when a Port Scan attack is detected. Instead, Figure (b) shows the offense generated but with more details.	143
4.15	Quality metrics based comparison of Classifiers	144
4.16	Quality metrics using PCA.	145
4.17	Quality metrics of Classifiers.	147
4.18	Data transmission security over quantum/classical channel . . .	150
4.19	Data collected from Github	153
4.20	Code Commits before Filtering	154
4.21	Bug Patterns	155
4.23	Mapping of Bugs Patterns with QDLC Phases	155
4.22	Mapping of Bugs Patterns with Vulnerability Patterns	156

4.24 Mapping of QDLC phases with Vulnerability Patterns	158
4.25 Mapping of QDLC phases with PbD and SbD Principles	159
4.26 Heatmap of PbD in Bug Patterns	160
4.27 Heatmap of SbD in Bug Patterns	161
4.28 Logic check for unsupported orbital configurations	163
4.29 Dependency support fix for custom gate sets	165
4.30 Circuit handling fix for better measurement and reuse	168
4.31 Circuit optimization fix with consistent functional transformations	170
4.32 Fix logic for improved circuit decomposition control	171

List of Tables

2.1	Privacy by Design Principles	19
2.2	Security by Design Principles	20
3.1	Research Questions and their Motivations.	31
3.2	Major, alternative, and related terms	32
3.3	Inclusion Criteria	33
3.5	Specific items for Data Extraction	33
3.5	Specific items for Data Extraction	35
3.4	Exclusion Criteria	36
3.6	Quality Assessment Questions	39
3.7	Relevance Scoring for Privacy and Security	44
3.8	Quality Assessment Scores	50
3.9	Quality Assessment Results	51
3.10	Compliance to Privacy by Design Principles	65
3.11	Security by Design Principles and Compliance	67
3.12	Comparison of Current and Proposed States in QSE	71
3.13	Impact of Principles of Privacy by Design Table 3.10 on QSE . .	73
3.14	Impact of Principles of Security by Design Table 3.11 on QSE .	74
3.15	Quantum Privacy Pattern Template	102
4.1	Car-Hacking Dataset [4, 5] overview for Intrusion Detection. . .	122
4.2	Average time performance of the Binary classifiers on DoS dataset. Grey shows the best performance	123
4.3	Average time performance of the Binary classifiers on Fuzzy dataset. Gray shows best performance	123
4.4	Performance of Binary Classifiers	125
4.5	Results obtained with QBoost in a multi-class strategy.	131
4.6	IoT Weather Data Attributes	132
4.7	Time Performance based comparison of Classifiers	143
4.8	Time Performance (PCA $n=2,4,6,8$).	146
4.9	Time efficiency: QBoost and Random Forest.	147

Chapter 1

Introduction

Quantum computing represents a significant technological shift with the potential to transform various industries. As we start in an era of so-called quantum supremacy, the emphasis is on the practical integration of quantum technologies, particularly regarding security and privacy. The rapid advancement of these technologies presents an opportunity to rethink the design and protection of future computing systems. However, it also introduces substantial challenges that must be addressed to safeguard critical infrastructure and data processing. This thesis focuses on studying frameworks that incorporate Security by Design and Privacy by Design within quantum software engineering. It aims to ensure that as quantum computing becomes more prevalent, the systems developed are innovative and secure.

1.1 Research Motivation

The advent of Quantum Computing (QC), in conjunction with an increase in focused research initiatives, signifies what numerous scholars regard as a Golden Era for this groundbreaking technology. As we find ourselves at the threshold of this novel frontier, it is imperative to acknowledge that, notwithstanding the substantial progress achieved in quantum research, a considerable portion of contemporary efforts remains in an embryonic phase. This ongoing investigation frequently lacks empirical metrics for practical execution within industrial frameworks and commercial endeavors.

Pragmatically, we may undergo a considerable duration before QC can be fully assimilated into routine business practices. Nonetheless, the potential it embodies is unequivocally transformative. The 21st century is anticipated to be characterized by quantum breakthroughs, with projections that pioneering software applications will arise across a diverse array of sectors.

QC is expected to significantly enhance capabilities in various domains, including privacy and security, through advanced data safeguarding. In supply chain and logistics, it promises improved routing and inventory management for

better operational efficiency. The fields of chemistry and pharmaceuticals could benefit from faster drug discovery and material science advancements through quantum simulations. In bioinformatics, QC may revolutionize the analysis of complex biological data, aiding in genetic disease research and personalized medicine. Additionally, quantum algorithms could improve risk assessments and investment strategies in economics and finance, optimize resource management in energy solutions, and innovate agricultural crop modeling for food security. Moreover, the defense sector may also leverage quantum technologies to strengthen cybersecurity and intelligence evaluations.

The rapid rise of quantum technology presents both benefits, such as enhanced computing capabilities, and challenges, particularly in security. Quantum computing threatens existing cryptographic methods essential for online security, risking the integrity of systems that protect against unauthorized access. This research aims to address the gap between advancements in quantum technology and the need for secure quantum software development by applying Security by Design (SbD) and Privacy by Design (PbD) principles. The focus is on integrating security measures throughout the software development lifecycle, ensuring that new quantum systems and hybrid models are built with security and privacy in mind from the outset. This proactive approach aims to prepare for a quantum-secure digital environment, enhancing defenses against vulnerabilities and mitigating risks associated with quantum technology implementation.

1.2 Objectives

The thesis essays a comprehensive investigation and analysis of the contemporary advancements regarding Privacy by Design and Security by Design principles within the realm of quantum software engineering. It scrutinizes prevailing propositions aimed at the integration of security and privacy within quantum software engineering, with the objective of discovering optimal practices conducive to effective models and prospective guidelines for the research domain. This thesis additionally outlines an array of methodologies and frameworks designed to ameliorate threats through the utilization of quantum technologies in the context of hybrid systems. These include methodologies that transform classical systems into hybrid systems through the incorporation of quantum elements, thereby enhancing security and privacy across sectors such as automotive, smart cities, and IoT. Furthermore, it underscores the efficacy of integrating quantum technologies to formulate innovative methodologies that bolster cybersecurity education across both industrial and academic spheres. In conjunction with the retrofitting of quantum technologies to enhance the current quantum software engineering methodologies, this dissertation advocates for a hybrid approach within the smart cities context, harnessing both quantum and classical technologies to safeguard infrastructures and assets.

In addition, it introduces a methodology for the direct integration of privacy and security principles into the quantum development lifecycle, thereby ensuring a proactive, and not reactive stance towards the development of secure and

private quantum software. Quite understandably, a substantial proportion of the challenges associated with security and privacy can be attributed to the presence of bugs and vulnerabilities. This thesis performs an exhaustive analysis of quantum bugs and vulnerabilities, proposing an initial skeleton for the analysis and mapping of bugs/vulnerabilities to privacy and security principles, thus offering valuable insights into the patterns of bugs and the management of vulnerabilities.

Finally, the thesis discusses Agile methodologies in the perspective of QSE and proposes an integrated framework following in the Agile mindset combining Scrum, DevSecOps, and MLOps methodologies to enhance privacy and security within Quantum Software Engineering (QSE). This approach ensures iterative quantum development, continuous security integration, and effective ML lifecycle management, thereby building robust, privacy-compliant quantum software that aligns with security-by-design principles throughout the development lifecycle. The aforementioned contributions were based on the following objectives:

1. **Goal 1:** To analyze and evaluate existing models and frameworks for integrating Security by Design and Privacy by Design in quantum software engineering, in short *Security and Privacy in QSE*. This goal can be subdivided further into below mentioned research questions (RQ):
 - **RQ1.1:** What existing frameworks exist in the literature for integrating Security by Design and Privacy by Design into classical software engineering, and how can they be adapted to quantum software?
 - **RQ1.2:** What are the current quantum-specific models at implementing SbD and PbD principles and how effective are they?
 - **RQ1.3** What are the primary challenges in adapting these models to quantum software engineering?
2. **Goal 2:** To analyze and propose novel methodologies for incorporating security and privacy into quantum software engineering, emphasizing empirical validation and verification of the proposed methodologies to ensure their effectiveness and robustness. This goal is referred as *Empirical Validation in QSE* in the rest of the thesis.
 - **RQ2.1:** What novel methodologies can be proposed to incorporate security and privacy into quantum software engineering?
 - **RQ2.2:** How can these methodologies be validated and verified to ensure robustness and effectiveness?
 - **RQ2.3:** What metrics and evaluation criteria should be employed to assess the quality of these proposed methodologies?
3. **Goal 3:** To adopt an agile approach to implement privacy and security imperatives within quantum software development, or in other words

Agile in QSE.

- **RQ3.1** How can agile frameworks, such as Scrum, DevSecOps, and MLOps, be effectively adapted for quantum software development?
- **RQ3.2** What specific challenges do agile methodologies face when implemented in the context of quantum software, and how can they be mitigated?
- **RQ3.3** How can these agile practices be used to improve security and privacy during quantum software development continuously?

1.3 Thesis Outline

The rest of the thesis is organized as:

- **Chapter 2:** Discusses the relevant background and related works in relation to classical Software Engineering, and gradual transition to Quantum Software Engineering including integration of Privacy and Security by Design Principles.
- **Chapter 3:** Presents the frameworks and models as per the goals established in section 1.2. Describes the secure hybrid quantum models and frameworks implemented for diverse real-world sectors, and Secure Quantum Software Development, Quantum Bugs and Vulnerability analysis, and integrating privacy and security in agile QDLC.
- **Chapter 4:** Presents the experiments, results and discussions based on the frameworks and models presented in Chapter 3. This chapter also includes relevant case studies.
- **Chapter 5:** Presents the conclusion and the future works.

Chapter 2

Background and Related Works

Quantum computing is poised to revolutionize computation, yet its adoption introduces significant security and privacy challenges. Traditional security frameworks struggle to accommodate quantum-specific risks, necessitating new approaches. This thesis explores Security by Design and Privacy by Design within Quantum Software Engineering (QSE), analyzing existing models, empirical validation techniques, and the role of Agile methodologies in integrating security seamlessly. By evaluating current frameworks and adapting validation strategies, we aim to propose innovative techniques that ensure quantum software remains secure, resilient, and adaptable. This research advances QSE by fostering iterative, security-conscious development aligned with evolving quantum technologies and threat landscapes.

2.1 Software Engineering: History and Methodologies

The inception of computer systems can be traced back to the early 1930s, initiating the evolution of hardware that constituted the foundation for contemporary computing. Throughout the decades, this initial hardware underwent significant transformations, culminating in the emergence of progressive programming languages. The introduction of FORTRAN in the 1950s, primarily developed for scientific and engineering applications, represented a notable advancement in facilitating programmers' ability to devise intricate algorithms with greater ease. This was subsequently followed by COBOL in the 1960s, which was specifically engineered for the management of business data and prioritized clarity and user-friendliness, thereby rendering programming more attainable for a broader demographic.

With time, it became increasingly evident that standardized methodologies were essential for addressing the complexities inherent in software development. In 1968, Edsger W. Dijkstra proposed the concept of structured programming [6], which advocated for a methodical approach to coding that emphasized

transparency and simplicity. This represented a key moment in the domain of software engineering, as it established a framework for reducing errors and improving code maintainability. Concurrently, methodologies introduced by pioneers such as Warnier and Jackson [7, 8, 9, 10] further propelled the formalization of software development practices, offering structured approaches for the analysis and design of software systems.

Today, we find ourselves in a highly sophisticated Software Engineering paradigm that encompasses a wide array of methodologies and best practices. This evolution has transformed the software development landscape, making it not only more economical but also focused on maximizing code reuse and accelerating time-to-market. The integration of agile methodologies [11, 12, 13], DevOps (Development Operations) practices [14], and continuous integration/continuous deployment (CI/CD) pipelines [15, 16, 17, 18] exemplifies the current emphasis on flexibility and rapid iteration. Consequently, software engineering has become a critical discipline that underpins virtually every industry, driving innovation and efficiency in an increasingly digital world. Moreover, the inclusion of Artificial Intelligence and Machine Learning in Software Engineering has given rise to newer practices like Machine Learning Operations (MLOps)¹ to streamline the deployment and monitoring of machine learning models, enhancing their reliability and security in real-world applications. We present some of the common software engineering methodologies and frameworks below at a glance, some of which are not in practice currently, while others are being actively used.

The Waterfall Model is one of the earliest software development methodologies. The model is a linear and sequential approach with each development phase being completed before moving on to the next. The phases typically include Requirement Analysis, System Design, Implementation, Integration and Testing, Deployment, and Maintenance. This model best suits projects with well-defined requirements and minimal expected changes. Its simplicity and structured approach make it easy to manage; however, it lacks flexibility and adaptability to changes during development [19].

The Incremental Model builds software incrementally, delivering a series of partial systems that gradually evolve into the final product. Each increment delivers a functional subset of the system, allowing developers to prioritize features and users to receive value early. This model combines the structure of the Waterfall Model with the flexibility of iterative approaches, making it suitable for projects with clear modular requirements or evolving needs [20].

The Spiral Model combines elements of both the Iterative and Waterfall models. It emphasizes risk assessment and iterative refinement through its four phases: Planning, Risk Analysis, Engineering, and Evaluation. Each cycle of the spiral focuses on a specific set of objectives, progressively building towards the final product. The model's strength lies in its ability to handle complex and high-risk projects by identifying and mitigating risks early [21].

¹<https://ml-ops.org/>

The V-Model, or Verification and Validation Model, extends the Waterfall Model by emphasizing testing at every development phase. For each development phase (e.g., requirements, design, implementation), there is a corresponding testing phase (e.g., unit testing, integration testing, system testing). The V-Model enhances quality assurance and ensures that all deliverables meet predefined requirements, making it ideal for safety-critical systems [22].

The Rapid Application Development (RAD) Model emphasizes speed and flexibility by focusing on rapid prototyping and iterative delivery. The process involves minimal planning, with the primary focus on delivering functional prototypes in a short time frame. RAD is highly suitable for projects with tight deadlines and well-understood requirements, allowing frequent user feedback to guide development [23].

The Iterative Model emphasizes repeated cycles of development, where a project is broken down into smaller parts or iterations. Each iteration involves planning, design, implementation, and testing, resulting in a functional intermediate product. Feedback from stakeholders at each iteration allows for continuous improvement and refinement. Unlike the Waterfall Model, the Iterative Model accommodates evolving requirements and reduces risks by tackling high-risk areas early in the development process [24].

Agile is a flexible and iterative approach to software development. It emphasizes collaboration, customer feedback, and incremental delivery of functional components. Agile promotes adaptive planning and continuous improvement, making it well-suited for dynamic and complex projects. Key principles of Agile include frequent delivery of working software, welcoming changing requirements, close collaboration between developers and stakeholders, and fostering self-organizing teams. Frameworks like Scrum are commonly used to implement Agile practices [25]. Scrum is an Agile framework that structures work into time-boxed iterations called "sprints," typically lasting 2-4 weeks. Scrum introduces defined roles, including the Product Owner, Scrum Master, and Development Team, and key artifacts such as the Product Backlog, Sprint Backlog, and Increment. Regular events, including Sprint Planning, Daily Scrum, Sprint Review, and Sprint Retrospective, provide structure and transparency. Scrum emphasizes continuous progress and alignment with customer needs through transparency, inspection, and adaptation [26].

DevOps is a cultural and technical movement aimed at unifying software development and IT operations to improve collaboration, automation, and deployment speed. Core principles of DevOps include Continuous Integration and Continuous Deployment (CI/CD), automation of testing and infrastructure provisioning, and real-time monitoring and feedback loops. By bridging the gap between development and operations, DevOps enhances productivity and reduces time-to-market, enabling faster and more reliable software delivery [27].

DevSecOps extends the principles of DevOps by embedding security into every phase of the software development lifecycle. It emphasizes proactive threat identification by incorporating security requirements early in development,

automated security testing through tools for vulnerability scanning and compliance, and collaboration between developers, security teams, and operations. By treating security as a continuous and integral part of the process, DevSecOps ensures that applications are secure by design, reducing vulnerabilities and risks [28].

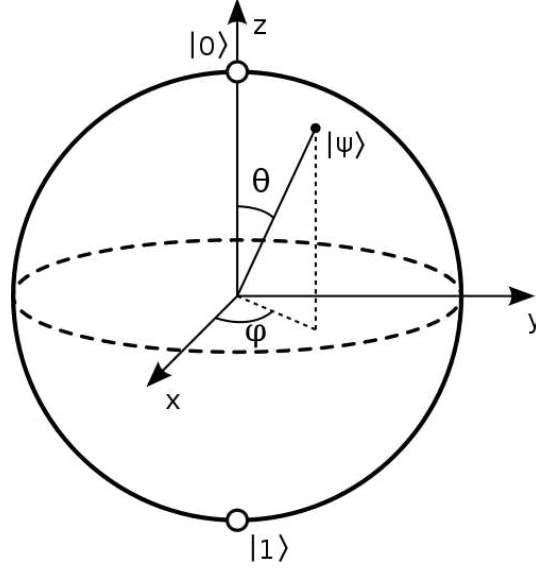
MLOps, or Machine Learning Operations, focuses on the deployment and maintenance of machine learning models. It adapts DevOps practices to the specific challenges of machine learning workflows, including data management to ensure high-quality and consistent training data, model training and versioning for tracking experiments, continuous deployment of models into production environments, and monitoring to address drift or bias in model performance. MLOps fosters collaboration between data scientists, engineers, and operations teams, enabling reliable and scalable machine learning applications [29].

2.2 Quantum Computing: Quick Primer

Quantum mechanics has undergone significant evolution since the commencement of the 20th century, with the foundational ideas of quantum computing emerging during the early 1980s [30, 31, 32]. Prominent quantum algorithms were devised in the latter decades of the century, among which are Deutsch’s algorithm established in 1985 [33], Shor’s algorithm introduced in 1994 [34], and Grover’s algorithm developed in 1996 [35]. At present, quantum computers signify a concrete technological progression, with specific computational tasks potentially being performed at an exponential efficiency on a quantum processor in contrast to a classical processor. In recent years, access to quantum computing resources has been significantly broadened through the advent of cloud computing infrastructures [9]. Currently, a diverse range of quantum computers is operational, each employing distinct layers within the technological framework (Fig. 1) [10], and there exists a considerable variety of quantum programming languages, frameworks, and libraries [11]. Furthermore, a plethora of algorithms (<https://quantumalgorithmzoo.org/>) has been proposed alongside the emergence of multiple quantum programming languages. Consequently, it is essential to advocate for and substantiate novel software engineering methodologies to facilitate the onset of a new illustrious epoch for quantum software engineering.

Quantum computing’s promise is not without challenges. Unlike traditional bits, quantum bits or qubits are inherently fragile and susceptible to decoherence. In classical computing, a bit can either be in a state of 0 or 1 at any given time. However, qubits can exist in a linear combination of both 0 and 1 states simultaneously. Superposition is one of the fundamental principles of quantum mechanics and is a key feature that makes this possible. Mathematically, this superposition is represented as:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$



Source: Smite-Meister, CC BY-SA 3.0

<<https://creativecommons.org/licenses/by-sa/3.0/>>, via Wikimedia Commons

Figure 2.1: Bloch Sphere.

Where, $|\psi\rangle$ represents the state of the qubit, $\alpha|0\rangle$ and $\beta|1\rangle$ represent the classical states 0 and 1, respectively, and α and β are complex probability amplitudes, which determine the probability of measuring the qubit in the $|0\rangle$ or $|1\rangle$ state when a measurement is made. A qubit is often represented by a Bloch Sphere as shown in Figure 2.1.

Apart from superposition, quantum computing also exhibits properties like entanglement, quantum tunneling/annealing, and quantum interference, enabling it to process information differently from classical computers [36]. It offers the potential for exponential speedup in solving specific problems, such as factoring large numbers and optimizing complex systems, making it a promising technology for various fields.

Quantum computers can be based on different physical qubit realizations like superconducting materials [37], trapped ion [38], and electron spin [39]. The qubits can be of different types like atom-based (trapped ion, cold atom), electron-based (superconducting loops and controlled spin, and quantum annealing), or photon-based. Quantum computing encompasses various approaches to harness quantum phenomena for performing computational tasks like Gate-based Quantum Computing, Topological Quantum Computing, and Adiabatic Quantum Computing.

1. *Gate-based quantum computing:* It employs quantum gates to manipulate qubits, which are the fundamental units of quantum information. These gates perform operations on qubits, enabling the creation of quantum circuits that implement algorithms. The operations include single-qubit gates (acting on individual qubits) and multi-qubit gates (acting on pairs

of qubits). The quantum gates like Hadamard gate (H), Pauli gates (X, Y, Z), CNOT (Controlled-NOT) gate, and phase gate (S) can be composed in sequences to construct Quantum circuits. The quantum gates are applied sequentially to the qubits, with each gate affecting the quantum state of the qubits (for example, Figure 2.2).

2. *Topological quantum computing*: Conceived by the physicist Alexei Kitaev in 1997, it employs anyons—quasiparticles in 2D systems—as logic gates, forming braids in 3D spacetime. Unlike traditional quantum computers relying on trapped particles prone to decoherence, topological quantum computers claim stability due to their robust topological properties. These properties, safeguarding against minor disruptions, make them promising for computational tasks. Microsoft’s Station Q and Majorana fermions research initiatives are actively exploring this field.
3. *Adiabatic Quantum Computing (AQC)*: Based on the idea of Adiabatic Evolution [40], this paradigm involves gradually transforming a quantum system from an initial Hamiltonian to a final Hamiltonian that encodes the solution to a problem. A popular approach to AQC uses the annealing process [41], where the system starts in the ground state of the initial Hamiltonian and slowly evolves to the final Hamiltonian through a controlled annealing process via the application of external controls, such as electromagnetic fields or laser pulses. Therefore, by manipulating the system’s Hamiltonian over time, the system can be driven to achieve the ground state of the final Hamiltonian, representing the solution to the problem. D-Wave Systems develops quantum annealers that implement adiabatic quantum computing.

Due to hardware limitations, currently, we are in the so-called Noisy Intermediate-Scale Quantum (NISQ) era [42], in which quantum computers have several errors (connectivity, gate errors, coherence times, etc.) associated at the hardware level. The development process in Quantum Computing can be seen as two different but intertwined pathways: hardware and software development. Any software research and development needs equal support from the hardware or engineering side. The current quantum computers are said to be in the NISQ era, where the solutions are error-prone. To attain a meaningful solution, we need error correction algorithms. Employing quantum computers for practical purposes includes hardware and software expertise, and it might take considerable time before we can see a fully functional pure quantum computer.

Generally, a Hamiltonian of a physical system is an operator corresponding to the total energy of the system [43]. Considering a two-level quantum system, such as a qubit, this can be written as:

$$H(t) = \frac{\omega(t)}{2}\sigma_z + \frac{\Delta(t)}{2}\sigma_x \quad (2.1)$$

Where:

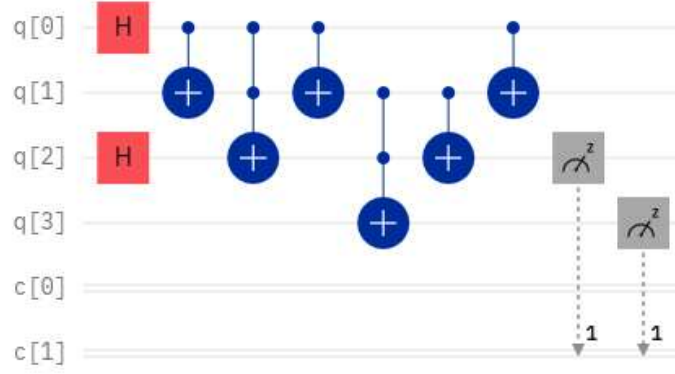


Figure 2.2: Quantum Circuit for Full Adder [using IBMQ (<https://quantum-computing.ibm.com/>)]

- $\omega(t)$ represents the time-dependent energy splitting between the two levels.
- $\Delta(t)$ represents the time-dependent strength of the interaction between the two levels.
- σ_z and σ_x are the Pauli matrices.

This transformation of this initial Hamiltonian (say H_0) is achieved by gradually changing the functions $\omega(t)$ and $\Delta(t)$ over time by applying a sequence of laser pulses with carefully designed amplitudes and frequencies to modulate these parameters. The time evolution of such quantum systems is given by the Schrödinger equation:

$$\hat{H}|\Psi(t)\rangle = i\hbar\frac{\partial}{\partial t}|\Psi(t)\rangle \quad (2.2)$$

Where:

- $|\Psi(t)\rangle$ is the quantum state of the system at time t .
- $H(t)$ is the Hamiltonian operator(observable) of the system at time t .
- $\hbar$ is the reduced Planck constant.

Thus, given $H(t)$, the goal of optimization is often to find the quantum state $|\Psi(t)\rangle$ that minimizes (or maximizes) an energy functional. This energy functional could be related to the total energy of the system, expectation values of certain observables, or other physical quantities of interest. Now, in complex potential landscape, the Hamiltonian $H(t)$ can be written as:

$$H(t) = \frac{\omega(t)}{2}\sigma_z + \frac{\Delta(t)}{2}\sigma_x + V(x, t) \quad (2.3)$$

Where:

- $V(x, t)$ represents the complex potential landscape, which can depend on both position x and time t .

- $\omega(t)$ and $\Delta(t)$ are time-dependent parameters as described earlier.

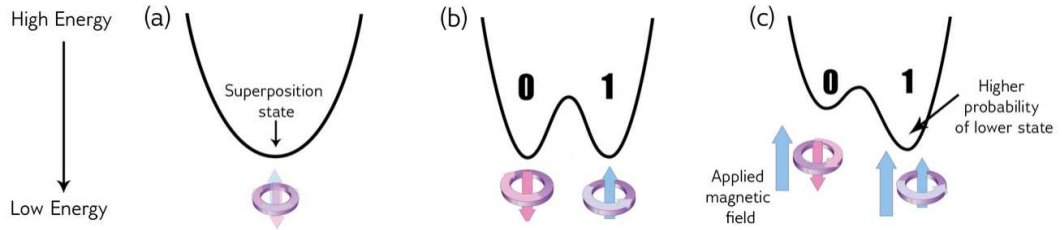
Considering an optimization problem where the aim is to find the quantum state $|\Psi(t)\rangle$ that minimizes the expectation value of the energy functional, denoted as $E[|\Psi(t)\rangle]$. This energy functional can be expressed as:

$$E[|\Psi(t)\rangle] = \langle \Psi(t) | H(t) | \Psi(t) \rangle \quad (2.4)$$

This is an optimization problem where the objective is to minimize $E[|\Psi(t)\rangle]$ with respect to $|\Psi(t)\rangle$ while satisfying any relevant constraints. Throughout the optimization process, the quantum state $|\Psi(t)\rangle$ is iteratively updated to approach the state that minimizes the energy functional. This involves evaluating the energy functional for different candidate states and adjusting the parameters of the quantum state accordingly. Once the optimization process converges, the resulting quantum state $|\Psi(t)\rangle$ represents the solution that minimizes the energy functional within the given constraints [44, 45, 46].

Quantum annealing is a specialized paradigm tailored primarily for addressing optimization problems. D-Wave's quantum annealing systems, exemplified by the Advantage_system5.3, harness qubits to navigate the energy landscapes associated with these optimization challenges, seeking to identify the configurations with the lowest energy, which corresponds to optimal solutions².

The intricate process of quantum annealing can be visualized through an energy diagram that evolves over time Figure 2.3³. Initially, the energy landscape resembles a single valley with a single minimum, representing a unique solution. As quantum annealing commences, a barrier is raised, transforming the landscape into a double-well potential. Here, the left valley corresponds to the 0 state, while the right valley signifies the 1 state. The qubit ultimately settles in one of these valleys at the end of the annealing process (Figure 2.4⁴). By default, there's an equal probability of it ending up in either state (50 percent).



Source: <https://www.dwavesys.com/>

Figure 2.3: Energy change diagram during Quantum Annealing.

However, the quantum annealing process can be controlled by applying an external magnetic field to the qubit. This tilts the double-well potential, increasing

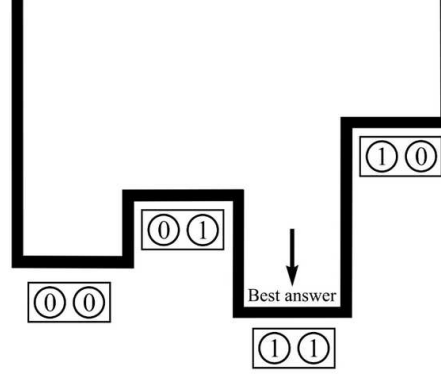
²<https://cloud.dwavesys.com/leap/>

³https://docs.dwavesys.com/docs/latest/c_gs_2.html

⁴https://docs.dwavesys.com/docs/latest/c_gs_2.html

the likelihood of the qubit ending up in the lower well. This programmable external magnetic field is called a bias, and the qubit adapts to minimize its energy under the influence of this bias.

The qubits are interconnected via devices called couplers to influence each other, leading to phenomena like entanglement. In an entangled state, two qubits collectively possess four possible states, determined by the biases and coupling strengths. During annealing, qubit states may span these states before settling into the optimal configuration.



Source: <https://www.dwavesys.com/>

Figure 2.4: Energy diagram - Best answer.

Drawing reference from the Hamiltonian representations discussed previously, D-Wave's quantum annealing⁵ approach can also be described in terms of a Hamiltonian, where the ground state of the system corresponds to the solution of the problem, which can be represented as:

$$\mathcal{H}_{ising} = \underbrace{-\frac{A(s)}{2} \left(\sum_i \hat{\sigma}_x^{(i)} \right)}_{\text{Initial Hamiltonian}} + \underbrace{\frac{B(s)}{2} \left(\sum_i h_i \hat{\sigma}_z^{(i)} + \sum_{i>j} J_{i,j} \hat{\sigma}_z^{(i)} \hat{\sigma}_z^{(j)} \right)}_{\text{Final Hamiltonian}} \quad (2.5)$$

Where,

- $\hat{\sigma}_{x,z}^{(i)}$ are Pauli matrices operating on a qubit q_i .
- h_i and $J_{i,j}$ are the qubit biases and coupling strengths⁶.
- $A(s)$ representing the tunneling energy and the $B(s)$ the problem Hamiltonian energy at s , the normalized anneal fraction, an abstract parameter ranging from 0 to 1.
- Initial Hamiltonian represents the lowest-energy state with all qubits in superposition of 0 and 1, also known as tunneling Hamiltonian.
- Final Hamiltonian encodes the solution of the problem being addressed where lowest-energy end state is classical, also known as problem Hamiltonian.

⁵https://docs.dwavesys.com/docs/latest/c_gs_2.html#id1

⁶https://docs.dwavesys.com/docs/latest/c_gs_4.html#getting-started-topologies

In a linear annealing process, $s = t/t_f$, where t is time and t_f is the total annealing time. Initially, $A(0) \gg B(0)$, leading to a quantum ground state. As annealing progresses, A decreases and B increases until t_f , when the qubits reach a low-energy solution state. At the end, only the $B(s)$ term remains in the Hamiltonian.

QML represents a compelling fusion of quantum principles and classical machine learning algorithms, promising breakthroughs in data analysis, optimization, and pattern recognition. One prominent player in this field is D-Wave Systems, which has championed quantum annealing technology. D-Wave Systems leads in quantum annealing with its Advantage_system5.3 by enabling QBoost [47], which is built on the principles of boosting algorithm and enhances binary classification using quantum computing. Quantum annealers complement QML by tackling optimization tasks and enhancing quantum machine learning models. IBM Quantum Computing is another notable contender, offering gate-based quantum computers providing QML algorithms like Variational Quantum Classifiers (VQC), Quantum Neural Network (QNN), and Quantum Support Vector Machine (QSVM). The synergy of quantum annealing, gate-based quantum computing, and innovative algorithms like QBoost holds promise for quantum machine learning and optimization tasks across various domains, including Smart City security.

2.3 Security and Privacy in QSE

Quantum Software Engineering (QSE) is an emerging discipline that adapts classical software engineering principles to the quantum domain. QSE integrates principles from quantum computing, software engineering, and programming methodologies to exploit the advantages of quantum computers, particularly in processing speed over classical systems [48, 49]. A structured framework for the quantum software lifecycle is essential, accommodating the hybrid nature of quantum applications that combine quantum and classical components [50, 51]. Thus, at the core of QSE is the Quantum Development Lifecycle (QDLC) [52, 53], which addresses the need for systematic quantum software development with defined phases like design, implementation, testing, and maintenance. Quantum programming languages and tools like QISKit, Q#, Cirq, and Forest are crucial for developing quantum programs and addressing complex control flow and synchronization issues [48]. The quantum circuit model, which employs universal gates, is fundamental for implementing quantum algorithms [54]. Verification frameworks such as CoqQ ensure the correctness of quantum programs through formal proofs [55]. The evolution of quantum software engineering responds to the limitations of classical computing and the challenges posed by the sensitivity of quantum states [56]. As the field matures, developing robust methodologies will be vital for maximizing the potential of quantum technologies across various domains [57]. Moreover, the Talavera Manifesto further underlines the principles and practices aimed at structuring and formalizing QSE processes.

2.3.1 Quantum Development Lifecycle (QDLC)

The Quantum Development Lifecycle (QDLC) is a structured framework designed to guide the development of quantum software applications, addressing the unique challenges posed by quantum computing. Unlike classical software development lifecycles, QDLC incorporates both quantum and classical components, reflecting the hybrid nature of most quantum applications today [51, 52], which is particularly relevant in the current Noisy Intermediate-Scale Quantum (NISQ) era. As mentioned in Figure 2.5 the lifecycle typically includes phases such as quantum feasibility study, requirement specification, system design, software coding and implementation, testing, and quality management [52]. This comprehensive approach is necessary due to the interdisciplinary nature of quantum computing, which requires a shared understanding among developers and researchers to effectively harness quantum capabilities [50, 51].

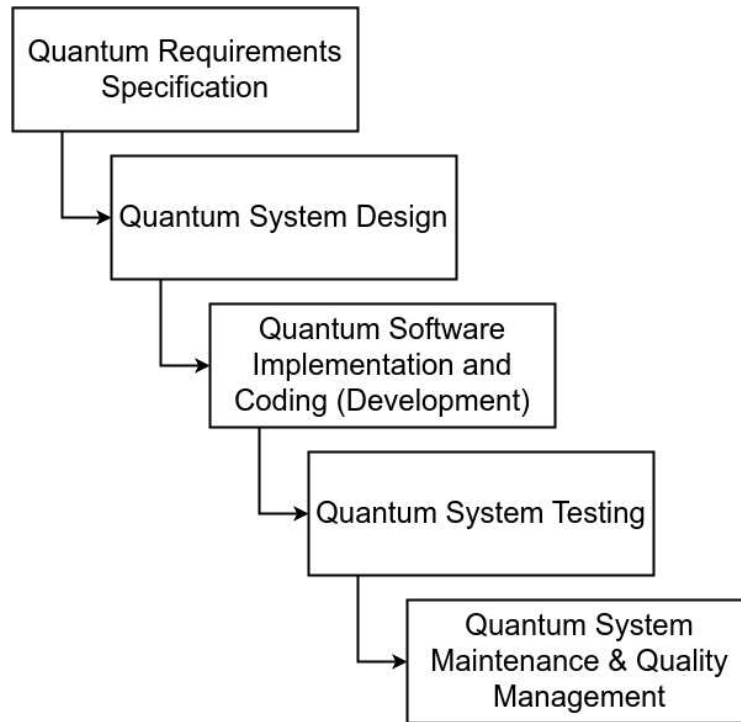


Figure 2.5: Quantum Development Lifecycle

The QDLC also emphasizes the integration of security and privacy measures from the outset, as highlighted by the Quantum Privacy-Oriented Software Development (QPOSD) approach, which ensures these critical aspects are embedded throughout the development process. Furthermore, the lifecycle addresses the orchestration of quantum and classical programs through workflows, which is essential for the execution of hybrid applications. The QDLC serves as a foundational model for developing high-quality quantum applications, encouraging innovation and providing a roadmap for future research and development in the rapidly evolving field of quantum computing [58]. As quantum computing continues to advance, the QDLC will play a crucial role in operationalizing quantum technologies and supporting the development of

industrial-centric quantum software applications.

2.3.2 Talavera Manifesto

The Talavera Manifesto [59] represents a collective effort by both academic and industry practitioners to establish foundational principles and commitments for the burgeoning field of quantum software engineering (QSE). The manifesto serves as a call to action, urging stakeholders to address the unique challenges and opportunities presented by quantum computing technologies. It emphasizes the need for a structured approach to software development in the quantum realm, which is crucial given the distinct nature of quantum computing compared to classical computing paradigms. The importance of the Talavera Manifesto lies in its role as a guiding framework that seeks to unify efforts across different sectors to advance quantum software engineering, ensuring that the development of quantum technologies is both systematic and collaborative. The key points and their importance have been elaborated below:

1. **QSE is agnostic regarding quantum programming languages and technologies.** QSE aims to deliver quantum software through processes and methodologies that are comprehensible, controllable, and reproducible by a wide range of communities.
2. **QSE embraces the coexistence of classical and quantum computing** and promotes the application of re-engineering techniques to incorporate new quantum algorithms into existing classical information systems. Additionally, reverse engineering methods are essential for parsing and abstracting quantum program information intended for integration into classical systems.
3. **QSE supports the management of quantum software development projects**, ensuring that the quantum software produced aligns with the initial business objectives and requirements while also adhering to quality, time, and cost constraints. It is crucial to either create or adapt methodologies for developing quantum programs. Furthermore, techniques for estimating efforts in quantum software development must be established.
4. **QSE considers the evolution of quantum software**, advocating for its maintenance and progression from inception to decommissioning. The evolution of quantum software must be managed throughout the entire lifecycle of quantum software.
5. **QSE aims at delivering quantum programs with desirable zero defects.** It is responsible for defining and implementing testing and debugging methodologies for quantum programs to ensure that the majority of defects are identified and rectified before the program's release.
6. **QSE assures the quality of quantum software.** Effective quality

management for both the processes and the products is crucial for producing quantum software that meets the expected quality standards. Since improvement is contingent upon measurement, it is necessary to develop new metrics for evaluating quantum programs and processes.

7. **QSE promotes quantum software reuse.** QSE can assist development teams in sharing, indexing, and locating reusable quantum software. The exploration of design and architectural patterns for quantum programs is essential, as is the facilitation of technical communication and the creation of reference libraries and application demonstrations.
8. **QSE addresses security and privacy by design.** Quantum information systems must be designed to be secure and to protect the privacy of both data and users. QSE aims to consider security and privacy from the initial stages of quantum software development, that is, by design.
9. **QSE covers the governance and management of software.** Managers should be cognizant of the unique processes, organizational structures, principles, policies, frameworks, information, culture, ethics, behaviors, personnel, skills, and competencies, as well as the services, infrastructure, and applications related to quantum software that organizations provide or should provide.

The Talavera Manifesto establishes key principles and best practices to guide the evolution of quantum software engineering. Serving as a unifying framework, it fosters a systematic and collaborative approach to developing quantum technologies. Its importance in shaping the future of QSE is mentioned below.

- **Guides QSE Development:** Lays the foundation for Quantum Software Engineering as a coherent discipline.
- **Accelerates Quantum Advancements:** Aligns software development with quantum hardware progress.
- **Fosters Collaboration:** Bridges knowledge gaps across domains.
- **Encourages Ethical Innovation:** Addresses societal impacts proactively.
- **Provides a Roadmap:** Serves as a guideline for researchers, educators, and policymakers to focus efforts effectively.

The foundational principles of the Talavera Manifesto align closely with the aims of this thesis, particularly regarding security, privacy, and structured methodologies in quantum software development. For instance, the manifesto explicitly emphasizes the importance of incorporating security and privacy into quantum systems right from the earliest design stages, reflecting precisely the first research goal of this thesis involving a systematic evaluation and adaptation of classical Security by Design (SbD) and Privacy by Design (PbD) frameworks specifically tailored for quantum software engineering.

In addition, the Talavera Manifesto underscores the necessity of rigorous testing

and validation to ensure quantum software achieves high quality with minimal defects. This directly relates to the second research goal of this thesis, which involves proposing and empirically validating novel methodologies aimed at enhancing security and privacy in quantum software. The proposed methodological framework involves detailed case studies and comparative analyses, supporting the principles set forth by the manifesto.

Furthermore, the manifesto advocates for effectively combining classical and quantum computing through structured software management approaches. This aligns strongly with the third research goal, which seeks to adapt Agile methodologies such as Scrum, DevSecOps, and MLOps for the quantum context. Our research leverages these iterative approaches to continuously strengthen security and privacy considerations throughout the quantum software development lifecycle.

In summary, the Talavera Manifesto serves as a fundamental guiding framework, reinforcing a clear connection between its established principles and the objectives pursued throughout this research.

2.3.3 Integrating Security and Privacy

Integrating Security by Design (SbD) and Privacy by Design (PbD) principles into quantum systems is critical, as quantum technologies offer both opportunities and threats to security and privacy. Existing frameworks for SbD and PbD have evolved from classical settings but require significant adaptations to effectively safeguard quantum systems. These adaptations focus on ensuring secure coding practices, minimizing attack surfaces, and leveraging quantum-enhanced security measures like Quantum Key Distribution (QKD).

Most of the work on privacy and security in this area was related to QKD or Quantum Cryptography [60]. In an in-depth survey about the evolution of Quantum Key Distribution [61], the authors introduce the basic elements, followed by reviewing the development of QKD networks in contemporary literature and their implementation in practice. In [62], the authors propose a novel QKD-based solution for developing secure network infrastructure to accommodate multiple applications. In a study on quantum cryptography, Janani et al. [63] proposed a two-level quantum cryptosystem to augment security for medical images and their sensitive content. Therefore, looking at the current scenario and contemporary research topography, it becomes evident that the area of QSE involving Privacy and Security by Design is mostly untouched.

Having said that, this thesis adopts the *7 Foundational Principles* as proposed by Ann Cavoukian (Table:2.1)[64] to translate the principles of QPbD from classical PbD. The principles of Security by Design were adopted from the OWASP's ten principles (Table:2.2)⁷. Although other standards like NIST and

⁷<https://github.com/OWASP/DevGuide/blob/master/02-Design/01-Principles%20of%20Security%20Engineering.md>

Table 2.1: Privacy by Design Principles

Foundational Principles of Privacy by Design
P1: Proactive not Reactive; Preventive not Remedial - Anticipates and prevents privacy-invasive events before they happen.
P2: Privacy as the Default Setting - Deliver the maximum degree of privacy by ensuring that personal data are automatically protected in any given system.
P3: Privacy Embedded into Design - Privacy by Design is embedded into the design and architecture of IT systems and business practices.
P4: Full Functionality – Positive-Sum , not Zero-Sum - Accommodate all legitimate interests and objectives in a positive-sum "win-win" manner.
P5: End-to-End Security – Full Lifecycle Protection - Strong security measures are essential to privacy, from start to finish.
P6: Visibility and Transparency – Keep it Open - All business practices or technology involved are operating according to the stated promises and objectives, subject to independent verification.
P7: Respect for User Privacy – Keep it User-Centric - Keep the interests of the individual primary.

NCSC exist, we have followed OWASP per our laboratory protocol. Therefore, since there are no existing Security and Privacy by Design principles The main approach is to adapt the existing classical principles of security and privacy by design into the Quantum Software Development Lifecycle.

2.4 Empirical Validation in QSE

Quantum software development is characterized by several challenges, particularly during the NISQ era, where quantum hardware is noisy and error-prone. Hybrid quantum-classical systems, which combine quantum processors with classical control units, present unique integration challenges. Effective error mitigation and synchronization between classical and quantum components are essential to maintaining system robustness, security, and privacy.

One significant challenge is the integration of security and privacy measures throughout the Quantum Software Development Life Cycle (QDLC), as highlighted by the Quantum Privacy-Oriented Software Development (QPOSD) approach, which emphasizes the need for privacy by design principles and strategies to address quantum-specific vulnerabilities and bugs [65]. The complexity of quantum computing systems themselves is a major barrier to adoption, as they introduce unique privacy challenges that require novel privacy engineering solutions to protect sensitive data from unauthorized access [66].

[67] and [2] highlight the gaps in quantum software security practices, emphasizing vulnerabilities like crosstalk and the need for robust protection measures. [68] further identified errors and predicted success rates using metrics such as

Table 2.2: Security by Design Principles

OWASP Principles of Security by Design
S1: Minimise attack surface area - Restricts the functions that users are allowed to access to reduce potential vulnerabilities.
S2: Establish secure defaults - The application must be secure by default.
S3: The principle of Least privilege - A user should have the minimum set of privileges required to perform a specific task.
S4: The principle of Defence in depth - Multiple security controls that approach risks in different ways is the best option for securing an application.
S5: Fail securely - Applications should fail in a secure way; should not give the user additional privileges.
S6: Do not trust services - Never trust third-party services from a security perspective.
S7: Separation of duties - Individuals should have their system accesses separated by duties.
S8: Avoid security by obscurity - Maintain sufficient security controls in place to keep your application safe without hiding core functionality or source code.
S9: Keep security simple - Complex mechanisms can increase the risk of errors.
S10: Fix security issues correctly - Programmers should be careful to identify all affected systems.

Quantum Vulnerability Factor (QVF). Regarding bugs in quantum platforms, [69] introduced Bugs4Q, a benchmark for real-world bugs, and proposed bug pattern categorization [70] to aid debugging in Qiskit. Later, QChecker was introduced by [71] as a static analysis tool to detect bugs via syntax trees without code execution. [72] introduced MorphQ, a metamorphic testing approach to identify quantum-specific bugs systematically, while their empirical study [1] emphasized differences between quantum and classical bugs.

Other studies focus on quantum bug characteristics. [73] identified 125 GitHub-hosted quantum projects and found gate operation and state preparation, in particular, to be more prone to bugs, while [74] identified math-related bugs as the most prevalent among a collection of 96 real-world quantum bugs. [75] presented Q-PAC, an automated tool for detecting bug-fix patterns, and [76] categorized various quantum circuit bugs to support systematic debugging during the testing phase.

Quantum blockchain technology, while offering enhanced security through its decentralized and tamper-proof nature, still faces privacy issues related to unauthorized access and security breaches, necessitating robust security policies and tools to mitigate these threats [77]. Additionally, the reliance on cloud-based quantum computing services introduces risks of intellectual property theft

and tampering, as untrusted third-party services can exploit vulnerabilities in the quantum computing stack, including through fault injection attacks and incorrect qubit calibration [2].

The noise and errors inherent in current quantum computing technologies further complicate security efforts, as they can mimic malicious attacks, leading to challenges in distinguishing between genuine threats and system imperfections [78]. Moreover, the development of privacy-preserving protocols, such as those for quantum federated learning, is crucial to prevent data leakage in distributed quantum computing environments, where gradient inversion attacks pose significant risks [79]. Overall, addressing these challenges requires a comprehensive approach that integrates security and privacy considerations into every phase of quantum software development, leveraging innovative protocols and methodologies to safeguard sensitive information in the quantum era [65, 80, 81].

Quantum algorithms such as Shor’s and Grover’s have profound implications for classical cryptography. Shor’s algorithm poses a critical threat to public-key cryptographic schemes by efficiently factoring large integers, rendering RSA and other encryption protocols vulnerable [34, 82]. This necessitates the integration of Quantum Key Distribution (QKD) to secure data transmission. Similarly, Grover’s algorithm provides a quadratic speedup for unstructured search problems, significantly reducing the effective key length of symmetric key cryptosystems like AES [35]. These developments underscore the urgent need to incorporate Security by Design (SbD) and Privacy by Design (PbD) principles into quantum software engineering to ensure resilience against evolving threats.

Shor’s algorithm employs quantum mechanics, including superposition and entanglement, to efficiently solve the factorization problem. The process begins with placing qubits into a superposition, allowing exploration of multiple possibilities simultaneously. The core computation involves period finding through the quantum Fourier transform, enabling efficient extraction of periodicity. Once measured, classical post-processing extracts prime factors, making the approach significantly faster than classical factorization methods. The efficiency of this algorithm in breaking cryptographic keys highlights the necessity of quantum-safe cryptographic alternatives.

Grover’s algorithm, on the other hand, accelerates search operations in unsorted databases, reducing the number of queries required from $O(N)$ to $O(\sqrt{N})$. The algorithm relies on superposition to represent all database entries and utilizes an oracle function to mark the correct solution. Through iterative amplitude amplification, Grover’s algorithm increases the probability of obtaining the correct answer in a fraction of the time required by classical methods. This has direct implications for symmetric key cryptosystems, as it effectively halves the security level of encryption. For instance, an AES-128 key would provide only 64-bit security against a quantum adversary, necessitating the use of longer key lengths to maintain security parity.

Quantum annealing further extends the impact of quantum computing on

cryptographic security by solving optimization problems through the transformation of cryptographic challenges into Quadratic Unconstrained Binary Optimization (QUBO) formats [83]. This approach is particularly concerning for cryptographic algorithms like SHA-256 and AES, as it allows certain cryptographic problems to be efficiently processed using quantum annealers. The transformation process involves representing cryptographic operations as Boolean expressions, linearizing these expressions into algebraic equations, and formulating them as a QUBO problem solvable by quantum annealers. While the quantum resource requirements for cryptanalysis remain high—breaking AES-256 may require approximately 250,000 qubits, while reversing SHA-256 demands around 50,000 qubits—ongoing advancements in quantum hardware could reduce these barriers over time.

Quantum Key Distribution (QKD) emerges as a promising countermeasure to these threats, leveraging quantum mechanics to enable secure communication resistant to eavesdropping [60]. By utilizing quantum entanglement and superposition, QKD allows two parties to securely share encryption keys while detecting any interception attempts [84]. The fundamental advantage of QKD lies in its reliance on quantum principles such as the no-cloning theorem and measurement disturbance, which ensure that any eavesdropping attempt introduces detectable anomalies.

Among the most well-known QKD protocols, BB84, introduced by Bennett and Brassard, encodes key information in quantum states using two complementary bases [85]. The security of the protocol is ensured through basis reconciliation and error detection, with privacy amplification mitigating any potential information leakage. Another simplified protocol, B92, reduces the resource overhead by using only two non-orthogonal quantum states for encoding [86]. In contrast, the E91 protocol, proposed by Ekert, relies on quantum entanglement to establish secure keys and detects eavesdropping through Bell’s inequality violations [87]. These protocols form the foundation of quantum-secure communication by preventing adversaries from intercepting cryptographic keys without introducing detectable disturbances.

The practical implementation of QKD has evolved significantly, transitioning from theoretical frameworks to real-world applications. The Micius satellite demonstrated QKD over long distances, paving the way for global quantum-secured communication [88]. Commercial enterprises such as ID Quantique and Toshiba have also developed QKD systems tailored for enterprise security applications [89]. Despite its promise, QKD faces challenges related to cost, scalability, and integration with classical networks. Optical fiber-based implementations suffer from signal loss over long distances, necessitating quantum repeaters for extended-range communication. Additionally, ensuring seamless interoperability between quantum and classical security infrastructures remains an ongoing research challenge.

On the other hand, quantum computing is increasingly being explored for its potential in enhancing security and privacy, particularly in intrusion detection

systems (IDSs) and threat assessment. While research on the direct application of quantum computing to cybersecurity remains in its early stages, several studies have proposed quantum and quantum-hybrid approaches for improving detection mechanisms and response times. These efforts primarily focus on quantum optimization, quantum-enhanced machine learning, and hybrid quantum-classical deep learning models.

One key direction of research involves using quantum optimization techniques for security detection. In [90], a quantum optimization framework was proposed for IoT security, leveraging the advantages of quantum computing for anomaly detection. Similarly, Radha et al. [91] introduced two quantum-inspired approaches—Quantum Swarm Evolutionary (QSE) and Quantum-inspired Evolutionary Algorithm (QEA)—to detect Denial-of-Service (DoS) attacks on cloud computing infrastructure. These studies demonstrate that quantum-inspired algorithms can significantly enhance security mechanisms by efficiently exploring large solution spaces.

With the rise of quantum computing, researchers have also investigated quantum machine learning (QML) techniques as replacements for traditional machine learning models [90, 92]. QML algorithms, such as quantum-enhanced support vector machines and quantum neural networks, have shown potential in improving IDS performance by reducing training and inference time while maintaining high detection accuracy. In [93], researchers proposed an IDS utilizing QBoost, a quantum annealer algorithm, to detect DoS and Fuzzy attacks. The results demonstrated that QBoost significantly reduced training and prediction time compared to classical Support Vector Machines (SVMs), highlighting the advantages of quantum-enhanced learning models in accelerating security tasks.

Another promising approach involves hybrid quantum-classical models, where quantum computing is combined with classical deep learning techniques to enhance threat detection. These models are particularly effective in scenarios requiring feature extraction and anomaly detection in large datasets. For instance, Salek et al. [94] proposed a framework integrating a classical neural network with a quantum-restricted Boltzmann machine (RBM) for intrusion detection. The classical neural network extracts features from structured network traffic data, while the quantum RBM reconstructs the network state to identify anomalies and malicious traffic injections. Similarly, Wang et al. [95] developed a quantum-classical hybrid deep neural network (QHDNN) designed to classify raw data images and exclude anomalous patterns from the learned model, demonstrating its effectiveness in anomaly-based detection.

Further extending hybrid quantum computing for security, quantum-enhanced anomaly detection systems have been explored for cybersecurity applications such as botnet detection and fraud analysis. Suryotrisongko and Musashi [96] implemented a hybrid quantum-classical deep learning model to detect botnets. Their findings indicate that while the hybrid model performed well in some cases, classical deep learning models still outperformed it in several instances. Meanwhile, Chen et al. [97] proposed a hybrid quantum-classical

Long Short-Term Memory (LSTM) model for analyzing temporal security data. Their results showed that the hybrid LSTM model converged faster and achieved better accuracy than classical LSTM, suggesting that quantum-enhanced architectures can provide advantages in time-sensitive security tasks.

Generative models have also been explored in the security domain. In [98], researchers introduced a quantum-classical Wasserstein Generative Adversarial Network (WGAN) by replacing the classical generator with a hybrid quantum-classical neural network while maintaining the classical discriminator. This model was applied to fraudulent transaction detection, achieving performance on par with classical methods in terms of the F1-score using a credit card fraud dataset.

Beyond individual quantum-enhanced security models, Quantum as a Service (QaaS) for security incident detection and threat assessment is gaining traction. In [99], researchers proposed a QaaS-based architecture for security monitoring in Smart Cities, implementing a DLQC-based QBoost algorithm to classify threats in large-scale IoT data. The use of cloud-based quantum services to support real-time security assessments highlights the potential for integrating quantum computing into large-scale cybersecurity frameworks.

As quantum and hybrid quantum-classical technologies continue to advance, their integration into cybersecurity is expected to move beyond experimental research and into practical security infrastructures. With ongoing progress in quantum computing, these technologies have the potential to transform threat detection, response mechanisms, and cryptographic resilience. However, to maximize their benefits, it is essential to address challenges such as scalability, robustness, quantum noise, error correction, and hardware constraints. Ensuring the smooth incorporation of quantum-based security models into existing frameworks is critical for maintaining efficiency, reliability, and long-term security.

Given the growing threat posed by quantum computing to traditional cryptographic frameworks, there is a pressing need to develop cryptographic techniques capable of resisting quantum-based attacks. While Quantum Key Distribution (QKD) provides a foundational method for secure communication, its practical constraints highlight the necessity for complementary advancements in post-quantum cryptography. Quantum-resilient cryptographic methods, designed to withstand threats from both classical and quantum adversaries, are emerging as a crucial area of research. These approaches, including lattice-based, hash-based, multivariate, and code-based cryptography, aim to establish long-term security in an era where quantum computational power continues to expand.

Regarding quantum-resistant alternatives, various cryptographic techniques have emerged, leveraging mathematical problems believed to be intractable for quantum algorithms. One such approach is lattice-based cryptography, which relies on the hardness of lattice problems to resist quantum attacks [100]. This technique supports encryption, digital signatures, and key exchange, making it an essential component of secure communication [101, 102]. Schemes

such as NTRU, Ring-Lizard, and Kyber have demonstrated their ability to enhance blockchain privacy through linkable ring signatures [103, 102, 104]. The ongoing standardization efforts by NIST further highlight the importance of lattice-based cryptography in securing digital communication across both embedded systems and cloud computing [102].

Another promising quantum-resistant approach is hash-based cryptography, which employs secure hash functions to generate digital signatures. Unlike traditional methods that rely on integer factorization and discrete logarithms, which are vulnerable to quantum attacks, hash-based techniques provide an alternative that remains resistant to quantum threats [105, 103]. The SPHINCS+ scheme, for example, has been recognized for its strong quantum resistance and is currently a candidate in NIST's post-quantum cryptography competition [106]. Hash-based methods also play a crucial role in privacy-sensitive applications such as digital signatures and electronic voting [107, 108]. Although implementation challenges and computational efficiency remain concerns, ongoing research is focused on optimizing these techniques for broader practical adoption [109, 110].

Multivariate and code-based cryptography are also being explored as viable post-quantum security solutions [105]. Multivariate cryptography is founded on the complexity of solving NP-hard polynomial equations, making it resource-efficient [101]. However, its security is sometimes questioned, as certain implementations, such as those based on Hopfield cryptosystems, have been found vulnerable to quantum attacks [111]. On the other hand, code-based cryptography, including McEliece and Niederreiter schemes, leverages the difficulty of decoding error-correcting codes to provide strong post-quantum security [101, 112]. While the large key sizes required for these methods pose deployment challenges, recent research efforts aim to optimize them for applications such as multi-party computation and digital signatures [113]. These cryptographic approaches collectively form the foundation of post-quantum security, ensuring resilience in an era where quantum computing capabilities continue to grow.

The contribution to this area of research can be found in the following works [65, 93, 114, 115, 116, 117, 118, 99, 119].

2.5 Agile in QSE

Agile-based frameworks such as Scrum, DevSecOps, and MLOps have been adapted to enhance the flexibility and security of classical software engineering lifecycles. Scrum offers iterative development suited for exploratory quantum programming, while DevSecOps integrates security practices into the continuous development cycle, addressing vulnerabilities early. MLOps, an automation focused contemporary machine learning framework complementing a vast majority of commonly used software systems, can also be applied to privacy and security automation also for secure software enhancement.

The current literature is devoid of any studies in this field and a curated attention

into this area is essential. However, Agile methodologies for secure quantum software development are evolving to address the unique challenges posed by quantum computing. Traditionally, SCRUM have been adapted to incorporate security measures, as seen in the Security-Based SCRUM (SBSCRUM) and Trustworthy SCRUM models, integrating security imperatives into the agile process to mitigate threats and ensure robust software development [120, 121]. These adaptations are crucial as conventional security approaches often conflict with agile methodologies, necessitating a balance between agility and security [121].

In QSE, integrating security and privacy is paramount, as proposed by the Quantum Privacy-Oriented Software Development (QPOSD) approach, which embeds security protocols throughout the Quantum Software Development Life Cycle (QDLC) [65]. This integration is essential to prevent costly post-production fixes and ensure data integrity. Furthermore, the concept of crypto-agility, which involves the swift adaptation of cryptographic algorithms to counter quantum threats, is supported by agile practices such as Continuous Integration and Coding Standards [122].

The need for agile methodologies in quantum software development is underscored by the challenges of incorporating these practices effectively, as highlighted by empirical studies on agile practices in this domain [123]. Additionally, the Quantum Agile Development Framework proposes a hybrid project management approach that combines classical and quantum paradigms, addressing the new roles and requirements of quantum software projects [124].

DevSecOps and MLOps represent some of the critical advancements in software engineering, focusing on integrating security and machine learning into development processes. DevSecOps extends the DevOps methodology by embedding security practices throughout the software development lifecycle, fostering collaboration among development, operations, and security teams. MLOps, on the other hand, emphasizes the operationalization of machine learning models, ensuring their reliability and scalability in production environments. The integration of these practices enhances both security and efficiency in software development.

DevSecOps incorporates security as a fundamental component, addressing vulnerabilities early in the development process. It promotes teamwork among development, operations, and security, leading to a more resilient software lifecycle. A Challenge-Practice-Tool-Metric model is proposed to guide organizations in implementing DevSecOps effectively [125]. Whereas, MLOps focuses on deploying and maintaining machine learning models in production, ensuring they perform reliably. It leverages automation for continuous integration and deployment of ML models, enhancing efficiency. MLOps practices help in optimizing system performance and resilience through predictive analytics and anomaly detection [126, 127].

Unfortunately, these such specific frameworks are not explicitly detailed in the context of quantum software, the principles of integrating security into

agile processes are evident across these adaptations, emphasizing the need for continuous security integration and agile adaptability in the face of quantum computing advancements [65, 122].

Chapter 3

Frameworks and Models

During the past few decades, Quantum Computing (QC) has become one of the most discussed topics worldwide. Various countries have dedicated a substantial amount of their technological funds to acquiring a piece of the quantum supremacy pie. An analysis by McKinsey highlighted that some of the largest world economies have already announced plans to adopt Quantum Technologies [128]. Alongside, there is also a trend of investments that have started to flow into Quantum Technologies through various startups and big firms wading into the area of quantum computing [128]. Furthermore, QC has shown promises to make extensive forays in different fields, including but not limited to [129]: privacy and cryptography; economics and financial services; genetics, medicine, and health; chemistry; energy and agriculture; defense and national security programs; supply chain and logistics.

Looking at the current state of Software Engineering, we can clearly see that it stands on highly stable, rich, and robust frameworks, strategies, and models. Furthermore, these models or artifacts had been empirically tested and validated in detail before being implemented. Although the ideal expectation is to have a similar body of knowledge for Quantum Software Engineering so that a well-tested and stable model can govern all processes involved in quantum software development, the reality seems not so close.

There has been exponential growth in empirical studies in Quantum Software Engineering in recent years. The Quantum Computation community has carried out outstanding research and surveys in different areas like Quantum Computing taxonomy [130], Quantum formal methods [131], Quantum software architecture [132], Quantum Computing platforms [133], Quantum key distribution [61], and post-quantum cryptography [134]. It is clear that quantum computing research is not limited to the areas mentioned above, and in the coming years, the volume of research in this direction is bound to increase rapidly.

Quantum software engineering (QSE) research is still immature and needs considerable research, time, and effort to become robust like classical Software Engineering. Although recently, much research has been dedicated to the

proposal and development of the Quantum Software Development Life Cycle (QDLC) [52], the scopes of further research and development in this field are humongous. To achieve a mature state of QSE, it is imminent that we need a well-established framework and a concept of QDLC. An expected solution is to translate the Software Development Life Cycle (SDLC) to fit into the requirements of quantum computing [52]. However, when we dived deeper into QDLC and the commitments deemed necessary in the Talavera Manifesto [59] for efficient quantum software development, we found that most of the studies proposed the current status, prospects, algorithms, and newer horizons, and very little catered to the areas of privacy and security aligned with quantum software engineering.

This chapter aligns with the goals outlined in the introduction, focusing on reviewing existing frameworks and models within the three key areas. It presents a systematic analysis of security and privacy considerations, empirical validation techniques, and agile methodologies in quantum software engineering as outlined in the goals in section 1.2. The discussion includes an overview of relevant literature, identifying strengths and limitations, along with proposed models that enhance security and privacy in QSE.

3.1 Security and Privacy in QSE

A fundamental challenge in quantum software engineering is ensuring that security and privacy are embedded into the development lifecycle. As quantum computing progresses, its impact on cryptographic security becomes increasingly significant, necessitating proactive measures to mitigate potential risks. With the invention of Shor's algorithm [34], it became pretty clear that, on the one hand, quantum computing promises to usher in a new era of super-fast computational advancements; on the other, it also brings with itself the downsides of quantum-enabled cyber attacks. Shor's algorithm solves the 15-digit integer factorization problem (basis of the RSA protocol) in polynomial time, thereby paving the way for more insidious attacks on the existing as well as post-quantum systems. Such attacks are not real yet, so we can keep using RSA encryption until then. However, in recent times, there has been much research on post-quantum encryption proposing algorithms that can protect against such attacks [135, 136]. Thus, as mentioned in the Talavera Manifesto [59], it is the responsibility of QSE to address security and privacy by design during the software development process of the system. Therefore, Privacy by Design (PbD) and Security by Design (SbD) must be addressed strictly by QSE. We transpile the principles of privacy and security by design from classical to quantum/hybrid software engineering and evaluate the existing empirical studies based on them to investigate the current state-of-the-art on PbD and SbD. Doing so gives us a clear view of where we stand in QSE regarding Privacy and Security by design when planning the next steps in this area.

The existing reviews comprehensively cover most parts of the QDLC but do not investigate the situation of PbD and SbD in QSE. We investigate the

aforementioned gap in the discussed field and elucidate the contemporary state-of-the-art PbD and SbD in QSE. We focus on studies in the area of QSE that address, define, and implement the principles of PbD and SbD in the area of QSE. We try to find out the current status of the state-of-the-art PbD and SbD for QSE by translating the privacy and security ideas into the quantum domain. We aim to investigate the existing empirical evidence regarding the definition and implementation of QPbD and QSbD that fulfills the guidelines of privacy and security provided in the Talavera Manifesto [59].

3.1.1 Research Methodology

To understand the gaps better, we follow the guidelines of systematic literature reviews [137] to conduct the current study, which consists of three principal phases: planning, conducting the review, and reporting the results. Before starting the review, it is crucial to define a protocol. A well-defined protocol ensures that the steps and procedures to conduct the Systematic Literature Review (SLR) are adequately followed. It also ensures that the SLR does not suffer from researchers' bias and inclinations. The protocol developed for this study includes objectives, Research Questions (RQs), search strategy, inclusion/exclusion criteria (part of selection strategy), data extraction form, and data synthesis.

Provided the research goal and the motivations, we formulate the RQs mentioned in Table 3.1. The RQs would serve as this study's guide for the search and selection processes. In addition, the individual RQs would help us find the trend of studies in QSE as a whole and discover the studies specific to PbD and SbD. Further, this would also help us determine the existing research fields and technologies considered important in the area of privacy and security by design.

Search Strategy

The search strings were constructed following these steps [138]:

- (i) Define major terms.
- (ii) Identify alternative terms (alternative spellings and synonyms of major terms) and related terms (terms related to major terms).
- (iii) Check the keywords in papers we already have.
- (iv) Use the Boolean operator *OR* to incorporate alternative spellings, synonyms or related terms.
- (v) Use the Boolean operator *AND* to link major terms, or conjoined *OR* clauses.

Based on the RQs, we have selected the following major (search) terms: Quantum Software Engineering, Privacy, Security, and Design. We have enlisted the major, alternative, and related terms in Table 3.2. The primary domain of our study is *Quantum Software Engineering*, so we considered the different combinations of it, *Quantum*, and after that, the abbreviation. Regarding

Privacy, we considered terms related to the application of privacy in QSE and its abbreviation. For the term *Security*, we developed the search terms similarly. The area of QSE is in its nascent state (as mentioned in Section 1), and so are the studies in this area. Including more generic terms like *Software Engineering* in an *OR* clause in the search query would return multitudes of papers, and again, on the other hand, keeping the search query strictly to the specific major search terms would confine the results to significantly less or no papers at all. Thus, we avoided including specific classical software engineering search terms and used a combination of related terms with *AND* and *OR* clauses with specific terms.

Table 3.1: Research Questions and their Motivations.

Research Questions and Motivations
RQ1: In which venue (journals, conferences, or workshop proceedings) are papers on the application of PbD and SbD in QSE published? <i>Motivation:</i> Track the trend of venues where researchers publish studies about privacy and security by design in quantum software engineering (QSE).
RQ2: What are the most frequently applied empirical strategies to investigate the application of PbD and SbD in QSE? <i>Motivation:</i> Identify the types of empirical strategy most frequently used to investigate the application of PbD and SbD in QSE.
RQ3: What quantum technologies are used by the authors to address the privacy and security concerns in Quantum Software Development (QSD)? <i>Motivation:</i> Identify the technology trends used in contemporary quantum studies.
RQ4: In which phases of the quantum software engineering process (i.e., requirements engineering, system design, object design, testing, operation, maintenance) are the privacy and security-oriented quantum models implemented? <i>Motivation:</i> Understand the impact of the implementation of privacy and security in each phase of QSD/QSE.
RQ5: What research areas implemented the quantum privacy and security principles? <i>Motivation:</i> Find out the most sought-after research areas/domains for quantum PbD and SbD implementation.
RQ6: What do the studies contribute towards the Privacy by Design principles in the scope of QSE? <i>Motivation:</i> Identify the most important PbD principles addressed in the current studies.
RQ7: What do the studies contribute towards the Security by Design principles in the scope of QSE? <i>Motivation:</i> Identify the most important SbD principles addressed in the current studies.

After carefully choosing the major and alternative terms, along with some abbreviations generally used in the literature, the following search string was

Table 3.2: Major, alternative, and related terms

Major Terms	Alternative and Related Terms
Quantum Software Engineering	quantum, QSE
Privacy	privacy by design, PbD
Security	security by design, SbD, secure

formulated: ("*quantum*" **OR** "*quantum software engineering*" **OR** "*QSE*") **AND** ("*privacy*" **OR** "*privacy by design*" **OR** "*PbD*") **AND** ("*security*" **OR** "*security by design*" **OR** "*SbD*" **OR** "*secure*"). The searches were case-insensitive and applied to the paper title, abstract, keyword, and body. We searched for primary studies in *ACM Digital Library*, *SCOPUS*, *Web of Science*, *Science Direct*, *IEEEExplore*, *Springer*, *JSTOR*, and *Google Scholar*. The aforementioned libraries were selected based on the recommendations of Brereton et al. [138] and Dyba et al. [139]. We extended the search radius by adding *Google Scholar* to include all sources of publications like dissertations, theses, reports, and white papers. We practiced an automatic search over digital libraries or snowballing strategy as commonly used in SLRs. This study considered journal, conference, workshop, and arxiv papers.

Selection Strategy

In the current SLR, we included only papers written in English that presented empirical evidence about the definition, elaboration, and implementation of the principles of PbD and SbD in the area of QSE. The time frame for the selection criteria was set from January 2000 to December 2023. Since QSE is a nascent domain, and a small number of studies are relevant to our specific research domain; the time range has to be large enough to include all relevant studies.

We excluded studies that lack practical relevance, such as those not focused on privacy and security in quantum software engineering. Moreover, we exclude duplicate studies published in more than one database or more than one journal, research focusing on topics other than privacy and security, or studies where the principal or alternative search terms were merely used for reference or general introduction. We also exclude outdated research that doesn't align with current technologies and standards, ensuring our review remains relevant and useful.

Conversely, we included studies that use quantum technologies to boost privacy and security. We include research that proposes models for integrating these aspects into quantum software design and features real-world implementations or evaluations. This ensures the review covers practical, impactful, and methodologically sound research in quantum privacy and security. The inclusion/exclusion criteria have been outlined in Table 3.3 and Table 3.4.

Data extraction was carried out based on Table 3.5, which enlists the specific items and their significance and eventually maps the values to the specific research questions. In short, data extraction includes paper metadata (title, abstract, keywords, authors, affiliations), digital library source, publication type

Table 3.3: Inclusion Criteria

Title	Quantum, Privacy, Security
Abstract	Privacy and security in quantum or quantum-hybrid systems using quantum technologies
Papers	1. Papers that discuss the application of quantum technologies specifically aimed at enhancing privacy and security. 2. Studies that explore or propose methodologies, frameworks, or models for integrating security and privacy by design in quantum software engineering. 3. Research that includes empirical evaluations, case studies, or real-world implementations demonstrating the feasibility and effectiveness of quantum technologies in addressing privacy and security concerns. 4. Articles that perform comparative analyses between quantum and classical technologies concerning privacy and security. 5. Surveys, reviews, or case studies discussing privacy and security considerations throughout the quantum software development life cycle. 6. Papers analyzing quantum bugs and vulnerabilities, particularly those impacting privacy and security. 7. Articles providing reviews or forward-looking perspectives on the state-of-the-art and future research directions in quantum privacy and security by design.

and year, and classification of work. Additionally, we also extract technologies used, software development life cycle phase, research area, and relevance to Privacy and Security by Design principles. We primarily used WPS Office 2019 to assist in our data extraction and elaboration of graphical images for the study. Some images were illustrated using the open-source application drawio¹.

Table 3.5: Specific items for Data Extraction

Items	Value	Impact	Map RQs
Paper meta-data	What are the title, abstract, keywords, authors, and affiliations of the paper?	Essential for identifying and referencing papers.	-

¹<https://github.com/jgraph/drawio>

Digital library	What is the name of the digital library where the paper is found, e.g., ACM Digital Library, IEEEExplore, Google Scholar, etc?	Helps in assessing the credibility and accessibility of the research.	-
Year of publication	When was the paper published	Shows the temporal distribution of research. Useful for understanding the evolution of the field and identifying trends.	RQ1
Type of publication	What is the venue of the publication, for example, journal, conference, workshop, etc.?	Differentiates between journals, conferences, technical reports, etc. Helps in evaluating the quality and impact of the research.	RQ1
Type of work	What is the empirical method used or in other words, what is the type of work, for example, experiments, surveys, discussions, etc.?	Important for understanding the scope and approach of the research.	RQ2
Objectives	What is the technical objective/aim of the study?	Helps in understanding the research focus and expected contributions.	RQ3
Methodologies	What methodology was used to carry out this study (high level)?	Essential for assessing the validity and reliability of the findings.	RQ3
Technologies	What specific technology was used in this study?	Important for evaluating the practical applications and innovations.	RQ3
Phases of development process	During what SDLC phase was the study conceived?	Helps in understanding the maturity and progress of the research.	RQ4
Applied Research Area	What is the overall research area of the study?	Useful for contextual relevance and application of findings.	RQ5
Relevance to Privacy by Design Principles	Is the study relevant and adherent to the principles of Privacy by Design	Important for ensuring that privacy considerations are integrated into the design.	RQ6

Relevance to Security by Design Principles	Is the study relevant and adherent to the principles of Security by Design	Critical for ensuring that security measures are considered throughout the research process.	RQ7
---	--	--	-----

Table 3.5: Specific items for Data Extraction**Review Protocols**

We thoroughly reviewed the protocol before conducting the SLR. It was conducted by five researchers, all of whom have considerable experience in literature review or quantum software engineering. While a group of two authors defined the protocol, it was validated by a group of two other authors, and the final author acted to resolve any conflicts in the review. Similarly, during each review phase, like elaborating the research questions, primary studies search, or inclusion/exclusion criteria, one group designed the phases, and the other verified and validated the same; the other author acted to resolve conflicts and finalize the results. The protocol development process was carried out with iterative validation that led to different suggestions and comments at every step, leading to subsequent changes in the protocol and, therefore, the SLR.

We have used the term "paper" to refer to an article published in a journal, conference, workshop, or arxiv that has been retrieved and analyzed in our review. The terms "paper" and "article" have also been used synonymously. Any empirical study reported in these papers has been referred to as a "primary study" or "empirical study". The terms "author" and "researcher" have been used as quasi-synonyms. All involved researchers have considerable experience in the area of SLRs and the area about the practice of Privacy by Design and Security by Design in the domain of Quantum Software Engineering. We conducted the below-mentioned steps while conducting the SLR [140]:

1. Execute search string;
2. Remove duplicates;
3. Exclude irrelevant papers;
4. Select relevant papers;
5. Perform snowballing;
6. Carry out quality assessment;
7. Analyze papers.

The activity of executing the search string was carried out in two phases. To begin with, the first author group executed the search strings in the digital libraries using only major search terms to include as many studies as possible and then discussed the outcome with the second author group. In the second phase, the first group, this time, includes the complete set of search terms in the search query using the Boolean operators applicable to each digital

Table 3.4: Exclusion Criteria

Standards	1. Papers that are duplicate publications of the same study or do not add any new information or insights compared to previously included papers. 2. Articles from non-peer-reviewed sources, such as opinion pieces, editorials, and non-academic publications. 3. Papers that are not written in English.
Relevance	1. Papers that are not written in English. 2. Papers that do not primarily focus on quantum technologies or do not address privacy and security in the context of quantum software engineering. 3. Studies that focus on general quantum computing or quantum mechanics without specific emphasis on security and privacy by design. 4. Papers that are purely theoretical without any practical applications, empirical data, or case studies related to quantum privacy and security.

library respectively and executes the search query. After that, they applied the inclusion/exclusion criteria to the studies, which were then validated by the second group and vice versa. This reduced bias during the selection of the primary studies. The fifth author helped resolve conflicts when there was an impasse in the decisions. All researchers participated in the data extraction, validation, and analysis of the results. Figure 3.1 presents the workflow of the current study as a flowchart.

Execute search string: We execute the search strings in the digital libraries listed in Section 3.1.1. It is complex and cumbersome to get results of queries for SLRs or Systematic Mapping Studies (SMSs) from the search engines for the digital libraries mentioned above as these are based on different models [141]. Different digital libraries use different modeling designs to power their search engines and hence cannot be used in the same capacity to extract results with the same query. Therefore, we split the generic search query presented in Section 3.1 into two or more parts and had to fit into the constraints and formats of the respective digital library’s search engine. Although the search strings practically used during the exact searches differed from the one derived in the previous section, they carried the same essence in the query. They contained the same major and related terms as proposed in the protocol. The initial search retrieved 588 papers from various digital libraries with 269 papers from *IEEEExplore*, 112 from *Google Scholar*, 63 from *ACM Digital Library*, 56 from *SCOPUS*, 32 from *Web of Science*, 20 from *JSTOR*, 19 from *Science Direct*, 17 from *Springer*. We have used Rayyan[142], an online tool for maintaining and

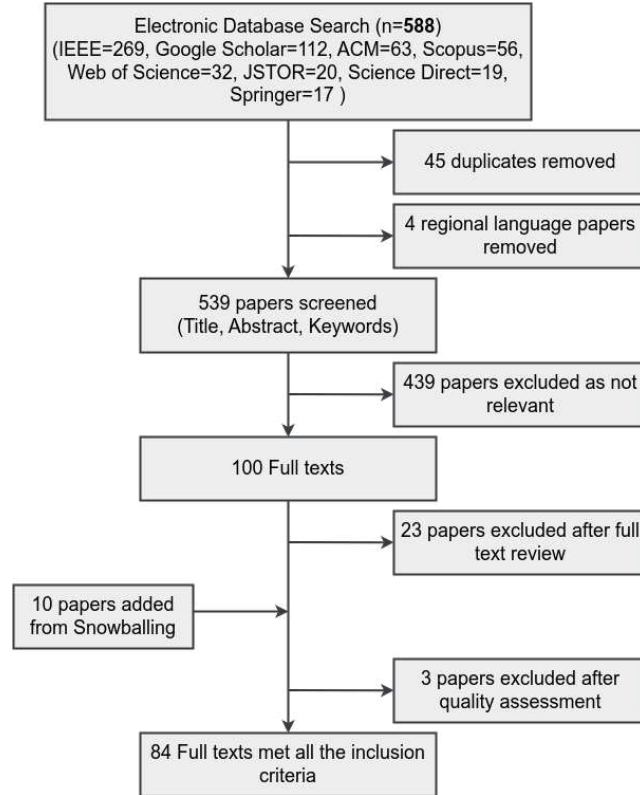


Figure 3.1: Systematic Literature Review flowchart.

processing the papers included in systematic studies. The BibTeX citations for all the papers were loaded into the Rayyan web application for further processing. In the subsequent steps, the papers were studied and then marked with inclusion and exclusion criteria. The web application’s functionality is out of this paper’s scope and can be easily found online on Rayyan’s website².

Remove duplicates: Due to using various digital libraries, our search returned some duplicate papers due to overlap between the digital libraries. Among the retrieved papers, 45 were duplicated during the primary search and were considered only once for the review. We could not find other discrepancies, like the same primary study published in multiple venues or extension papers. The duplicated papers were retrieved separately as part of proceedings or journals, so we removed the duplicates and went ahead with one entry for each of them. We found more than one SLR on QSE, but those relevant to the research area were not removed. Moreover, each QSE SLR or discussion presented a different point of view on different aspects of the area. After removing the duplicate papers, we were left with 543 papers for the next step.

Exclude irrelevant papers: Firstly, we excluded the papers written in languages other than English. We encountered two papers in Mandarin and two in Spanish, which were removed, bringing the count of papers to 539. We carefully read the titles, abstracts, and keywords mentioned in the extracted

²<https://rayyan.ai/>

papers to exclude the irrelevant ones from the SLR using Table 3.3 and Table 3.4.

To ensure a more comprehensive exploration of the field, we included research on all related areas of QSE, such as quantum cryptography, quantum key distribution, and network security, since they are integral to the broader context of quantum security and privacy, offering valuable insights into the overall landscape of QSE. Quantum cryptography and key distribution are crucial for secure communication in quantum networks, which is fundamental to building secure quantum systems. Network security in the quantum realm involves protecting data and communications within quantum networks, contributing to the overall security infrastructure. Including these topics in our systematic literature review (SLR) allows us to gain a more detailed understanding of the evolution of the QSE field and its future directions, as advancements in these areas can significantly influence the development of privacy and security in QSE. Based on the above observations, we further removed 439 papers from the list of identified papers, and we were left with 100 papers.

Select relevant papers: The 100 papers retrieved from the previous steps were grouped into two categories based on their alignment with the research area: *(A) 9 papers* were considerably aligned with *PbD and SbD in QSE* whereas *(B) 91 papers* needed further investigation. The existing papers, especially from the group *B*, were then re-analyzed based on the inclusion/exclusion criteria (Table 3.3 and Table 3.4). The relevant papers were included in the study if they specifically served the purpose of this SLR to contribute to privacy and security in QSE. We excluded 23 papers because although they represented some empirical studies on quantum computing, they primarily discussed benchmark studies of quantum algorithms, quantum taxonomy, reviews of quantum computing software architectures, quantum computing testing suites, and post-quantum encryption strategies. None of these are relevant to the area of interest of the current SLR. After careful scrutiny, we considered 77 papers for the next step; we included all nine papers from group *(A)* and sixty-eight from group *(B)*.

Perform snowballing: We conducted both forward and backward snowballing during full-text reviews, applying the aforementioned inclusion and exclusion criteria iteratively to the list of intermediate papers from the references and papers citing already included studies. This process identified 10 additional papers, bringing the total to 87.

Carry out quality assessment: We conducted a quality assessment on the final list of papers, adhering to the SLR guidelines. To evaluate the rigor of each empirical study, we adapted the criteria from Dybå and Dingsøyr [143, 144], a widely recognized quality assessment tool in software engineering [145]. The papers were rated based on an 11-question scale and categorized into three groups according to their scores: Low Quality (0-3 points), Medium Quality (4-7 points), and High Quality (8-11 points). The checklist used for this quality assessment can be found in Table 3.6. During this step three papers with low

quality were removed. The results of this step are provided in further detail in the results section.

Table 3.6: Quality Assessment Questions

Criteria	Categories	Questions
Reporting	Research	Is the paper based on research (or is it merely a "lessons learned" report based on expert opinion)?
	Aim	Is there a clear statement of the aims of the research?
	Context	Is there an adequate description of the context in which the research was carried out?
Rigor	Research design	Was the research design appropriate to address the aims of the research?
	Sampling	Was the recruitment strategy appropriate to the aims of the research?
	Control Group	Was there a control group with which to compare treatments?
	Data collection	Was the data collected in a way that addressed the research issue?
	Data analysis	Was the data analysis sufficiently rigorous?
Credibility	Reflexivity	Has the relationship between researcher and participants been considered to an adequate degree?
	Findings	Is there a clear statement of findings?
Relevance	Value	Is the study of value for research or practice?

Analyze papers: We thoroughly read and analyzed the remaining 84 papers, extracted data based on the protocol mentioned in Table 3.5, and recorded the data in WPS Excel 2019.

3.1.2 Data Synthesis and Results

This section presents the data synthesis and results of our systematic literature review (SLR) on Privacy and Security by Design in Quantum Software Engineering (QSE). From an initial pool of studies, we identified 84 relevant papers for in-depth analysis. These papers were evaluated based on criteria such as reporting, rigor, credibility, and relevance. Our synthesis aims to offer a comprehensive overview of current research trends, key findings, and gaps in the literature, highlighting the evolving landscape of privacy and security in QSE.

3.1.3 Answers to the Research Questions

We present insightful trends and patterns that address our research questions, shedding light on the current state of privacy by design (PbD) and security by design (SbD) principles in quantum software engineering.

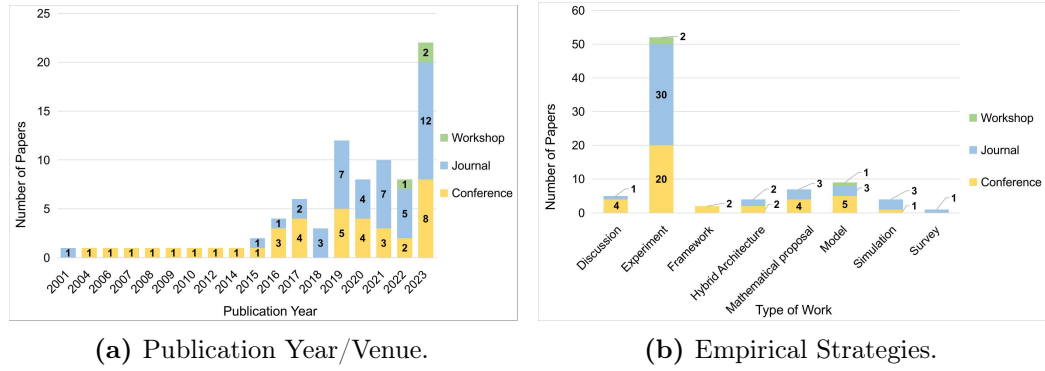


Figure 3.2: Distribution of papers across Years, Venues, and Type of Work (RQ1 and RQ2)

RQ1: *In which venue are papers published?*

Figure 3.2a illustrates the distribution of papers published across different years and venues (workshop, journal, and conference). From 2001 to 2014, the number of publications remained consistently low, with only one(1) paper published each year. The highest number of papers were published during the years 2019 to 2023: twelve, eight, ten, eight, and twenty-two, respectively, for each year. The first paper [146] to be published in this area was in 2001. The majority of the papers were published in journals and conferences. The selected studies include 43 journal papers, 38 conference papers, and three workshop papers. The papers were published in a relatively wide range of journals, where some are highly ranked. To count a few: Future Generation Computer Systems [147], Journal of Information Security and Applications [63], Journal Of Ambient Intelligence And Humanized Computing [148] and IEEE Transactions on Circuits and Systems I: Regular Papers [149]. There are also papers from highly ranked conferences, like the IEEE Conference on Nanotechnology [150] and IEEE/ACM International Conference on Computer-Aided Design [151].

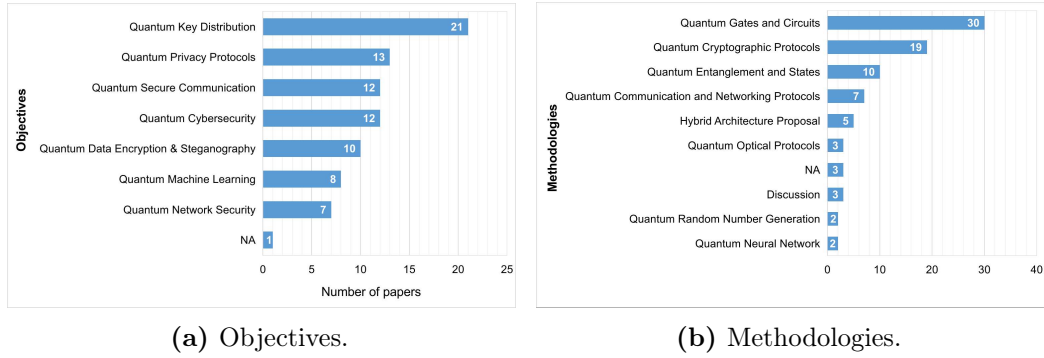
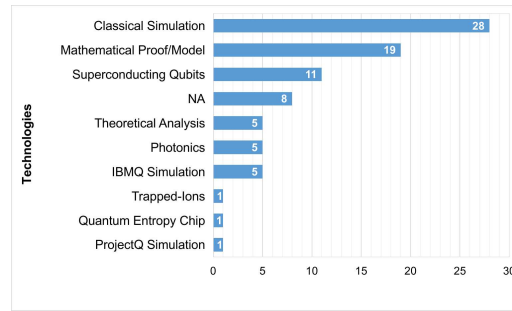
RQ2: *What are the most frequently applied empirical strategies?*

Figure 3.2b shows experimentation is the most popular empirical strategy used in the studies included in the review. The conducted studies consist of 52 experiments, more than 60% of the studies, followed by nine model proposals as the next preferred empirical strategies. Apart from these, the selected studies include seven mathematical proposals, five discussions, four simulations, four hybrid architecture proposals, two frameworks, and one survey.

RQ3: *What quantum technologies/methodologies are mostly sought after to address privacy and security by design in QSE?*

To answer the question, we adopted a three-tiered approach. First, we examined the primary objectives of the studies included in our analysis, and then, we explored the methodologies applied within these studies. Finally, we discussed the specific technologies utilized to implement these methodologies.

Figure 3.3a illustrates the distribution of papers by their objectives. Quantum Key Distribution is the most prominent topic, with 21 papers (25%) address-

**Figure 3.3:** Objectives and Methodologies used (RQ3).**Figure 3.4:** Technologies used (RQ3).

ing it. This highlights its critical importance in quantum communications. Quantum Privacy Protocols and Quantum Secure Communication are also significant, with 13 (15.48%) and 12 papers (14.29%) respectively. Quantum Cybersecurity matches this interest with 12 papers (14.29%). Other notable topics include Quantum Data Encryption & Steganography (10 papers, 11.9%), Quantum Machine Learning (8 papers, 9.52%), and Quantum Network Security (7 papers, 8.33%). There is one paper (1.19%) that does not fall into these categories, presenting an all-important discussion on the overall field.

Starting from the goals mentioned above, Figure 3.3b details the methodologies employed in these studies. Quantum Gates and Circuits are the most frequently used methodology, appearing in 30 papers (35.71%), indicating their foundational role in quantum computing. Quantum Cryptographic Protocols are used in 19 papers (22.62%), aligning with the high number of studies focusing on security. Methodologies involving Quantum Entanglement and States are featured in 10 papers (11.9%), while Quantum Communication and Networking Protocols are discussed in seven (7) papers (8.33%). Hybrid Architecture Proposals appear in five (5) papers (5.95%), and Quantum Optical Protocols in three (3) papers (3.57%). Quantum Random Number Generation and Quantum Neural Networks are mentioned in two (2) papers (2.4%) each. Three papers (3.57%) utilized discussion as their primary methodology to emphasize key aspects of their research, including identifying requirements, challenges, and potential solutions. In contrast, another three papers (3.57%) did not present any specific methodology in their implementation.

Figure 3.4 examines the technologies discussed in the papers. Classical Simulation is the most commonly used technology, with 28 papers (33.33%) utilizing it, emphasizing its role in modeling and testing quantum algorithms. Mathematical Proof/Model is the medium of proposals and proofs in 19 papers (22.62%), reflecting the theoretical underpinnings. Implementation of IBMQ³ and Dwave⁴ used Superconducting Qubits, presented in 11 papers (13.1%), showing significant interest in practical quantum hardware. Theoretical Analysis, Photonics, and IBMQ Simulation⁵ each appear in five (5) papers (5.95%), indicating a range of technological approaches. Less frequently mentioned technologies include Trapped-Ions, Quantum Entropy Chip, and ProjectQ Simulation, each appearing in one (1) paper (1.19%). Interestingly, eight papers (9.52%) did not present the use of any technologies being used.

Some of the important contributions in this field came from the development and optimization of QKD, which was implemented Quantum cryptographic protocols [146, 152, 153], Quantum Communication [154, 155], Quantum Circuits/Gates [156, 157], and Quantum optical protocols [157, 158]. Quantum Privacy Protocols [148, 159] and Quantum Secure Communication [62, 160, 161] also contributed substantially to the field. In recent times, Quantum Machine Learning [162, 93, 90] also made some innovative contributions to the field of Privacy and Security by Design in QSE. Discussions and Surveys took up exciting and important topics relevant to Quantum Software Engineering, and Privacy and Security and addressed the imperatives and requirements necessary for future research [67, 163, 164].

Majority of the studies used Classical Simulation and Mathematical Proof/-Models to attend to the implementation needs. Such classical simulations mostly used C, C++, Java, MATLAB, and Python as programming languages, OpenLDAP, OpenSSL, OpenFlow, etc., as network protocols, and mathematical tools like Huffman encoding, Monte Carlo Simulation, and Linear Feedback Shift Register(LFSR) based Toeplitz matrix.

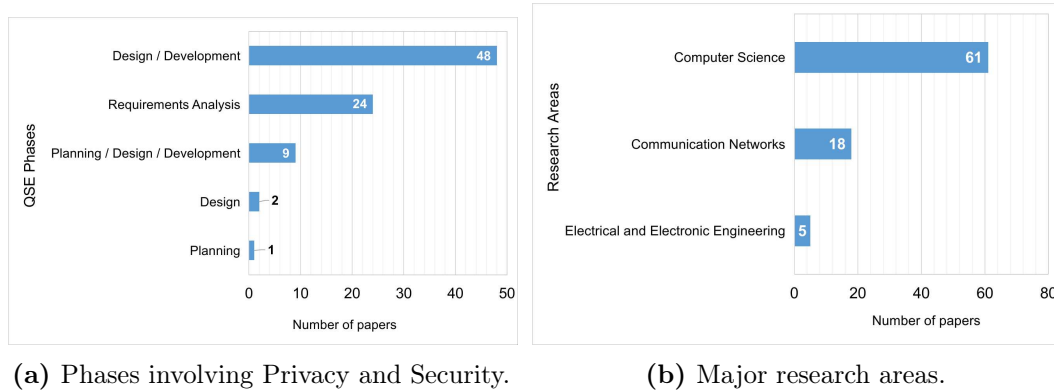


Figure 3.5: Graphs related to Research Questions RQ4 and RQ5

³<https://quantum.ibm.com/services/resources>

⁴<https://www.dwavesys.com/>

⁵<https://quantum-computing.ibm.com/>

RQ4: *In which phases of QSE are privacy and security mostly involved?*

As per the privacy and security by design guidelines by Ann Cavoukian in the "7 Foundational Principles"⁶ and OWASP⁷, the principal implementations of privacy and security by design impacts planning, design, development, and testing phase of any software development life cycle. Based on the systematic literature review (SLR), the analysis reveals distinct trends in the implementation of principles of privacy by design (PbD) and security by design (SbD) within various phases of Quantum Software Engineering (QSE). Figure:3.5a shows that 48 studies impact the Design and Development phases, representing 57.14% of the total. Meanwhile, 24 studies impact the Requirements Analysis phase, accounting for 28.57%. Additionally, nine (9) studies impact all Planning, Design, and Development phases together, making up 10.71%. Smaller contributions are noted in studies impacting only the Design and Planning phases separately, with two (2) and one (1) studies, respectively, together representing 3.57%. This data highlights that the majority of studies focus on integrating PbD and SbD principles during the design and development stages, emphasizing their critical role in the secure and privacy-aware development of quantum software.

⁶<https://www.ipc.on.ca/wp-content/uploads/Resources/7foundationalprinciples.pdf>

⁷<https://owasp.org/>

Table 3.7: Relevance Scoring for Privacy and Security

Relevance	Score	Basis of Classification
Extremely Relevant	2	1. Direct implementation of principles of privacy by design (PbD) and security by design (SbD) in quantum software engineering through specific algorithms or methodologies. 2. Development of methodologies to incorporate privacy and security in hybrid quantum-classical software development. 3. Proposal or design of comprehensive frameworks or methodologies that integrate PbD and SbD principles within QSE.
Relevant	1	1. Discussions on the principles of privacy and security by design, specifically within the context of quantum software engineering. 2. Research on ancillary topics within the field of quantum software engineering, such as quantum cryptography, quantum key management, and quantum teleportation, which support the broader objectives of PbD and SbD. 3. Studies focused on the development and enhancement of quantum physical principles related to security and/or privacy. 4. Implementation of quantum programming techniques that contribute to achieving partial PbD or SbD objectives within QSE.
Not Relevant	0	1. Does not discuss or in any way related to principles of PbD and SbD in QSE.

RQ5: *What are the main research areas that implemented the quantum privacy and security principles?*

In terms of research areas contributing to PbD and SbD in QSE, Figure 3.5b indicates that Computer Science is the dominant field, contributing 61 papers, which is 72.62% of the total. Communication Networks follows with 18 papers, representing 21.43%. Electrical and Electronic Engineering has the least contribution with five (5) papers, making up 5.95%. This distribution suggests that computer science is the leading discipline driving research in secure and privacy-preserving quantum software engineering, reflecting its central role in the development and theoretical foundations of quantum computing technolo-

gies. The results agree with the fact that the computer science community has seen an exponential thrust in quantum technology research.

RQ6: *What are the principle impacts of the studies on Privacy by Design (PbD)?* To answer this question better, we label the relevance of a study based on the principles with extremely relevant (2), relevant (1), and not relevant (0) (Table 3.7). We label each study according to its degree of alignment with the principles of PbD (Table 3.10). Figure 3.6 provides an analysis of the relevance of various studies to the principles of privacy by design (PbD) in quantum software engineering. Out of the total studies assessed, 64 papers (76%) were classified as Relevant, indicating they have substantial content or methodologies pertinent to PbD. Another 12 papers (14%) were deemed Extremely Relevant, highlighting their direct and significant contributions to implementing PbD principles in quantum software engineering. Meanwhile, 8 papers (10%) were categorized as Not Relevant, showing little to no connection to PbD. This data underscores that a substantial majority (90%) of the studies reviewed are relevant to the principles of privacy by design in the context of quantum software engineering.

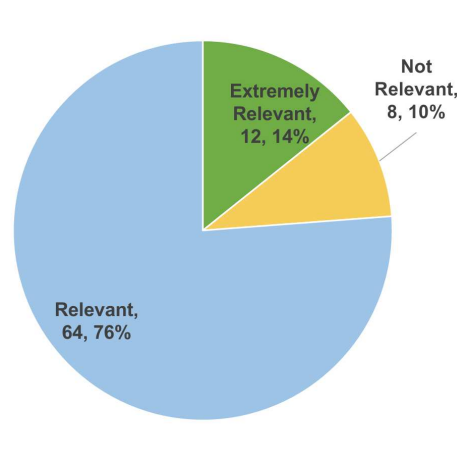


Figure 3.6: Relevance to Principles of Privacy by Design in QSE

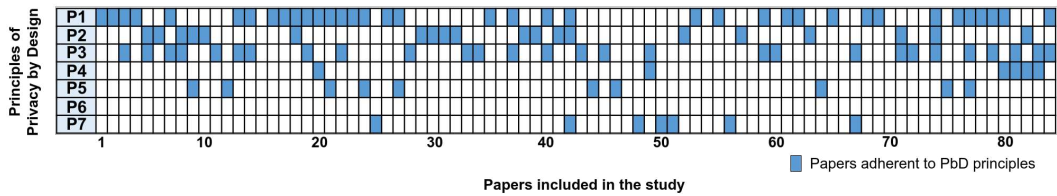


Figure 3.7: Adherence to Principles of Privacy by Design in QSE

Figure 3.7 presents a heatmap that visualizes the adherence of different papers to the seven principles of privacy by design (PbD), labeled as P1 to P7. Each blue square represents a paper that adheres to a specific PbD principle, while white squares indicate non-adherence. The heatmap reveals that P1 has the strongest adherence, with a significant number of blue squares indicating that many papers follow this principle. P2 also shows notable adherence, though

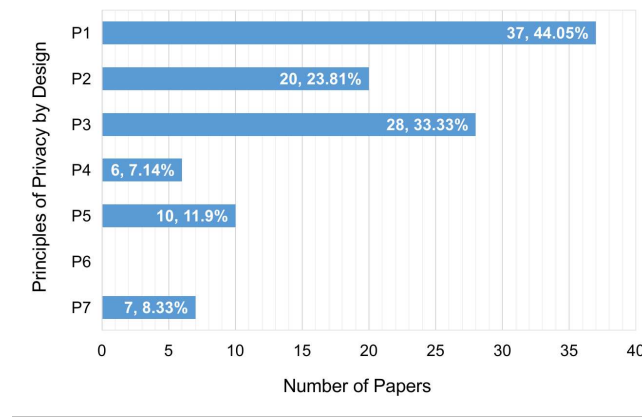


Figure 3.8: Distribution of Papers in Privacy by Design

slightly less than P1, with several papers adhering to this principle. The adherence to P3 is moderate, with a mix of blue and white squares spread across the chart, suggesting varied adherence among papers. P4 has sporadic adherence, with a few clusters of blue squares indicating that some papers follow this principle while many do not. P5 shows less adherence compared to P1 and P2, with occasional blue squares scattered across the chart. The adherence to P6 is minimal, with only a few blue squares, indicating that a lower number of papers adhere to this principle. Similarly, P7 shows limited adherence, with few papers following this principle. Overall, the heatmap indicates that P1 and P2 have the highest adherence among the PbD principles, while P6 and P7 have the least, providing a clear visual representation of which PbD principles are most commonly implemented in the literature.

A substantial portion of the papers (47) adhere to only one principle, showing a selective implementation of PbD principles. Additionally, 26 papers demonstrate adherence to two principles, with varied combinations such as 11 papers adhering to principles P1 and P3 predominantly and others following combinations like P1-P2, P2-P5, P2-P4, P1-P4, P1-P5, P2-P3, and P3-P4. Furthermore, 3 papers adhere to three principles, each with combinations like P1-P2-P7, P1-P2-P3, and P1-P3-P5. Following the results of Figure 3.6, eight papers being Not Relevant to PbD do not adhere to any principles.

Figure 3.8 shows the individual distribution of adherence among the reviewed papers. Each principle of Privacy by Design (P1 to P7) is represented, with P1 accounting for the highest share at 44.05% (37 papers), followed by P3 at 33.33% (28 papers) and P2 at 23.81%, with the others having lesser representation.

RQ7: *What are the principle impacts of the studies on Security by Design (SbD)?*

We grade the relevance of a study to the principles of SbD based on extremely relevant, relevant, and not relevant by scoring every study with scores two (2), one (1), and zero (0), respectively (Table 3.7). We label each study according to its degree of alignment with the principles of SbD to find out the overall status of the study group. We refer to the ten principles of Security by Design

proposed by OWASP (Table 3.11) to address this research question.

Figure 3.9 shows the relevance of papers to the principles of Security by Design (SbD). Out of the total papers analyzed, 67 papers, which constitute 80%, are categorized as relevant to SbD. A smaller portion, 13 papers, making up 15%, are considered extremely relevant, indicating a strong alignment with SbD principles. In contrast, only 4 papers, accounting for 5%, are classified as not relevant. This distribution indicates that a significant majority of the papers either directly address or are closely related to the principles of Security by Design, underscoring the importance of SbD in the field.

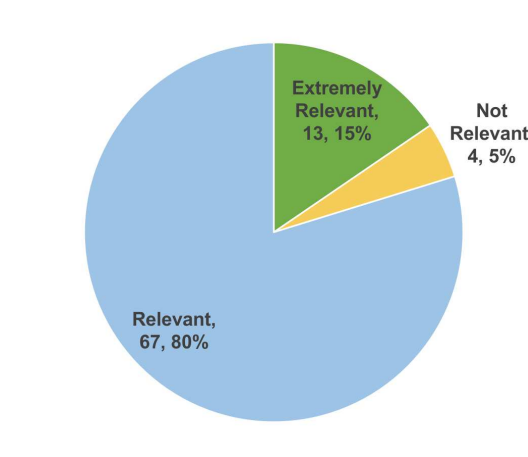


Figure 3.9: Security by Design in QSE

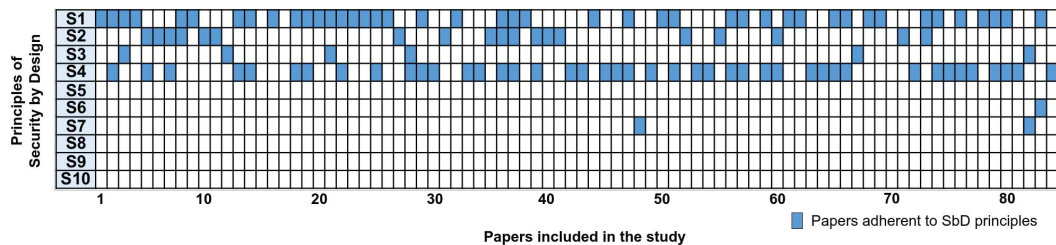


Figure 3.10: Adherence to Principles of Security by Design in QSE

Figure 3.10 is a heatmap that shows how different research papers adhere to ten principles of Security by Design (S1 to S10). Each column represents a paper, with blue squares indicating adherence to a principle and white squares indicating non-adherence. The heatmap reveals that S1 and S4 have the strongest adherence, with a significant number of blue squares indicating that many papers follow these principles. S2 also shows notable adherence, though slightly less than S1 and S4, with several papers adhering to this principle. The adherence to S3 is moderate, with a mix of blue and white squares spread across the chart, suggesting varied adherence among papers. S6 and S7 have sporadic adherence, with a few blue squares indicating that few papers follow these principles while many do not. Overall, the heatmap indicates that S1 and S4 have the highest adherence among the SbD principles, while S6 and S7

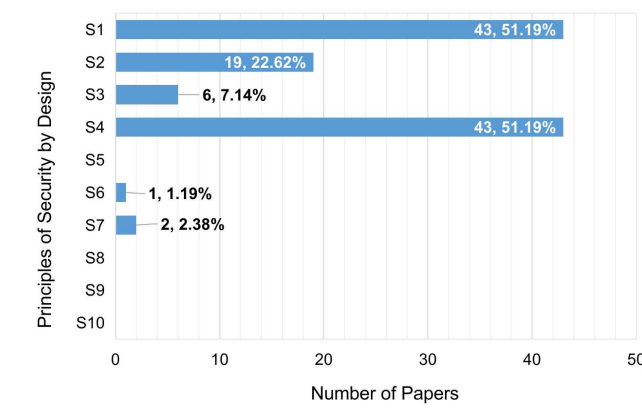


Figure 3.11: Distribution of Papers in Security by Design

have the least, providing a clear visual representation of which SbD principles are most commonly implemented in the literature.

Four papers do not adhere to any of the principles at all, corroborating the results presented in Figure 3.9. On the other hand, 48 papers follow one principle, with 17 papers adhering to S1 and S4 each, 11 to S2, two to S3, and one paper to S7. In terms of papers adhering to two principles, 30 papers fit this category. Among these, 19 papers adhere to both S1 and S4, 4 papers adhere to S2 and S4, and 2 papers each to S1-S2 and S1-S3. Additionally, three (3) papers adhere to combinations like S1-S6, S3-S4, and S3-S7. There are also 2 papers that follow three principles, specifically S1, S2, and S4. This visual representation effectively highlights the varying degrees of adherence to Security by Design principles across the research papers, giving a clear picture of which principles are most commonly followed.

Figure:3.11 reflects the adherence of each individual paper to the SbD principles. Each of the principles are shown as S1 to S10, with S1 and S5 being the most discussed, each representing 51.19% (43 papers). S2 accounts for 22.62% (19 papers), while the remaining principles have minimal representation.

Quality Assessment of Studies

Table 3.8 categorizes the papers reviewed in a systematic literature review (SLR) based on their quality, which is determined by their scores on a questionnaire (Table 3.6 [143, 144]). The papers were marked with identifiers **S01**, **S02**, **S03**, and so on, for easier handling. The quality levels are divided into Low, Medium, and High. There were three papers in the low-quality paper group that scored between 0-3 on the questionnaire. These papers, although they mentioned research goals and aims in a couple of scenarios, did not discuss the research design, implementation, data handling, and findings and, thus, were ultimately deemed not relevant and excluded from the SLR during the full-text review. Since these papers were removed from the study, they are not shown in the quality assessment results. The medium quality category includes 57 papers that scored between 4-7, reflecting a moderate level of quality. In the high-quality category, 27 papers scored between 8-11, indicating strong

alignment with the research criteria. The table provides a clear overview of the distribution of papers across different quality levels and highlights which papers were relevant enough to be included in the final analysis.

The results of the quality assessment in Table 3.9 presents a detailed overview of the quality assessment conducted on 84 papers, covering key criteria within the categories of Reporting, Rigor, Credibility, and Relevance. Reporting reveals that nearly all papers clearly outlined their research goals and aims(84). However, there was a slight dip in how well the papers contextualized their research, with only 58 papers effectively presenting a clear contextualization of their research.

The assessment of Rigor indicates a mixed performance. While 67 papers effectively implemented a Research Design, there was a significant drop in the use of Sampling (11 papers) and Control Groups (0 papers). Despite this, Data Collection (44 papers) and Data Analysis (46 papers) were handled more thoroughly, reflecting a reasonable level of methodological rigor. Reflexivity was not demonstrated in any of the papers, signaling an area for potential growth. However, the findings were generally well-reported, with all papers (84) delivering strong and credible results. Almost all the papers(83) were recognized for their significant contributions to the area of research.

Overall, the quality scores reflect that while the majority of papers had a strong foundation in their research aims and findings, there were areas, particularly in Rigor and Reflexivity, where future research could benefit from more focus. The final quality scores demonstrate a solid foundation but with clear opportunities for further improvement.

Table 3.8: Quality Assessment Scores

Quality	Questionnaire Score	Number of Papers	List of Papers
Low	0-3	3	Not Relevant (The papers were removed from the SLR during full-text review)
Medium	4-7	57	[S03][150], [S04][152], [S05][165], [S06][166], [S07][162], [S08][167], [S09][154], [S10][148], [S11][164], [S13][168], [S14][169], [S15][158], [S17][159], [S18][170], [S19][160], [S21][171], [S24][172], [S25][173], [S26][174], [S28][175], [S29][176], [S30][177], [S31][178], [S32][179], [S33][180], [S35][161], [S36][181], [S38][155], [S40][182], [S41][183], [S42][63], [S43][184], [S44][185], [S45][186], [S46][163], [S48][149], [S49][187], [S50][188], [S51][189], [S52][190], [S53][191], [S55][192], [S57][193], [S58][194], [S59][195], [S60][196], [S61][197], [S62][153], [S65][198], [S67][199], [S70][67], [S71][157], [S72][200], [S73][117], [S75][201], [S81][202], [S82][203]
High	8-11	27	[S01][146], [S02][204], [S12][62], [S16][205], [S20][206], [S22][207], [S23][208], [S27][147], [S34][151], [S37][209], [S39][210], [S47][211], [S54][212], [S56][93], [S63][213], [S64][214], [S66][215], [S68][156], [S69][216], [S74][99], [S76][90], [S77][217], [S78][218], [S79][96], [S80][219], [S83][220], [S84][221]

Table 3.9: Quality Assessment Results

Papers	Reporting			Rigor					Credibility		Relevance	Quality Score
	Research	Aim	Context	Research design	Sampling	Control Group	Data collection	Data analysis	Reflexivity	Findings	Value	
	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	
[S01][146]	1	1	1	1	0	0	1	1	0	1	1	8
[S02][204]	1	1	1	1	0	0	1	1	0	1	1	8
[S03][150]	1	1	1	1	0	0	0	0	0	1	1	6
[S04][152]	1	1	1	1	0	0	0	0	0	1	1	6
[S05][165]	1	1	1	0	0	0	0	0	0	1	1	5
[S06][166]	1	1	1	1	0	0	0	0	0	1	1	6
[S07][162]	1	1	0	1	0	0	1	1	0	1	1	7
[S08][167]	1	1	1	1	0	0	0	0	0	1	1	6
[S09][154]	1	1	1	1	0	0	0	1	0	1	1	7
[S10][148]	1	1	1	1	0	0	0	0	0	1	1	6
[S11][164]	1	1	1	0	0	0	0	0	0	1	1	5
[S12][62]	1	1	1	1	0	0	1	1	0	1	1	8
[S13][168]	1	1	1	1	0	0	0	0	0	1	1	6
[S14][169]	1	1	1	0	0	0	0	0	0	1	1	5
[S15][158]	1	1	1	1	0	0	1	0	0	1	1	7
[S16][205]	1	1	1	1	0	0	1	1	0	1	1	8
[S17][159]	1	1	1	1	0	0	0	0	0	1	1	6
[S18][170]	1	1	1	1	0	0	0	0	0	1	1	6
[S19][160]	1	1	1	1	0	0	0	0	0	1	0	5
[S20][206]	1	1	1	1	0	0	1	1	0	1	1	8
[S21][171]	1	1	1	1	0	0	0	0	0	1	1	6
[S22][207]	1	1	1	1	0	0	1	1	0	1	1	8
[S23][208]	1	1	1	1	0	0	1	1	0	1	1	8
[S24][172]	1	1	1	1	0	0	0	0	0	1	1	6
[S25][173]	1	1	1	1	0	0	0	0	0	1	1	6
[S26][174]	1	1	1	1	0	0	0	0	0	1	1	6
[S27][147]	1	1	1	1	0	0	1	1	0	1	1	8
[S28][175]	1	1	0	1	0	0	0	1	0	1	1	6
[S29][176]	1	1	0	0	1	0	0	0	0	1	1	5
[S30][177]	1	1	0	0	1	0	1	1	0	1	1	7
[S31][178]	1	1	0	1	0	0	1	1	0	1	1	7
[S32][179]	1	1	1	1	0	0	0	0	0	1	1	6
[S33][180]	1	1	1	1	0	0	0	0	0	1	1	6
[S34][151]	1	1	1	1	0	0	1	1	0	1	1	8

Table 3.9: Quality Assessment Results (continued)

Papers	Reporting			Rigor					Credibility		Relevance	Quality Score
	Research	Aim	Context	Research design	Sampling	Control Group	Data collection	Data analysis	Reflexivity	Findings	Value	
	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	
[S35][161]	1	1	0	0	1	0	0	0	0	1	1	5
[S36][181]	1	1	0	1	1	0	0	0	0	1	1	6
[S37][209]	1	1	1	1	0	0	1	1	0	1	1	8
[S38][155]	1	1	1	1	0	0	0	1	0	1	1	7
[S39][210]	1	1	1	1	0	0	1	1	0	1	1	8
[S40][182]	1	1	1	0	0	0	0	0	0	1	1	5
[S41][183]	1	1	0	1	1	0	0	1	0	1	1	7
[S42][63]	1	1	1	0	0	0	1	1	0	1	1	7
[S43][184]	1	1	0	0	1	0	1	1	0	1	1	7
[S44][185]	1	1	1	1	0	0	1	0	0	1	1	7
[S45][186]	1	1	0	1	0	0	1	1	0	1	1	7
[S46][163]	1	1	0	1	0	0	0	0	0	1	1	5
[S47][211]	1	1	0	1	1	0	1	1	0	1	1	8
[S48][149]	1	1	0	1	0	0	1	1	0	1	1	7
[S49][187]	1	1	0	1	0	0	1	1	0	1	1	7
[S50][188]	1	1	0	0	1	0	1	1	0	1	1	7
[S51][189]	1	1	0	1	0	0	1	1	0	1	1	7
[S52][190]	1	1	1	1	0	0	0	0	0	1	1	6
[S53][191]	1	1	0	1	0	0	0	0	0	1	1	5
[S54][212]	1	1	1	1	0	0	1	1	0	1	1	8
[S55][192]	1	1	1	0	0	0	0	0	0	1	1	5
[S56][93]	1	1	1	1	1	0	1	1	0	1	1	9
[S57][193]	1	1	0	1	0	0	1	1	0	1	1	7
[S58][194]	1	1	0	1	0	0	0	0	0	1	1	5
[S59][195]	1	1	0	1	0	0	1	1	0	1	1	7
[S60][196]	1	1	0	0	0	0	1	1	0	1	1	6
[S61][197]	1	1	0	0	0	0	1	1	0	1	1	6
[S62][153]	1	1	0	0	1	0	1	1	0	1	1	7
[S63][213]	1	1	1	1	0	0	1	1	0	1	1	8
[S64][214]	1	1	1	1	0	0	1	1	0	1	1	8
[S65][198]	1	1	1	1	0	0	0	0	0	1	1	6
[S66][215]	1	1	1	1	0	0	1	1	0	1	1	8
[S67][199]	1	1	1	0	0	0	0	0	0	1	1	5
[S68][156]	1	1	1	1	0	0	1	1	0	1	1	8

Table 3.9: Quality Assessment Results (continued)

Papers	Reporting			Rigor					Credibility		Relevance	Quality Score
	Research	Aim	Context	Research design	Sampling	Control Group	Data collection	Data analysis	Reflexivity	Findings	Value	
	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	
[S69][216]	1	1	1	1	0	0	1	1	0	1	1	8
[S70][67]	1	1	0	1	0	0	0	0	0	1	1	5
[S71][157]	1	1	0	0	1	0	0	0	0	1	1	5
[S72][200]	1	1	0	1	0	0	0	0	0	1	1	5
[S73][117]	1	1	1	0	0	0	0	0	0	1	1	5
[S74][99]	1	1	1	1	0	0	1	1	0	1	1	8
[S75][201]	1	1	1	1	0	0	0	0	0	1	1	6
[S76][90]	1	1	1	1	0	0	1	1	0	1	1	8
[S77][217]	1	1	1	1	0	0	1	1	0	1	1	8
[S78][218]	1	1	1	1	0	0	1	1	0	1	1	8
[S79][96]	1	1	1	1	0	0	1	1	0	1	1	8
[S80][219]	1	1	1	1	0	0	1	1	0	1	1	8
[S81][202]	1	1	1	1	0	0	0	0	0	1	1	6
[S82][203]	1	1	1	1	0	0	0	0	0	1	1	6
[S83][220]	1	1	1	1	0	0	1	1	0	1	1	8
[S84][221]	1	1	1	1	0	0	1	1	0	1	1	8
	84	84	58	67	11	0	44	46	0	84	83	

3.1.4 Discussion

In the following section, we discuss the insights gained from our the results and data extracted from our research and then present the findings. We analyze the results in light of Quantum Software Engineering practice and describe the outcomes of the analysis to understand the existing challenges and solutions in incorporating privacy by Design and Security by Design in QSE.

RQ1: *In which venue are papers published?*

The publication trends for Privacy by Design (PbD) and Security by Design (SbD) in quantum software engineering indicate a growing interest and increasing research activity in this field over time. Starting at 7.14% publications in 2017, there was a slight dip in 2018 (3.57%), but since then, the numbers have risen, with notable spikes in 2019 (14.29%) and 2023 (26.19%). This pattern suggests that the field is gaining momentum, with more researchers and institutions focusing on integrating PbD and SbD principles in QSE. The significant jump in publications in 2023 indicates a particularly strong surge in interest and possibly an acceleration in advancements or recognition of the importance of these principles in the quantum computing domain.

The near-even distribution of publications between journals and conferences indicates that researchers disseminate their findings through a balanced mix of academic venues. Moreover, theoretical fields such as quantum information, quantum key management, quantum network security, and quantum secure communication have found a strong presence in conferences and journals, likely due to the well-established and readily available venues tailored to these areas. In contrast, emerging areas such as quantum software engineering, the application of security, and privacy in quantum-hybrid software development often find their way into workshops, especially those co-located with larger conferences, indicating that dedicated venues for these fields are still evolving. This underscores the need for more specialized or conducive platforms to support the dissemination of work in these newer areas. Looking at the current trend of publications in QSE, the number of studies is expected to increase at a breakneck pace in the coming years. The need for studies like this will help the QSE community identify the gaps in privacy and security and define and implement PbD and SbD efficiently in QSE, further driving the growth and maturity of this important field.

RQ2: *What are the most frequently applied empirical strategies?*

In addressing the most commonly used empirical strategies in the studies reviewed in the area of PbD and SbD in QSE, experiments are by far the most preferred strategy of choice, representing 61.90% of the total studies. This indicates that researchers in this field prioritize experimental approaches to validate theories, test hypotheses, and gather concrete empirical evidence. The high preference for experiments suggests the field's focus on practical, hands-on research to solidify concepts within quantum software engineering.

In contrast, other strategies, such as surveys (1.19%) and hybrid architectures (4.76%), are far less frequently employed. The limited use of surveys could

imply that the field prioritizes empirical, data-driven methods over surveys or architectural proposals, likely because quantum software engineering is still an emerging and complex domain where solid, practical evidence is considered more reliable or necessary for advancement.

Similarly, the low usage of frameworks (2.38%) and simulations (4.76%) could indicate that these methods, while valuable, are not yet as widely adopted within the research community for this specific area. The strong preference for data and model-driven experiments over these other strategies suggests that the field is still focused on foundational research, where practical validation is key, rather than developing or utilizing high-level models or theoretical frameworks.

Overall, the strong emphasis on experimentation, particularly with specific subjects such as quantum algorithms, security protocols, and privacy mechanisms, highlights the critical need for concrete evidence to support theoretical advancements in the field. The limited use of other strategies like surveys and frameworks points to potential areas for further investigation and refinement as quantum software engineering continues to mature. As the field progresses, these alternative strategies may become increasingly important, complementing the foundational work established through experimental research.

RQ3: *What quantum technologies/methodologies are mostly sought after to address privacy and security by design in QSE?*

The findings from the results section reveal significant advancements and research efforts focused on key objectives such as Quantum Key Distribution, Quantum Privacy Protocols, Quantum Secure Communication, Quantum Cybersecurity, Quantum Data Encryption and Steganography, Quantum Machine Learning and Quantum Network Security. These findings emphasize the important role of advancing quantum technologies to tackle the growing security and privacy challenges in the quantum domain.

- **Quantum Key Distribution:** A substantial 25% of the reviewed papers focus on Quantum Key Distribution (QKD), indicating its prominence as a well-established and extensively studied area within the field. QKD is highly popular in research due to several key factors. Primarily, it is one of the earliest and most well-researched areas of quantum cryptography, with foundational protocols like BB84[85] providing a strong basis for further study. Secondly, QKD can be implemented using existing photonic technologies, making it more accessible and practical for real-world deployment, even before fully functional quantum computers are available, for example, *ID3100 Clavis 2 QKD systems*⁸ and simulators. Additionally, QKD offers theoretically unbreakable encryption, making it particularly attractive for securing communications. The strong interest from industry and government sectors in protecting against future quantum attacks has further promoted research in QKD.

In one of the initial studies, D'Angelo et al.[146] proposed the enhance-

⁸<https://www.idquantique.com/quantum-safe-security/products/>

ment of practical QKD systems by implementing a methodology that involves a two-stage protocol utilizing a one-way quantum channel for photon transmission and a classical channel for data exchange. A real-time JAVA implementation of a QKD protocol was presented by Capraro et al.[152], specifically a modified Cascade algorithm for error correction and privacy amplification. The methodology involves performance testing in a controlled environment using dedicated computers and a MATLAB simulator to optimize key rates over a TCP-IP network. Dong et al.[174], Wang et al.[197], and Kaewpuang et al.[153] contributed to the advancement of quantum cryptographic protocols by introducing optimized algorithms for secure multicast services, enhancing key exchange security through trusted relays, and improving resource allocation in quantum key distribution networks for federated learning applications. All of them used classical simulation techniques and classical programming languages and tools like Java, MATLAB, Apache TOMCAT, and GAMS/CPLEX solver⁹.

Instead, Runse et al.[204], proposed the integration of QKD into existing optical networks. The study provides experimental validation, focusing on performance metrics such as throughput and quantum bit error rate (QBER). In an important contribution, Privacy amplification in continuous-variable quantum key distribution (CVQKD) systems is presented by Li et al.[208]. It introduces a memory-saving algorithm implemented through mathematical models/proofs of quantum optical technologies to enhance efficiency and speed while minimizing memory usage, thereby improving the performance of CVQKD systems. Other notable contributions underscore the ongoing efforts to refine and secure quantum cryptographic protocols, particularly in enhancing data privacy and security through advanced key distribution techniques[168] like B92[222] and device-independent (DIQKD) approaches[164].

Further research was carried out in enhancing quantum security using classical simulation and mathematical models by analyzing relay-assisted satellite QKD systems[154, 155] and introducing a 4C-oriented control architecture[201] for quantum cloud services, utilizing QKD, optical networks, and AES-256 encryption. This paper enhances QKD networks by introducing IC-QKD, which caches keys based on application importance and historical requests. In another work, Zhang et al.[156] proposed the enhancement of QKD networks by introducing IC-QKD, which caches keys based on application importance and historical requests. The contribution lies in using a key server architecture within a 5G/6G framework to efficiently distribute keys, improving quantum signal security through semantic analysis. Similar research implements the development of a high-performance FPGA-based privacy amplification scheme for QKD[216] and proposes a practical Quantum Anonymous Private Information Retrieval (QAPIR) protocol[157], both enhancing security and efficiency in quan-

⁹<https://www.gams.com/>

tum communication systems. Similarly, other related studies developed QKD protocols for security performance[207], authentication and liar detection[150], synchronization in QKD systems[158], intelligent security in power networks[214], and secure long-distance communication using polarization multiplexing[218], all leveraging classical simulation and mathematical modeling of advanced quantum technologies like quantum entanglement, single-photon modes, and photonic systems.

- ***Quantum Secure Communication, Quantum Privacy Protocols, & Quantum Cybersecurity:*** These technologies account for approximately 44.05% of the studies and, while crucial in the field of quantum technology, receive slightly less focus than quantum key distribution (QKD) due to the complexities of large-scale implementation. However, their significance is acknowledged, as reflected in the attention they receive, highlighting their essential role in securing quantum communication networks. For instance, Kirkland and Koppal[217] proposed a quantum-based privacy-preserving camera that balances privacy and utility through quantum-state imagery. It utilizes a double deep Q-learning control algorithm to anonymize images before measurement, demonstrating the potential of quantum computing in enhancing privacy in imaging technologies. In addition, Shi[149] proposed a quantum sealed-bid auction (QSA) protocol that ensures privacy and security without relying on a trusted third party. It employs quantum cryptography to encode private bids and utilizes secure quantum summation protocols, thereby guaranteeing confidentiality, fairness, and bidder anonymity while preventing unauthorized bid disclosure.

Having said that, there are some notable works that contribute significantly to the field of quantum privacy and security through various methodologies and applications. Tan et al.[148] developed a quantum private comparison scheme for big data using quantum mechanics and photonic systems, while Zhou et al.[159] analyzed differential privacy in quantum operations through theoretical frameworks. Chen et al.[170] introduced a secure quantum-blind dual-signature scheme, and Cai et al.[173] enhanced blockchain smart contracts against quantum attacks with quantum-blind signature schemes. Ji et al.[179] designed quantum private comparison protocols with multi-particle entangled states. Gao et al.[185] developed a quantum election protocol using quantum digital signatures, and Shi et al.[188] proposed quantum sealed-bid auction schemes ensuring privacy without an auctioneer. Shi[190] also presented a quantum protocol for sealed-bid auctions ensuring anonymity and fairness, and a quantum scheme for privacy in range queries within IoT using single photons. Yang et al.[202] proposed a secure quantum private query protocol against joint-measurement attacks, while Shi & Zhang[203] utilized Grover's search algorithm for privacy-preserving quantum sealed-bid auctions. The grand majority of them resorted to mathematical model-

ing/proofs, whereas the others used IBMQ simulation¹⁰ and real IBMQ backends¹¹.

While discussing about quantum gate-based approaches, Phalak et al.[219] contribute to enhancing quantum computing security by developing a Quantum Physically Unclonable Function (QuPUF), which validates quantum hardware identity using IBM quantum systems to ensure secure quantum program execution, and Akarshita et al.[220] focus on improving security and privacy in quantum circuit design by proposing a quantum obfuscation technique that employs dummy CNOT gates. This method, tested using IBM's quantum simulators, increases Total Variation Distance (TVD), effectively safeguarding circuits from reverse engineering. Both studies relied on superconducting qubits (IBMQ backends) to carry out the experiments.

Recent advancements in quantum security and technology direction also include Park et al.[151] developing a Side-Channel Resistant Quantum Random Number Generator (SCR-QRNG) to strengthen security against side-channel attacks. Saki et al.[184] enhanced noise resilience and security in quantum computing by implementing qubit isolation and reallocation techniques, tested on IBM systems, and further improved circuit security through split compilation strategies[187]. Yang et al.[191] proposed the Quantum Blockchain Identity Framework (QBIF) for secure identity management in decentralized applications, using quantum key distribution and entanglement. Saki et al.[193] identified vulnerabilities in trapped-ion quantum systems, focusing on shuttle-exploiting attacks. Witek et al.[194] introduced a Quantum True Random Number Generator (QTRNG) platform with a SHA256 accelerator for efficient, secure random bit generation. Paramasivam et al.[198] improved quantum circuit design for lightweight ciphers to resist quantum key recovery attacks, while Deshpande et al.[215] developed a quantum antivirus system to detect and mitigate malicious circuits. Gupta et al.[199] created a secure e-voting system by combining blockchain with Quantum Key Distribution (QKD) for data integrity and user authentication. Baldassarre et al.[117] introduced a vision model using Quantum Artificial Intelligence to enhance cybersecurity education in software engineering [99]. These studies collectively advance the security of quantum computing environments and applications. The papers banked on Quantum Entropy Chips[223], Superconducting Qubits¹², Trapped-Ions, Photonics, Mathematical Proofs/Models, and Classical Simulations.

Going forward, the following studies highlight innovative approaches to secure communication using Quantum Key Distribution (QKD) and other cryptographic techniques. Tanizawa et al.[62] proposes a secure communication network infrastructure utilizing Quantum Key Distribution (QKD)

¹⁰<https://quantum-computing.ibm.com/>

¹¹<https://quantum.ibm.com/services/resources>

¹²<https://quantum.ibm.com/services/resources>, <https://cloud.dwavesys.com/leap/>

technology to manage global keys effectively across multiple applications within a network, integrating an application directory and encryption functions via OpenSSL API. Armanuzzaman et al.[205] enhance secure and efficient data transmission by integrating Quantum Key Distribution (QKD), One-Time Pad (OTP) encryption, and Huffman encoding, leveraging quantum channels for key distribution and classical channels for data transmission, implemented using Java. In some other works, Li et al.[160] and Tysowski et al.[171] focus on developing scalable architectures for quantum key distribution, integrating both quantum and classical cryptographic methods. Xia et al.[180, 210] and Aguado et al.[182] also contribute to enhancing data security by implementing quantum key distribution in power systems and telecommunications, respectively. Qoussini et al.[176] and Choi et al.[163] advance secure communication protocols, with the former exploring quantum deterministic protocols and the latter addressing standardization for 5G networks. Meanwhile, Ngobeni et al.[161, 183] and Ralegankar et al.[192] apply quantum cryptography to specific fields, such as secure power distribution networks and UAV communications, underscoring the versatility and growing importance of quantum cryptography in various applications. Classical simulations were primarily used in the selected studies, along with theoretical analyses and discussions.

- ***Quantum Data Encryption & Steganography:*** Building on advancements in quantum cryptography, recent studies have proposed innovative approaches to enhance data encryption and steganography, particularly in sensitive applications like healthcare, accounting for approximately 11.9% of the total papers selected. Bassem et al.[213] proposed a highly sensitive cryptosystem for image encryption, integrating bit-level and pixel-level permutations with chaotic systems like the 3D logistic map, significantly advancing security in quantum software engineering. A secure image encryption mechanism was developed by Ahmad et al.[221] tailored for healthcare applications using controlled alternate quantum walks (CAQWs), enhancing medical image security in IoT environments by applying quantum cryptography principles.

A quantum-classical hybrid architecture was proposed by Rafael et al.[167] to accelerate the McEliece cryptosystem's decoding process using Grover's search algorithm, showcasing the synergy between quantum and classical processing. In another study, a big data protection framework combined quantum cryptography with neural networks, employing the BB84 protocol and Tree Parity Machines for secure key generation (Sharma et al.[169]).

Quantum image encryption has also seen progress with methods like DNA Controlled-Not operations for enhanced security[181] and controlled alternate quantum walks for robust encryption against attacks [209]. Janani et al.[63] proposed secure medical image transmission schemes using spatial transformations and quantum block-based encryption, while

Ismail et al.[186] proposed quantum spin-based encryption-based digital image security. Additionally, medical image steganography techniques using Particle Swarm Optimization and Quantum Walks were introduced to secure embedded data by Abd-El-Atty[189], and Lee et al.[212] optimized lightweight hash functions for quantum computing to enhance IoT communication security. The studies employed a variety of approaches, including photonics, classical and IBMQ simulations, mathematical modeling/proofs, and ProjectQ simulations¹³, to conduct their research and validate their quantum encryption and security methods.

Quantum data encryption and steganography slightly lead over Quantum Machine Learning (QML) and Quantum Network Security in research focus, likely because they *draw inspiration from well-established quantum cryptographic protocols* and Quantum Key Distribution (QKD), which is the most popular research goal. Additionally, the technology and *resources for implementing quantum data encryption and steganography are more accessible*, contributing to their popularity. Furthermore, during the review, it was noted that most studies implemented quantum gate or state-based approaches, which are new and innovative techniques that have significantly contributed to the advancement of these goals.

- **Quantum Machine Learning:** Comprising 9.5% of the studies in this SLR, the latest innovations in Quantum Computing have driven significant advancements, particularly in the areas of Quantum Artificial Intelligence and Quantum Machine Learning. Caivano et al.[93] propose an innovative optimization of intrusion detection for the CAN(in-vehicle controller area network) bus using quantum annealing, demonstrating the effectiveness of the QBoost quantum classification algorithm compared to classical SVM, utilizing D-Wave's quantum hardware¹⁴. Hatma et al.[96] enhance DGA detection with a hybrid quantum-classical deep learning model, using PennyLane¹⁵ for quantum circuits and Keras for deep learning, offering a comparative analysis of quantum versus classical model performance.

Recent research has made significant strides in applying quantum machine learning to enhance cybersecurity and smart infrastructure. Yong et al.[162] demonstrated the use of the Quantum Growing Hierarchical Self-Organized Map (QGHSOM) algorithm for intrusion detection, improving detection rates by integrating quantum principles into neural networks. Xue et al.[211] employed quantum machine learning via IBM Qiskit¹⁶ to detect electricity theft, achieving greater prediction accuracy. Al-Hawawreh et al.[195] created a privacy-aware quantum deep learning framework aimed at protecting IoMT(Internet of Medical Things) devices

¹³<https://github.com/ProjectQ-Framework/ProjectQ>

¹⁴<https://cloud.dwavesys.com/leap/>

¹⁵<https://pennylane.ai/>

¹⁶<https://www.ibm.com/quantum/qiskit>;<https://qiskit.org/>

from cyber threats, using differential privacy and quantum neural networks. Lakshmi et al.[196] developed quantum-based security measures against cyber attacks on smart grids, utilizing the Quirk quantum circuit simulator. Barletta et al.[99] introduced a Quantum as a Service (QaaS) model to bolster cybersecurity in smart cities, leveraging QBoost[47] and IBM Qiskit¹⁷ for data processing. Lastly, Barletta et al. (2022) optimized IoT security detection by applying the Qboost algorithm on a D-Wave quantum processor, resulting in faster detection and improved accuracy.

Therefore, the results show that QML, unlike Quantum Data Encryption & Steganography, remains nascent and has not yet gained widespread acceptance. We are currently in an era where traditional AI is just beginning to be socially and commercially accepted, thanks to state-of-the-art algorithms and techniques. Consequently, the integration of QML is still in its infancy, gradually gaining traction in the published literature. One resisting factor is that explainable AI has helped traditional AI gain acceptance, a feature that is not yet fully available in Quantum AI (QAI), likely making its application more challenging.

- **Quantum Network Security:** The integration of quantum technologies into network architectures, as demonstrated by 8.3% of the studies selected for this SLR, highlights significant advancements in enhancing security and efficiency in modern networking systems. The integration of quantum technologies into network architectures, as demonstrated by recent studies, marks significant advancements in enhancing security and efficiency in modern networking systems. Aguado et al.[206] contributed to the field by demonstrating a secure optical network architecture that integrates Quantum Key Distribution (QKD) with Network Function Virtualization (NFV) and Software-Defined Networking (SDN), showcasing the practical implementation of a time-shared QKD network design on real hardware. El-Latif et al.[147] advanced quantum-based security in 5G networks by developing efficient quantum hash functions, QWHF-1 and QWHF-2, using quantum walks to improve security and efficiency in data protection and information sharing.

Recent studies have explored enhancing network security through various quantum cryptographic methods. Rass et al.[166] utilized a game-theoretic model to evaluate quantum network security, offering strategies for optimal path selection. Aguado et al.[172] integrated QKD with virtual network functions to secure software-defined networks, while Diovu et al.[175] proposed a QKD-based framework for secure key exchange in cloud-based smart grids. Li et al.[177] and Wang et al.[178] concentrated on strengthening the security of power data transmission and optical networks through lightweight quantum cryptography and specialized protection schemes. These studies collectively push forward the use of quantum technologies in protecting modern communication

¹⁷<https://www.ibm.com/quantum/qiskit>; <https://qiskit.org/>

infrastructures.

So, quantum network security is also gaining attention, closely trailing QML in research focus. This can be attributed to the fact that quantum network security is a relatively new field, with most researchers relying on mathematical modeling or theoretical analyses due to the lack of accessible infrastructure, such as quantum networks, channels, or the quantum internet. As quantum technologies advance, we expect rapid growth in these fields.

RQ4: *In which phases of QSE are privacy and security mostly involved?*

The majority of research within Quantum Software Engineering (QSE) is concentrated in the Design/Development (57.14%) and Requirements Analysis (28.57%) phases. This suggests a strong emphasis on the practical aspects of constructing and refining quantum software systems, with considerable attention devoted to identifying and defining the necessary requirements for these systems. The relatively lower focus on Planning (1.19%) and isolated Design (2.38%) phases indicates that, while these early-stage activities are essential, the field is currently more driven by the imperative to develop and implement quantum technologies.

This study aligns with the findings of Piattini et al.[129], which emphasizes that heavily impacts software design, software construction, software testing, software maintenance, and software quality. This correlation reinforces the view that quantum technologies are increasingly integrated with classical computational methodologies to optimize processing, particularly during the design and development stages. Given that quantum computing significantly influences the modeling and implementation phases of software development, it is evident that both theoretical frameworks and practical applications emphasize these aspects of the life cycle. This emphasis reflects a broader trend within the field, where the practical application of quantum computing is prioritized to address immediate challenges and advance the implementation of quantum solutions to implement Privacy and Security in QSE.

In practical terms, this trend underscores the quantum computing community's focus on advancing beyond theoretical frameworks to the actual development and deployment of quantum solutions. This shift highlights the pressing need to address immediate challenges and refine existing models. As QSE continues to evolve, it is anticipated that there will be a more balanced emphasis across all phases, recognizing that early-stage planning and design are equally critical to the success of quantum software initiatives.

RQ5: *What are the main research areas that implemented the quantum privacy and security principles?*

The majority of research within QSE is concentrated in the field of Computer Science. This strong focus highlights that the technical aspects of quantum computing, particularly those related to software development, algorithms, and computational methods, are the main areas of investigation. Many of these studies, which focus on QKD and its various implementations, further

solidify the integration of fields like the IoT, quantum cryptography, data transmission schemes, attack detection, and quantum blockchain within the broader realm of Computer Science applied research. In contrast, the fields of Communication Networks and Electrical and Electronic Engineering are less represented. Although these areas are currently secondary to the primary computational research, they represent important avenues for future exploration. As quantum technologies continue to evolve, it is anticipated that research will increasingly extend into these interdisciplinary fields, leading to broader applications and innovations. This trend is significant, as the integration of quantum computing with communication networks and electronic systems has the potential to drive major advancements in these areas, influencing the direction of future technological development.

Therefore, the prevalence of studies in Computer Science research within QSE arises from its *fundamental role in software development, its interdisciplinary scope, and the increasing accessibility of quantum computing resources*. Computer Science's connection with critical areas like cryptography and algorithms makes it an ideal field for exploring quantum technologies. Furthermore, the emphasis on practical applications and industry demand drives more research within the domain.

RQ6: *What are the principle impacts of the studies on Privacy by Design (PbD)?*

The analysis reveals that the majority (about 76%) of the reviewed papers find Privacy by Design (PbD) relevant, with an additional 14% considering it extremely relevant in the context of QSE. This reflects a strong acknowledgment within the research community of the importance of incorporating privacy considerations from the beginning in quantum software development. The significant relevance of PbD highlights the increasing need to address privacy issues proactively, especially as quantum computing brings new challenges to data security and privacy. The rest of the studies that did not find PbD relevant indicate specific contexts where privacy is less of a focus, but overall, the trend emphasizes the critical role of privacy in the design phase of quantum applications.

While Figure 3.7 illustrates the adherence of individual papers to the principles of PbD, Figure 3.8 and Table 3.10 provide insights into how different principles of PbD are reflected across the reviewed QSE papers, highlighting the varying levels of emphasis on specific privacy aspects within these studies supported by the analysis below:

- **P1:** *Proactive not Reactive; Preventive not Remedial (44.05%)* - This principle shows the highest adherence, indicating a strong focus on incorporating proactive privacy measures. The research community appears to prioritize building privacy protections into the software from the outset rather than relying on remedial actions after issues arise.
- **P2:** *Privacy as the Default Setting (23.81%)* - This principle is observed in nearly a quarter of the papers, highlighting the importance of maintaining

privacy as the default state without requiring user intervention. This reflects the community's understanding of the significance of default privacy protections in quantum software.

- **P3: Privacy Embedded into Design (33.33%)** - A substantial proportion of the papers integrate privacy considerations into the design phase, showing that embedding privacy into the core architecture of quantum software is a significant focus for many researchers.
- **P4: Full Functionality – Positive-Sum, not Zero-Sum (7.14%)** - The low adherence to this principle suggests that achieving a balance between privacy and other functional aspects of the software remains a challenge and is less commonly addressed in current research.
- **P5: End-to-End Security – Full Lifecycle Protection (11.90%)** - A moderate level of adherence indicates that some papers emphasize maintaining security throughout the entire lifecycle of the software, from creation to decommissioning.
- **P6: Visibility and Transparency – Keep it Open (0.00%)** - No papers adhere to this principle, indicating that visibility and transparency in privacy practices are either not prioritized or are difficult to implement in the context of QSE, or have been left as an imperative for the adjoining classical system in case of quantum-hybrid systems.
- **P7: Respect for User Privacy – Keep it User-Centric (8.33%)** - The relatively low adherence suggests that while user-centric privacy is considered in some papers, it is not a major focus in the current body of research.

RQ7: What are the principle impacts of the studies on Security by Design (SbD)?

SbD is considered relevant in most of the studies and this indicates a strong focus on the importance of security in QSE. As quantum computing introduces significant new challenges to traditional security frameworks, the high relevance of SbD in the literature underscores the necessity of integrating security measures throughout the software development process. The small percentage of studies that did not find SbD relevant reflects specific research areas where security concerns are either secondary or addressed through other means. Nevertheless, the strong consensus on the importance of SbD highlights its role as a key principle in developing secure quantum software systems.

The distribution of focus across the principles of SbD in QSE reveals key insights into the priorities and potential gaps within the research community. Figure:3.10 shows how individual papers comply with the principles of Security by Design (SbD), while Figure:3.11 and Table:3.11 detail how these principles are reflected across the QSE papers we reviewed. This breakdown highlights the varying emphasis on specific security aspects within the studies, which we further explore in the following analysis.

- **S1: Minimize Attack Surface Area (51.19%)** - This approach is central

Table 3.10: Compliance to Privacy by Design Principles

Privacy by Design Principles	Papers Compliant to the Principles
P1	[S01][146], [S02][204], [S03][150], [S04][152], [S07][162], [S13][168], [S14][169], [S16][205], [S17][159], [S18][170], [S19][160], [S20][206], [S21][171], [S22][207], [S23][208], [S24][172], [S26][174], [S27][147], [S35][161], [S37][209], [S40][182], [S42][63], [S53][191], [S55][192], [S59][195], [S61][197], [S62][153], [S65][198], [S68][156], [S69][216], [S74][99], [S76][90], [S77][217], [S78][218], [S79][96], [S80][219], [S84][221]
P2	[S05][165], [S06][166], [S08][167], [S09][154], [S10][148], [S18][170], [S29][176], [S30][177], [S31][178], [S32][179], [S38][155], [S39][210], [S41][183], [S42][63], [S52][190], [S57][193], [S63][213], [S71][157], [S74][99], [S82][203]
P3	[S03][150], [S05][165], [S07][162], [S08][167], [S11][164], [S13][168], [S14][169], [S19][160], [S22][207], [S28][175], [S33][180], [S34][151], [S37][209], [S40][182], [S43][184], [S45][186], [S49][187], [S59][195], [S60][196], [S67][199], [S71][157], [S72][200], [S74][99], [S77][217], [S79][96], [S81][202], [S83][220], [S84][221]
P4	[S20][206], [S49][187], [S80][219], [S81][202], [S82][203], [S83][220]
P5	[S09][154], [S12][62], [S21][171], [S24][172], [S27][147], [S44][185], [S46][163], [S64][214], [S75][201], [S77][217]
P6	-
P7	[S25][173], [S42][63], [S48][149], [S50][188], [S51][189], [S56][93], [S67][199]

to building resilient quantum systems that are difficult to breach. The high emphasis on minimizing the attack surface suggests that researchers are particularly focused on reducing the number of potential entry points for attackers. The large percentage indicates that this is a primary concern within the research community, reflecting a proactive stance toward securing quantum software environments.

- **S2: Establish Secure Defaults (22.62%)** - Establishing secure defaults is moderately addressed, indicating that while researchers recognize its importance, it may not be as prioritized as other defense mechanisms. This could imply a focus on more direct security strategies, with the assumption that defaults will follow best practices, though this is an area that could benefit from increased attention.
- **S3: Principle of Least Privilege (7.14%)** - The relatively low focus on the principle of least privilege suggests that researchers might not be giving

adequate attention to access control measures. This could indicate a gap in ensuring that users and processes have only the minimum necessary access. On the other hand, in the case of quantum-hybrid systems, this can be taken care of by classical computational resources.

- **S4: Principle of Defense in Depth (51.19%)** - Like the focus on minimizing the attack surface, the strong emphasis on defense in depth indicates a strategic approach to security, where multiple layers of defense are implemented to protect quantum systems. This principle clearly stands as a key focus in the research community's efforts to build strong and resilient systems, showing a careful and comprehensive approach to security.
- **S5: Fail Securely (0.00%)** - The complete lack of focus on failing securely is concerning, as it suggests that researchers may be overlooking the importance of secure failure mechanisms. Overlooking on how systems behave under failure conditions, quantum systems could be left vulnerable during unexpected or catastrophic events. However, this responsibility might also be seen as part of the broader scope of classical software engineering, which shares the burden of ensuring system resilience in quantum-hybrid systems.
- **S6: Do Not Trust Services (1.19%)** - The minimal focus on not trusting services indicates that third-party risks and dependencies may not be a major concern for most researchers. This could be a potential oversight, as reliance on external services like quantum cloud APIs without sufficient trust measures could introduce vulnerabilities.
- **S7: Separation of Duties (2.38%)** - The low emphasis on separation of duties suggests that internal security processes might be undervalued in current research. This could leave quantum systems vulnerable to insider threats or conflicts of interest, highlighting a potential area for future research to ensure that security controls are appropriately distributed across roles.
- **S8: Avoid Security by Obscurity (0.00%)** - The complete absence of focus on avoiding security by obscurity could imply an implicit assumption that transparency is inherently valued in quantum research. However, without explicit attention to this principle, there is a risk that some solutions may inadvertently rely on obscurity rather than robust security mechanisms.
- **S9: Keep Security Simple (0.00%)** - The lack of attention to keeping security simple could reflect the complexity of quantum systems, where researchers might prioritize advanced defense strategies or classical defense measures over simplicity since most quantum systems are housed beyond the classical computational wall of defense. However, neglecting simplicity can lead to increased complexity, which in turn can introduce vulnerabilities and make security harder to manage effectively.
- **S10: Fix Security Issues Correctly (0.00%)** - The absence of focus on fixing security issues correctly suggests that the research community

may not be placing enough emphasis on thorough, lasting solutions to security vulnerabilities. This oversight could result in persistent security flaws that are inadequately addressed, emphasizing the need for more attention to this principle. Another reason for this might be because of the fact that QSE is still immature and the literature does not yet support standardized security issue fix guidelines.

Table 3.11: Security by Design Principles and Compliance

Security by Design Principles	Papers Relevant to the Principles
S1	[S01][146], [S02][204], [S03][150], [S04][152], [S08][167], [S09][154], [S13][168], [S14][169], [S16][205], [S18][170], [S19][160], [S20][206], [S21][171], [S22][207], [S23][208], [S24][172], [S25][173], [S26][174], [S29][176], [S32][179], [S36][181], [S37][209], [S38][155], [S44][185], [S47][211], [S50][188], [S51][189], [S56][93], [S57][193], [S59][195], [S61][197], [S62][153], [S65][198], [S66][215], [S68][156], [S69][216], [S73][117], [S74][99], [S76][90], [S78][218], [S79][96], [S80][219], [S83][220]
S2	[S05][165], [S06][166], [S07][162], [S08][167], [S10][148], [S11][164], [S27][147], [S31][178], [S35][161], [S36][181], [S37][209], [S39][210], [S40][182], [S41][183], [S52][190], [S55][192], [S60][196], [S71][157], [S73][117]
S3	[S03][150], [S12][62], [S21][171], [S28][175], [S67][199], [S82][203]
S4	[S02][204], [S05][165], [S07][162], [S13][168], [S14][169], [S18][170], [S19][160], [S22][207], [S25][173], [S28][175], [S29][176], [S30][177], [S33][180], [S34][151], [S36][181], [S37][209], [S39][210], [S42][63], [S43][184], [S45][186], [S46][163], [S47][211], [S49][187], [S51][189], [S53][191], [S54][212], [S56][93], [S57][193], [S59][195], [S60][196], [S63][213], [S64][214], [S65][198], [S66][215], [S72][200], [S74][99], [S75][201], [S76][90], [S77][217], [S79][96], [S80][219], [S81][202], [S84][221]
S5	-
S6	[S83][220]
S7	[S48][149], [S82][203]
S8	-
S9	-
S10	-

The analysis demonstrates substantial compliance with the majority of privacy and security by design principles, reflecting a commitment to these essential aspects. However, several gaps in adherence have been identified, indicating areas that require further improvement. While many of the reviewed papers align well with principles such as privacy by default and data minimization,

there is less compliance with certain security principles, particularly those related to maintaining data integrity, availability, and resistance to potential attacks.

To develop more resilient quantum-safe environments, it is crucial that these gaps be addressed. However, as noted by Piattini et al.[129], not all phases of QSE are directly impacted by quantum computing. This suggests that some of the identified gaps in adherence to privacy and security principles may not necessarily require quantum technologies for resolution. In many cases, classical technologies can effectively address these issues, particularly in areas where quantum impacts are minimal or absent. All the more so because the next step in advancing quantum-safe environments will involve the seamless integration of quantum computers with classical systems.

Future research and development in quantum systems should aim for comprehensive adherence to all relevant principles, whether through quantum or classical means. Strengthening compliance in underrepresented areas will be crucial for ensuring that quantum technologies not only meet current security and privacy standards but are also robust enough to protect against future threats. As the field of quantum technology evolves, integrating these principles into both the design and implementation stages—utilizing a blend of quantum and classical technologies—will be critical for safeguarding against vulnerabilities.

By recognizing the distinct roles of quantum and classical technologies in different phases of QSE, we can more strategically address these gaps. This approach will play a significant role in building a secure, private, and resilient quantum ecosystem, ensuring its long-term viability and trustworthiness for evolving challenges.

This study, anchored in the research questions outlined in Table 3.1, deep dives into the existing literature to discover the challenges and gaps in applying Privacy by Design (PbD) and Security by Design (SbD) principles in Quantum Software Engineering (QSE). Through this investigation, we gain a clearer picture of the current landscape in this area of research and its implications for real-world practice. Based on this thorough analysis, we can identify several critical action points that will be crucial for developing effective frameworks or methodologies to better integrate PbD and SbD into QSE. Please find below the list of curated action points:

1. **Integrate Privacy and Security by Design as a Core Requirement in QSE:** Privacy by Design (PbD) and Security by Design (SbD) principles must be integrated as foundational elements throughout the Quantum Software Engineering (QSE) lifecycle. This approach ensures that privacy and security considerations are not treated as afterthoughts but are embedded from the initial design phase through to deployment and maintenance.
2. **Adapt PbD and SbD Principles for Quantum Contexts:** PbD

and SbD principles should be systematically adapted and translated into Quantum Privacy by Design (Q-PbD) and Quantum Security by Design (Q-SbD) frameworks. This involves tailoring traditional principles to the unique challenges posed by quantum computing and embedding them into quantum software development processes, especially in areas where quantum computing significantly impacts the software lifecycle.

3. **Systematic Identification and Mitigation of Bugs and Vulnerabilities:** Implement a comprehensive approach to identify and mitigate bugs and vulnerabilities that could lead to privacy and security issues at every stage of Quantum Software Engineering. Develop best practices and standardized protocols to ensure that these issues are effectively addressed, especially in quantum-hybrid systems where classical and quantum components interact.
4. **Innovate Quantum-Hybrid Software Engineering Practices for Privacy and Security:** Develop innovative methodologies and frameworks that specifically tackle the privacy and security challenges unique to quantum-hybrid systems. These new approaches should facilitate the seamless incorporation of PbD and SbD principles into the quantum software development process, ensuring that these principles are not only integrated effectively but also remain flexible and scalable as quantum technology continues to advance.
5. **Promote the Exploration of Advanced Quantum Techniques:** Promote the exploration and application of advanced quantum techniques to strengthen the integration of privacy and security principles in quantum software. Leverage quantum-specific properties to improve data protection and system security, aiming to build quantum software that is robust against both current and future threats.
6. **Address Research Gaps with Comprehensive Strategies:** Identify and prioritize existing research gaps in the implementation of PbD and SbD principles within QSE. Develop focused research strategies to address these gaps, ensuring that quantum software systems are designed to be secure, scalable, and capable of effectively integrating privacy and security considerations as quantum technologies are increasingly adopted.

While the proposed action points are designed to advance Quantum Software Engineering (QSE), there are several potential risks associated with their implementation. These include potential resistance to change, technical difficulties in adapting privacy and security principles to quantum contexts, increased complexity in managing bugs and vulnerabilities, significant resource demands for developing new methodologies, and challenges in adopting advanced quantum techniques. Effectively managing these risks will be essential for the successful integration of these strategies.

1. **Resistance to Change:** Integrating PbD and SbD as foundational elements in QSE could encounter resistance from teams used to traditional

methods. This may slow down adoption, leaving systems vulnerable during the transition period.

2. **Technical Difficulties in Adapting PbD and SbD for Quantum Contexts:** Tailoring these principles for quantum computing could present unforeseen technical challenges, such as difficulties in adapting classical security measures to quantum environments. These obstacles might delay the development of Quantum Privacy by Design (Q-PbD) and Quantum Security by Design (Q-SbD) frameworks.
3. **Increased Complexity in Identifying and Mitigating Bugs and Vulnerabilities:** Implementing a systematic approach to identifying and addressing bugs and vulnerabilities could add complexity, potentially leading to gaps in coverage. This is particularly concerning in quantum-hybrid systems, where vulnerabilities might be overlooked and exploited.
4. **Resource Demands for Developing New Methodologies:** Developing new methodologies and frameworks for quantum-hybrid systems could be resource-intensive, requiring significant time, expertise, and funding. Insufficient resources could result in incomplete or ineffective methodologies.
5. **Challenges in Advancing Quantum Techniques:** Promoting the use of advanced quantum techniques may be hindered by the steep learning curve and lack of expertise and tools. This could slow adoption and lead to gaps in security measures.
6. **Uncertainty in Closing Research Gaps:** Addressing research gaps could be challenging if the gaps are not clearly defined or if interdisciplinary collaboration is lacking. This uncertainty could lead to incomplete solutions, leaving quantum software systems inadequately secured.

Building on the analysis, Table:3.12 presents a comparison between the current state of Quantum Software Engineering (QSE) regarding the integration of PbD and SbD and the improvements outlined in the proposed action points. This table compares the current state with proposed improvements, highlighting the key areas where embedding PbD and SbD principles is essential, identifies the stakeholders responsible for driving these changes and maps to potential risks in implementation.

We presented the results of a systematic literature review investigating the existing empirical evidence regarding the definition and implementation of Quantum PbD and SbD that fulfills the guidelines of privacy and security provided in the Talavera Manifesto [59]. In other words, we analyzed the existing state-of-the-art Quantum PbD and SbD in the area of QSE and identified the challenges in the area. We collected empirical evidence from all studies in Quantum Software Engineering published until now. Specifically, we found 84 papers, 43 from journals, 38 from conferences, and three from workshops. The most popular empirical strategy was experimentation, the technology/methodology was Quantum Key Distribution, the software engineering implementation

Table 3.12: Comparison of Current and Proposed States in QSE

Action Point	Current State	Proposed New State	Who Should Act	Potential Risks
Integrate Privacy and Security by Design as a Core Requirement in QSE	Privacy and security considerations are often added later in the development process, if at all.	Privacy by Design (PbD) and Security by Design (SbD) principles are embedded from the start in QSE.	Quantum software engineers, project managers, and developers	Resistance to change may slow adoption, leaving systems vulnerable during the transition.
Adapt PbD and SbD Principles for Quantum Contexts	PbD and SbD principles are designed for classical systems and not fully adapted to quantum contexts.	PbD and SbD principles are tailored specifically for quantum computing, creating Q-PbD and Q-SbD frameworks.	Researchers in quantum computing, QSE architects, and developers	Technical challenges in adapting these principles for quantum contexts may delay framework development.
Systematic Identification and Mitigation of Bugs and Vulnerabilities	Bugs and vulnerabilities are identified and addressed on an ad-hoc basis, often after they cause issues.	A structured, proactive approach to identifying and mitigating bugs and vulnerabilities at all stages.	Quality assurance teams, software developers, and security experts	Increased complexity may lead to gaps in coverage, particularly in quantum-hybrid systems.
Innovate Quantum-Hybrid Software Engineering Practices for Privacy and Security	Existing practices are not sufficiently adapted to handle the unique challenges of quantum-hybrid systems.	New methodologies and frameworks are developed specifically for quantum-hybrid systems, ensuring flexible integration of PbD and SbD principles.	Quantum software engineers, researchers, and development teams	Resource demands could strain teams, resulting in incomplete or ineffective methodologies.
Promote the Exploration of Advanced Quantum Techniques	Limited exploration of quantum-specific properties to enhance security and privacy in software.	Advanced quantum techniques are widely applied to enhance security and privacy in quantum software.	Researchers, developers, and quantum computing experts	The steep learning curve and lack of tools may hinder adoption and create security gaps.
Address Research Gaps with Comprehensive Strategies	Research gaps in PbD and SbD within QSE are not systematically identified or addressed.	Comprehensive strategies are developed to close research gaps, ensuring secure and scalable quantum software.	Academic researchers, funding bodies, and QSE communities	Uncertainty in defining and addressing research gaps may lead to incomplete solutions.

phase was design and development, and the applied research area was computer science. For researchers and practitioners, these observations likely stem from the need to test and validate hypotheses, methods, or technologies. The emphasis on the design and development phases of software engineering, especially within computer science, signifies the attention to the area where most practical and theoretical efforts are concentrated.

To advance Quantum Software Engineering (QSE), embedding PbD and SbD principles throughout the development lifecycle is essential. These principles should be tailored to quantum applications, ensuring that vulnerabilities, especially in quantum-hybrid systems, are systematically addressed. Developing innovative methodologies and applying advanced quantum techniques are crucial for enhancing security. Additionally, addressing research gaps with focused strategies will help create secure, scalable quantum software that prioritizes privacy and security as the technology evolves.

Summarizing the most important outcomes of this study, we discovered: *(i)* Despite advances in quantum computing, research has mainly focused on software engineering and hardware, with limited attention to Quantum Privacy by Design (Q-PbD) and Quantum Security by Design (Q-SbD). This gap calls for more empirical studies on privacy and security in quantum systems. *(ii)* QSE research should prioritize quantum privacy and security, exploring quantum-hybrid programming, including quantum-enhanced machine learning and other advanced quantum technologies. *(iii)* This study is the first to analyze Q-PbD and Q-SbD in Quantum Software Engineering, laying the groundwork for future research.

Developing and validating specific frameworks and methodologies tailored for quantum applications presents an excellent avenue for future research stemming from this SLR. Empirical studies should focus on exploring the effectiveness of these frameworks in real-world quantum-hybrid systems, evaluating how well they address identified vulnerabilities and how seamlessly they can be integrated into the software development lifecycle.

3.2 Empirical Validation in QSE

This section will not only address the immediate challenges posed by emerging technologies but also lay the groundwork for a resilient framework that adapts to future advancements, ensuring robust protection against potential vulnerabilities in Quantum Hybrid Systems. This approach emphasizes the importance of a proactive stance in cybersecurity, enabling stakeholders to anticipate and mitigate risks before they manifest, thereby fostering trust and confidence in these systems.

3.2.1 QuIPS Framework

Based on the key action points from section 1.3, an impact analysis of PbD and SbD in Quantum Software Engineering carried out to propose a high-level framework for integrating PbD and SbD in Quantum-Hybrid Software Engineering Figure 3.12 called QuIPS, abbreviation of **Q**uantum **I**mpact on **P**rivacy and **S**ecurity. The concepts of privacy and security by design were analyzed through the perspective of Quantum Software Engineering (QSE). After a thorough review of the principles of Privacy by Design and Security by Design, we outline several key steps regarding the transition of security and privacy from classical models into quantum software engineering.

1. Moving from Classical Software Engineering to QSE mainly means using Quantum Programming Languages, working with data related to quantum systems, and performing quantum measurements to reach the same goals as in Classical Software Engineering.
2. Each principle regarding privacy and security by design is carefully reviewed based on the definitions outlined in Ann Cavoukian's Privacy by

Design [64] and the Open Web Application Security Project’s Secure by Design ¹⁸.

3. Each principle’s application to hybrid systems is assessed to determine its need, practicality, and significance for quantum hybrid systems.
4. The results are thoroughly recorded for each principle, highlighting any changes if applicable, along with the degree of change to be incorporated into the future system considering strategic planning, programming, and/or the application of the principles.

The principles of privacy and security by design are a mix of both conceptual and implementation-based directions to provide a holistic guideline for privacy and security-oriented software development. Thus, even if the programming paradigm or platform changes, the imperatives for privacy and security remain intact. However, some principles may apply more suitably to the changing paradigms while other may not be as relevant, necessitating ongoing evaluation and adaptation to ensure robust protection mechanisms are maintained throughout the system’s evolution. Consequently, following a meticulous examination of the foundational concepts of Privacy and Security by Design within the domain of Quantum Software Engineering, we subsequently delineate the implications of compliance with Quantum Technologies on Privacy by Design (PbD) and Security by Design (SbD)(Table 3.13 and Table 3.14). However, it is important to understand that all principles are equally important and should be adhered to during the development and maintenance of all quantum hybrid systems. The findings stated below are aligned with Piattini et al. [224], describing the degree of impact that quantum computing has on the current software engineering paradigm, where the most impacted phase of the quantum development lifecycle (QDLC) are the design and development phase.

Table 3.13: Impact of Principles of Privacy by Design Table 3.10
on QSE

High Impact	P1 (Implementation of Privacy Enhancing Technologies)
	P2 (Establish privacy settings as default)
	P5 (End-to-End Security - Data security retained at all times, and destruction on termination)
Medium Impact	P3 (Embedding privacy during the design phase)
Low/No Impact	P4 (Positive sum and not zero-sum, implementing privacy and security together conceptually)
	P6 (All privacy components and parts should be visible and transparent to the stakeholders)
	P7 (Privacy measures should be user-centric)

As seen in Table 3.13, PbD principles P1, P2 and P5 have a High Impact on the quantum software development lifecycle as they are involved in the core of design and development phase, which are the main phase where the contemporary systems are being impacted by quantum computing for example,

¹⁸<https://owasp.org/>

implementation of privacy enhancing technologies, embedding default privacy settings by default within applications, and end-to-end data security. P3 projects Medium Impact where it integrates privacy during design, but on the other hand P4, P6, and P7 have Low or No impact as they rarely impact the quantum implementations and provide mostly process oriented guidelines. However, these principles should be strictly followed for the development of the classical counterpart of the said quantum-hybrid applications.

Table 3.14: Impact of Principles of Security by Design Table 3.11 on QSE

Impact of Security by Design Principles	
High Impact	S4 (Defense against attack is provided by multiple security controls)
	S10 (Programmers should identify and fix issues correctly)
Medium Impact	S2 (Application must be secure by default)
	S5 (No extra privilege on failure)
	S8 (Not hiding core functionality, but maintaining security controls)
	S9 (Keeping the application simple from all points of view)
Low/No Impact	S1 (Minimizing the attack surface)
	S3 (Least privileges)
	S6 (Do not trust 3rd-party services)
	S7 (System access should be separated by duties)

As seen in Table 3.14, six security-by-design principles among 10 are impacted by the introduction of quantum computing in software engineering. The four remaining principles are uniform across programming paradigms or platforms dealing with conceptual principles. The most important security by design principles, S4 and S10, are directly correlated with the programming paradigms and technologies used and are, thus, impending radical changes while implementing multiple security controls to defend systems against attacks and identifying and fixing bugs correctly and in a timely manner. The SbD principles with Medium Impact are S2 (Application must be secure by default), S5 (No extra privilege on failure), S8 (Not hiding core functionality, but maintaining security controls), and S9 (Keeping the application simple from all points of view). If we look closely, these principles are mostly implementation-based and, given the situation, can be implemented via various paths, like clever utilization of classical elements in the quantum-hybrid architecture or via tweaking of quantum components, which is limited, pertaining to the limited resources.

Although quantum computing impacts most phases of classical software engineering, some phases are impacted more than others, whereas some receive very little or no impact. The QuIPS framework shows that the principles of PbD and SbD have a High and Medium impact on the Design phase and the Implementation phase of the QDLC. We can see in Figure 3.12 that the principles S2, S5, S8, and S9 impact the Quantum Design phase, which means that integration of these principles in the design phase would make the software more

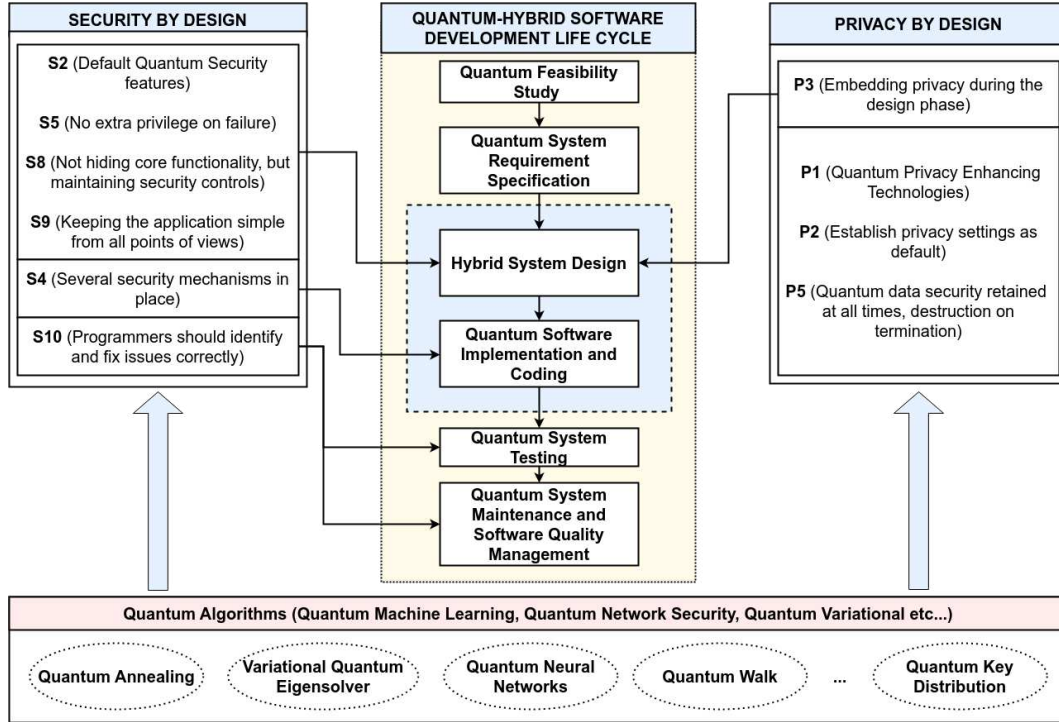


Figure 3.12: QuIPS Framework

security compliant. Similarly, S4 impacts the implementation phase, which deals with the development and integration of quantum security or encryption algorithms to embed security. S10, on the other hand, impacts Quantum Testing and Software maintenance, thereby enforcing the importance of bug and vulnerability identification and fixing in QSE. Regarding the principles of PbD, P3 directly impacts the Design phase, whereas P1, P2, and P5 impact the Development phase. It is important again to note that these principles of PbD have high and medium impacts on the implementation of QC.

The principles of PbD and SbD can be both process-oriented and data-oriented [225]. Process-oriented changes are mostly handled in the classical counterparts of hybrid systems or conceptually implemented in the system design. In contrast, the data-oriented changes could be implemented with the help of various quantum solutions like quantum machine learning (QML), quantum network security (QNS), quantum encryption, and quantum pure random number generator (QRNG).

In the rest of this section, we present various methodologies that implement different quantum solutions to embed or improve security and privacy in Quantum Hybrid Systems (QHS) in different sectors, for example, Automotive, Smart City, and Education sectors. Furthermore, we also developed a couple of frameworks regarding the integration of security and privacy principles in QDLC.

This thesis delves into automotive, smart cities, and education sectors for empirical case studies because of several essential considerations specifically

tied to research in Quantum Software Engineering and Cybersecurity.

The automotive sector was selected because of its increasing dependence on connected and autonomous vehicles, requiring advanced security measures to protect against sophisticated cyber threats. Quantum Software Engineering (QSE) can potentially address critical vulnerabilities related to real-time security in automotive systems through quantum-enhanced Intrusion Detection Systems (IDS) and advanced encryption methods. Given that automotive security directly impacts human safety and the fact that future vehicles will be technologically more sophisticated and connected, this sector provides a relevant and practical environment to test and validate quantum-enabled security approaches rigorously.

The smart cities sector was chosen because it involves complex, interconnected systems such as IoT networks, communication infrastructures, energy grids, and public safety mechanisms. Various cities around the world are actively exploring new technologies and approaches to transition into smart cities. These integrated urban systems present substantial challenges regarding data security, integrity, and privacy, particularly in the context of quantum computing's capabilities and associated threats. By embedding quantum-hybrid software engineering methodologies, this sector enables the empirical testing and validation of innovative quantum frameworks designed to enhance resilience and security across diverse urban infrastructures.

Finally, the education sector was selected for its strategic role in cybersecurity awareness, skill-building, and training. This sector provides a well-defined yet realistic environment to apply quantum technologies to enhance cybersecurity best practices and train and educate software development teams in industry, students and researchers in academia, and stakeholders on cyber threats and their mitigation strategies. By working closely with the educational community and industry, this thesis aims to foster the adoption of quantum-enhanced security practices among future software engineers and cybersecurity professionals, thus enhancing preparedness for the emerging quantum hybrid threat landscape.

In summary, the automotive, smart cities, and education sectors collectively provide diverse yet complementary environments that allow comprehensive empirical validation of the quantum security and privacy methodologies proposed in this thesis. Each sector's unique characteristics contribute directly to achieving the research objectives, ensuring that the outcomes demonstrate practical relevance and effectiveness within realistic quantum and quantum-hybrid scenarios.

3.2.2 Automotive Sector

The area concerning cyber security in the automotive field is growing faster to introduce more sophisticated techniques to detect possible attacks that may manifest on modern vehicles [226]. Therefore, Intrusion Detection Systems

(IDSs) have been proposed in the literature [227] dealing with attack identification that may occur mainly on the Controller Area Network (CAN) bus, which implements CAN protocol to allow communication between the various Electronic Control Units (ECUs) deployed in the vehicles [228].

The CAN protocol has several vulnerabilities like lack of encryption and authentication, allowing an adversary to inject malicious messages causing attacks, for example, fuzzing [229], spoofing [230, 231] and Denial of Service (DoS) [232]. Traditional machine learning-based solutions are usually time-consuming, computationally heavy, and don't meet real-time requirements, which are fundamental in the automotive field. On the other hand, lightweight algorithms are faster but less accurate [233]. The main challenge is to detect attacks on the CAN bus in the shortest possible time without affecting the accuracy of the algorithm negatively, emphasizing the need for new real-time solutions. So, the time factor is important in detecting attacks on the CAN bus because vehicles are considered safety-critical systems [234], and an attack or anomaly could compromise drivers' and passengers' lives. To this end, using Quantum algorithms could help identify attacks in a shorter time than traditional machine learning algorithms [235].

Based on these considerations, we study the applicability of Quantum solutions in the automotive security field and present some of the relevant methodologies proposed. In assessing the different approaches, we used Model Accuracy, Precision, Recall, and F1-score as measurable metrics to determine how well various Quantum Machine Learning (QML) models performed compared to traditional Machine Learning (ML) models in the proposed models. Additionally, we considered the time efficiency related to both the training and prediction stages in the comparison of the models.

Quantum-based Intrusion Detection System

We proposed a Quantum-based Intrusion Detection System to study the applicability of quantum solutions in the automotive industry. To prove the effectiveness and optimization capabilities of quantum annealing in detecting CAN bus attacks, the performance of a hybrid quantum-classical classifier was compared to that of a classical one. We have compared two binary classifiers: QBoost algorithm [47] and Support Vector Machine (SVM) using Support Vector Classifier (SVC) as its implementation from Scikit-learn¹⁹. Google and D-Wave have already described how the QBoost ensemble algorithm could be used efficiently to extend binary classification to quantum computing. The use of SVM is discussed by Mukkamala et al. [236] in which they have reached good results for DoS attacks. Tanksale [237] used SVM to detect anomalous behavior with high accuracy in the automotive field. Thus, its SVC implementation from Scikit-learn was used for this purpose here. We chose two types of datasets, DoS and Fuzzy because they have two different structures and different injection attacks. We used the generic IDS pipeline (Figure 3.13) during the experiments

¹⁹<https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html>

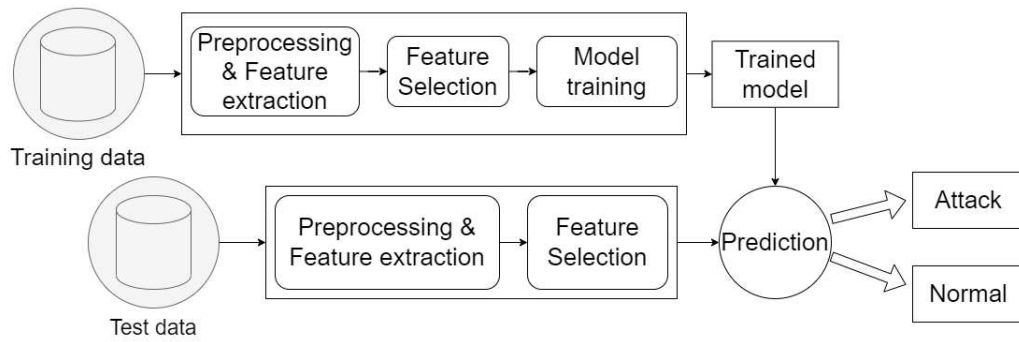


Figure 3.13: IDS pipeline

for both SVC and QBoost, which is summarized below:

- *Pre-processing & Feature Extraction*: Messages with a data field shorter than 8 bytes ('DLC' != 8) were removed, as classifiers are unable to process missing values. The *Frequency* feature, calculated from the timestamp, represents the time interval in milliseconds between two consecutive messages on the CAN bus with the same CanId. To address categorical data, two encoding techniques were employed: One-Hot Encoding and binary encoding, applied to the *CanId* and the string *Data*, which was formed by concatenating data bytes Data[0-7]. Then, each feature was individually scaled to a normalized range [min, max] using a MinMax scaler. Additionally, the classification labels were mapped as follows: 'R': non-malicious, 'T': malicious to '+1': non-malicious, '-1': malicious.
- *Feature Selection*: For each experiment, a subset of N features was chosen. These features included 'Frequency' and those derived from encoding the attributes 'CanId' and 'Data'.
- *Model Training*: Two binary classifiers, SVC and QBoost, were trained on each dataset.
- *Prediction*: The test data was input separately into each of the two trained models to generate predictions.

We used the QBoost algorithm available in the DWave Leap Quantum Cloud Service (DLQC)²⁰ that provides access to a portfolio of hybrid solvers located in different geographical regions. The back-end server consists of Quantum Processing Units (QPUs) and Solvers (a resource that runs a problem by directly interfacing with the QPU or by leveraging its CPU and GPU), which accept jobs and return the solution to the problem. The DLQC²¹ uses a series of quantum computers based on adiabatic quantum computing (AQC) technology using quantum annealing to solve optimization problems. Quantum annealing is a specific approach to quantum computing that focuses on finding the lowest energy state, or the optimal solution, of a given problem. QBoost

²⁰<https://cloud.dwavesys.com/leap/>

²¹<https://www.dwavesys.com/>

uses a boosting methodology by selecting a set of N weak classifiers. This optimization problem aims to minimize the squared loss between the actual and predicted targets of strong classifiers. The resulting optimization problem can be formulated as a binary quadratic model and solved on the D-Wave system.

To summarize, this study investigated the feasibility of using quantum technologies for security by comparing the performance of a hybrid quantum-classical classifier using QBoost against a classical Support Vector Machine (SVM) for detecting CAN bus attacks. Two datasets, DoS and Fuzzy, were used, processed through a standardized IDS pipeline. The workflow included pre-processing, feature selection, model training, and prediction. The results aimed to assess the effectiveness of quantum annealing for cybersecurity in automotive systems.

Quantum Powered Developer Support for Automotive Security

From previous discussions, we already know that modern cars use microcontrollers or Electronic Control Units (ECUs) to improve safety, security, and comfort for passengers. These ECUs manage many vehicle functions and advanced features, using communication methods like Controller Area Network (CAN), mainly favored for its speed and dependability. However, the weakness of CAN can be exploited through cyberattacks, which, on the other hand, can be handled using Intrusion Detection Systems (IDS) and the use of network segmentation. Moreover, machine learning (ML) algorithms offer potential solutions, but they come with long training times and accuracy issues. Therefore, human factors and real-time decision-making in risk management are critical, along with the need for secure software development practices in the automotive field. This study aims to present a quantum artificial intelligence-based vision model to help development teams effectively integrate cybersecurity measures throughout the automotive development process.

We propose a multi-class intrusion detection developer support methodology (Figure 3.14) that aims to identify attacks, which can be conducted on the CAN protocol using an IDS, and proposes a vision model to map the identified attacks using a taxonomy to support developers in fixing the vulnerabilities or protecting the ECUs from attacks.

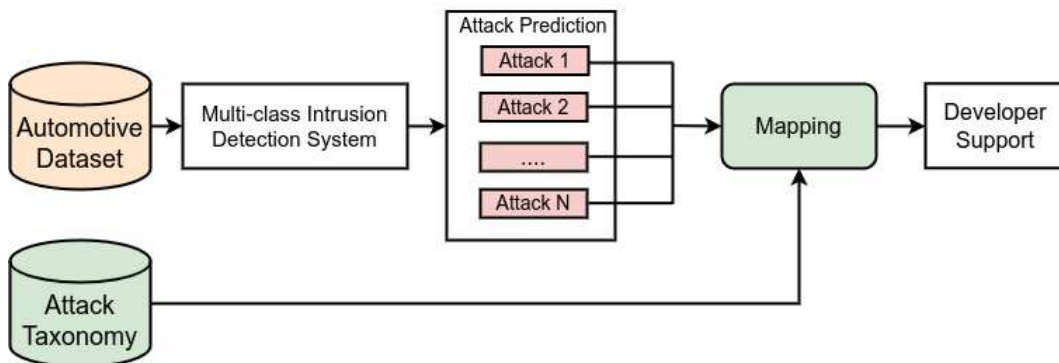


Figure 3.14: Proposed Dev-Support methodology.

The proposed methodology has two roles: the first is to identify attacks on the CAN bus, and the second is to map the attack typology with the taxonomy. Here, we benchmarked two traditional ML algorithms and a Quantum ML algorithm. In particular, the traditional ML algorithms considered here are Adaptive Boosting (AdaBoost) and Random Forest (RF). On the other hand, for QML, a quantum annealing-based algorithm QBoost is considered to perform the required classifications [235]. QBoost is inherently a binary classification algorithm, so the One-vs-Rest (OvR) methodology has been implemented to use the same for multi-class classification. The algorithms are tested on a state-of-the-art Car-Hacking Dataset [238, 239]. Once the IDS identifies an attack, it can be mapped using a taxonomy that must contain extensive information about the attack and support the developers in devising remedial measures. In particular, we choose the taxonomy proposed by Sommer et al. [240] as the authors proposed a comprehensive classification of 162 existing attacks from 74 publications that can be considered for the automotive security development process. This taxonomy was published by the authors in the Automotive Attack Dataset (AAD).

The taxonomy is hierarchically subdivided into three levels of abstraction, where level 1 uses input information from a high level of abstraction since the technical details are not yet known. On the contrary, level 2 and level 3 use a low abstraction level, thus more technical details, which can be used to provide information for analysis, development, and testing phases. To apply the taxonomy effectively, Description, Attack Class, Vulnerability, Interface, Consequence, and Exploitability categories were extracted.

The **Description** provides a brief overview of the analyzed attack. This includes the effect, target, and type of attack. It includes only level 1 of abstraction as it does not include any information for security testing. However, on a higher level, it can be used by the developer to understand the type of remediation that may be applied to fix the vulnerability that leads to the attack.

The **Attack Class** maps the attack with the typology identified by the IDS. This category presents information for levels 1, 2, and 3, providing comprehensive information about the identified attack. For level 1, the authors proposed the classification based on the Microsoft STRIDE [241] (Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, and Elevation of privilege). Level 2 specifies the attack and the different varieties of the identified threat. Finally, in level 3, the authors used the concrete term presented in the attack publication. In relation to the proposed methodology, only level 1 is used for the Attack Class as it is important to map the attack type.

The category **Vulnerability** mentions the exact vulnerability that has made the attack possible. The authors used the Common Weaknesses Enumeration (CWE) [242] that distinguishes different research, development, and architectural concepts. Both level 2 and level 3 abstractions were chosen as they mention and describe the vulnerabilities as described in the attack publication.

The **Interface** describes where the attack has been conducted (for example,

on-board diagnostic (OBD) port, ECU, Bluetooth). In this case, using the description and the vulnerability category, the developer can update methodologies or devise new ones to increase security, for example, encryption or authentication mechanisms.

The **Consequence** describes the exact impact of the attack or, in other words, the result when the vulnerability is exploited. Level 1 information is enough for this category, as the developer can use this information to map the consequences of the attack with the taxonomy to get further insight into the attacks and related CWEs.

Finally, the **Exploitability** represents a metric that can be used for the attack's relevance. The authors used the Common Vulnerability Scoring System (CVSS), which provides a way to capture the principal characteristics of a vulnerability and produce a numerical score.

The proposed methodology can be better understood through an example wherein the proposed methodology performs the multi-class classification to find the typology of an attack and consequently uses the taxonomy to support the developer during different phases of the software development life-cycle. Figure 3.15 shows the aforementioned example in the purview of the proposed methodology. This example explains how the methodology classifies a Denial-of-Servie (DoS) attack and thereafter uses the taxonomy to provide development strategies for the developer.

In the first phase, the multi-class IDS connected to the CAN bus identifies the typology of all possible attacks. In particular, in a real scenario, the IDS model can encounter different types of attacks also identified by the automotive industry. Using the attack taxonomy, the developer extracts detailed information about the attack, class, related vulnerability, CVE IDs if exists, and the description of the attack publication from Sommer et al. [240].

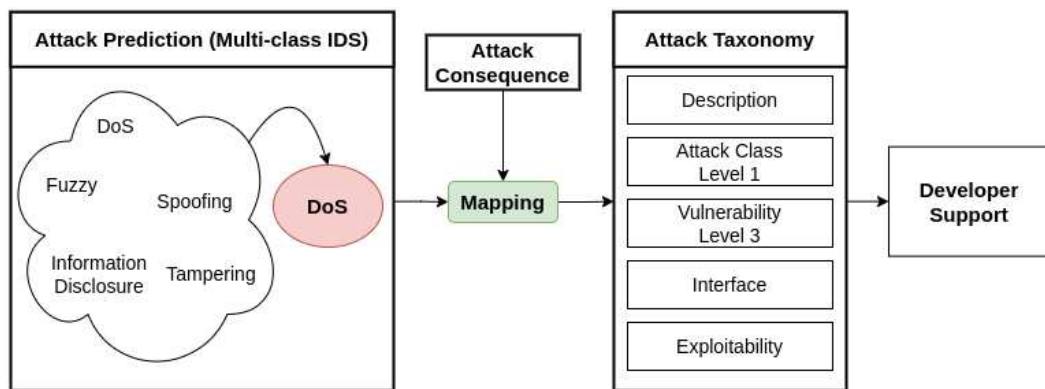


Figure 3.15: Proposed example to map the DoS attack using the taxonomy to support the developer.

Once a security incident is classified as a DoS attack, the developer can map the identified DoS attack using the "Attack Class" category considering level 1. This field will give back the list of all attack publications mapped by Sommer

et al. that performed a DoS attack. The description field allows the developer to get an overview of the attack. The taxonomy maps several DoS attacks with the attack typology classified by the IDS. At this stage, without any additional information about the impact of the attack, the developer can still devise a plan to fix the vulnerability at a more holistic level. Based on the CVSS exploitability score, the developer can plan to implement fixes for the components or cases where the exploitability scores (CVSS) are the highest. Such an approach will fix the problem for sure, but it can be time-consuming and rigorous.

With additional information about the attack, the most important being the **attack consequence**, the developer can quickly pinpoint the exact attack incident. In this example, we assume that the consequence, in this case, is: **Engine doesn't start**. This leads to the exact attack description as per the taxonomy: "Denial of service attack to prevent the start of the vehicle engine by continuous sending of CAN messages with ID 0 (violation of CAN standard)", which was carried out in [243] by Miller and Valasek. The incident description provides the information that a CAN standard violation by sending numerous high-priority messages having CAN message ID 0 prevents the vehicle engine from starting. Mapping this information with level three of the vulnerability category shows that the attack is caused because of "Improper Adherence to Coding Standards, Violation of the CAN standard". The developer can, thus, implement proper coding standards and guarantee that the CAN standards are not violated in the system. For example, the developer can add authentication mechanisms to avoid such attacks. In particular, the interface category shows that the attack was conducted by exploiting the OBD port, generally used for diagnostic purposes, to send malicious messages or receive information sent by the ECUs. This incident was evaluated with a score of 2.84.

To summarize, the developer can use authentication mechanisms to allow only certain users with a key to send or receive data from the CAN bus using the OBD port. This information can prove very useful during the testing and development stage of the automotive software life-cycle. In the analysis phase, the developer can also use the Common Vulnerability and Exposure (CVE), a dictionary of publicly known vulnerabilities, to map the same attack using the **CWE-ID** category, allowing the developer to identify an existing vulnerability and add remediation. The proposed methodology provides additional information about mitigating existing vulnerabilities with the help of Attack Typology and associated Attack Taxonomy to support the developer in implementing proper planning and design right from the analysis phase. The information from the attack consequence aids in mapping the Attack Typology to corresponding vulnerability details, thereby helping the developer in the Design and Analysis Phase. The developer, on the other hand, gains invaluable insight and information on secure automotive development.

Quantum Artificial Intelligence in Extending Developer Support

Using Quantum algorithms could help identify attacks in a shorter time than

traditional machine learning algorithms [235]. However, using only IDS is not enough to ensure the security of vehicles. It is also necessary to offer other perspectives to automotive developers: to understand how to solve a particular vulnerability (if the attack has occurred) or prevent it (e.g., through testing steps) [244, 245].

Based on these considerations, we proposed a methodology called SeQuADE (Secure Quantum Automotive Development and Engineering) to identify CAN attacks and, based on the predictions, provide the developers with the information to fix the vulnerabilities that cause these attacks. In particular, using an open-source taxonomy created by Sommer et al. [240], we extended it considering other vulnerabilities obtained from the NVD using a Natural Language Processing (NLP) pre-trained model to obtain additional information in the NVD. This information could help the developers understand how to fix the vulnerabilities or prevent automotive attacks [246]. QBoost was used for the IDS, and since it provides only a binary classification, the principle of multi-classification using the One-vs-Rest (OvR) technique was used. Since the ECUs have limited computational power, deploying a quantum machine learning model into these components is difficult. However, it is possible to re-engineer the ECUs to send the CAN messages to the D-Wave servers for the training and prediction phase.

Figure 3.16 shows the SeQuADE methodology adopted to identify attacks that can occur in the CAN bus and to support developers by proposing associated automotive vulnerabilities and solutions obtained from the NVD by exploiting the taxonomy of Sommer et al. [240, 247, 248]. Firstly, the SeQuADE presents the IDS model. Since QBoost provides only a binary classification, the OvR approach was used to provide a multi-classification solution. Based on the model prediction, the description is extracted from the taxonomy exploiting the attack class category. Finally, a pre-trained sentence transformer model automatically obtains sentences similar to the description obtained from the taxonomy and the CVE description from the NVD. Then, the top three similar sentences are recommended to the developers with additional information: the CVE ID, the description of the CVE, the references, the CVSS score, the Common Weaknesses Vulnerability (CWE) ID, and the CWE description.

The referenced methodology assists developers by providing structured insights into security threats, mitigation strategies, and prioritization of vulnerabilities. By leveraging the obtained references, developers can understand real-world cyberattacks and apply appropriate security measures. The CVSS framework enables risk-based prioritization, while CWE descriptions offer targeted remediation strategies, such as cryptographic solutions for replay attacks. Additionally, taxonomy models like those proposed by Sommer et al. [240] facilitate the identification of similar threats through open-source databases, enhancing proactive security measures.

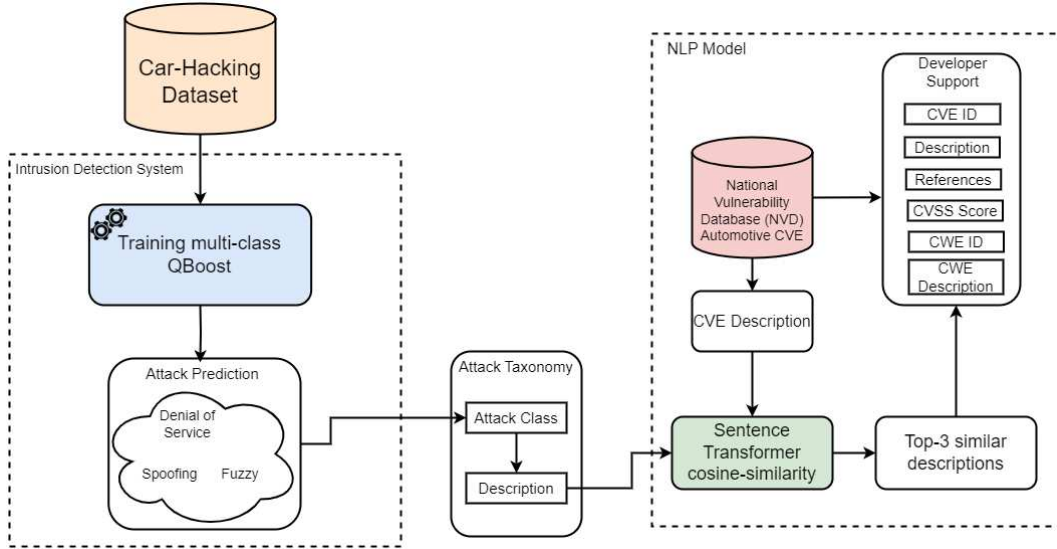


Figure 3.16: Proposed methodology to support the developer in the automotive field: SeQuADE (Secure Quantum Automotive Development and Engineering).

3.2.3 Smart City Sector

Smart Cities are not visions of the future; they are a growing reality, transforming urban environments globally. Cities are adopting advanced technologies such as IoT devices, AI, ML, Cloud Computing, Machine-to-Machine Communications, and Mesh Networks to evolve into Smart Cities. The aim is to enhance six key fields: ‘smart government,’ ‘smart economy,’ ‘smart environment,’ ‘smart living,’ ‘smart mobility,’ and ‘smart people’ [249, 250]. Some cities like Barcelona, Dubai, London, Singapore, Tokyo, and Hong Kong are leading in the smart city movement.

Smart cities leverage an extensive deployment of devices, sensors, and interconnected infrastructures that facilitate seamless interaction among individuals and objects [250]. Technologies such as Big Data, IoT, Cloud, and Mobile Computing have become foundational to smart cities [251]. However, these advancements create vast data networks vulnerable to cyber threats, including DoS, DDoS, password attacks, backdoor exploits, and ransomware [252, 253]. Thus, robust security and privacy measures are paramount to protect the collection, processing, and anonymization of smart city data.

To manage increasing data demands, Intelligent Operations Centers (IOCs) enhance city operations. Notably, the IBM Intelligent Operations Center (IOC) autonomously detects conflicts, optimizes operations, and adapts based on real-time data [254, 255]. IOCs not only optimize city operations but also enforce privacy controls [256, 257]. Parallel to IOCs, Security Operations Centers (SOCs) focus specifically on safeguarding organizations from cyber threats [226]. SOC teams utilize Security Information and Event Management (SIEM) systems to detect and mitigate potential security incidents using ML-based and fuzzy logic algorithms [258, 259].

The emergence of Quantum Computing (QC) introduces a transformative paradigm to computational sciences, promising exponentially faster processing and solutions to previously unsolvable problems [34, 82, 35]. Currently used mainly in research, QC is expected to become more accessible and integral to various applications, including smart cities. Quantum Machine Learning (QML), such as QBoost [47], can accelerate decision-making, crucial for timely threat detection in Smart City IoT scenarios [90]. Compared to traditional classifiers, such as Support Vector Machines (SVM), QBoost demonstrates better performance in reducing detection time.

However, the integration of quantum technologies into smart city systems involves overcoming several challenges, particularly in combining privacy and security protections across both classical and quantum paradigms. Hybrid quantum-classical architectures could enhance threat detection capabilities and decrease data processing time. The following subsections discuss reference architectures integrating quantum and classical solutions.

Hybrid Quantum-Classical Methodology

We propose a Quantum as a Service (QaaS) architecture for Secure Smart City using Quantum Machine Learning (QML) algorithms for the detection of threats or security incidents (Figure 3.17). This is an extension of security incident detection in existing systems through a service-based architecture that can potentially improve the existing IOCs or SOCs. It amalgamates two distinct security implementation systems: QRadar and the DWave Leap Quantum Computing platform. QRadar, a widely recognized SIEM system, identifies intrusions and issues by employing a rule-based approach. Security specialists configure a set of rules to analyze data received from various interfaces within the Smart City ecosystem. In contrast, the DWave quantum platform harnesses QML algorithms to train classification models for detecting attacks and intrusions within the system. It achieves this by processing data collected from the Smart City infrastructure.

The envisioned system consists of a robust three-layer architecture, meticulously organized into six core components for optimal functionality. These components include: (a) *Smart City Data Source*, (b) *Dashboard*, (c) *Principal Application*, (d) *Database*, (e) *Quantum Cloud Service* and (f) *QRadar - SIEM*. In this context, the presentation, application, and data tiers were developed as distinct infrastructures, offering increased security and modularity. Each of these components contributes uniquely to the overall functionality of our system, ensuring a robust and comprehensive approach to data processing and security in the Smart City environment. The details of each of the components are presented in greater detail in the following subsections.

Smart City Data Source

The Smart City data source, illustrated in Figure 3.17.a, acts as the central intelligence hub within the urban ecosystem. It functions as a pivotal point for seamless information exchange among the interconnected components of the Smart City, all designed to enhance the quality of life of the residents. These data sources are distributed across the city, forming an essential network

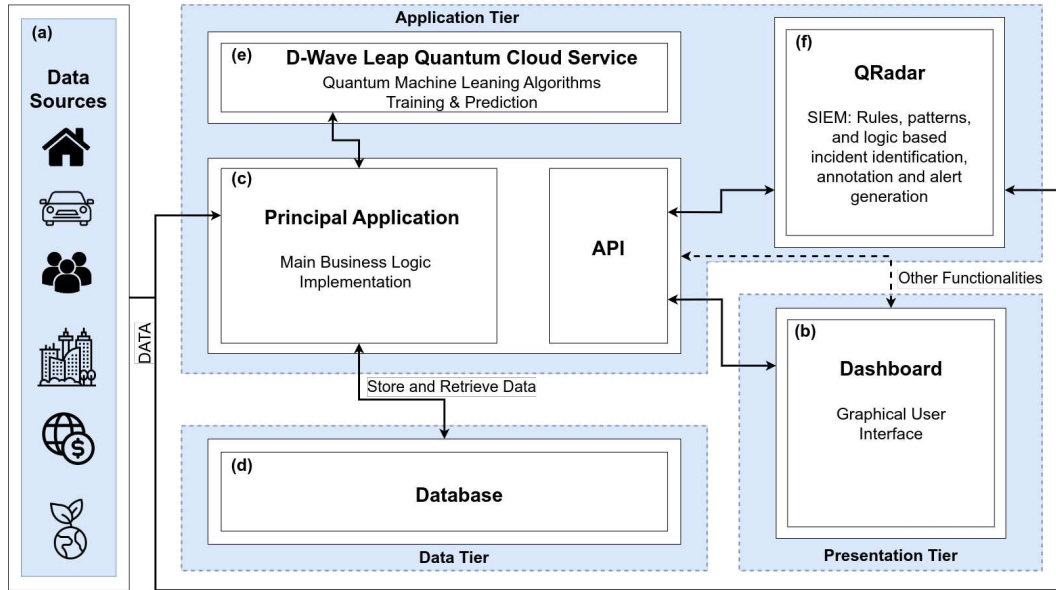


Figure 3.17: Hybrid Quantum-Classical Architecture for Smart City Security.

that drives the system's functionality. Although physically located outside the core system, they are indispensable to the overall ecosystem. Data originating from various Smart City elements is delivered in formats specific to the type of log source or data source integrated into the system. Typically, this data is received in multiple formats, including Common Event Format (CEF) for network and security logs, Syslog (a protocol for transmitting event messages and log data), JSON (JavaScript Object Notation), XML (eXtensible Markup Language), and SNMP (Simple Network Management Protocol), along with logs generated by network devices, database systems, and flat files such as CSV (comma-separated values).

Smart City Security Dashboard

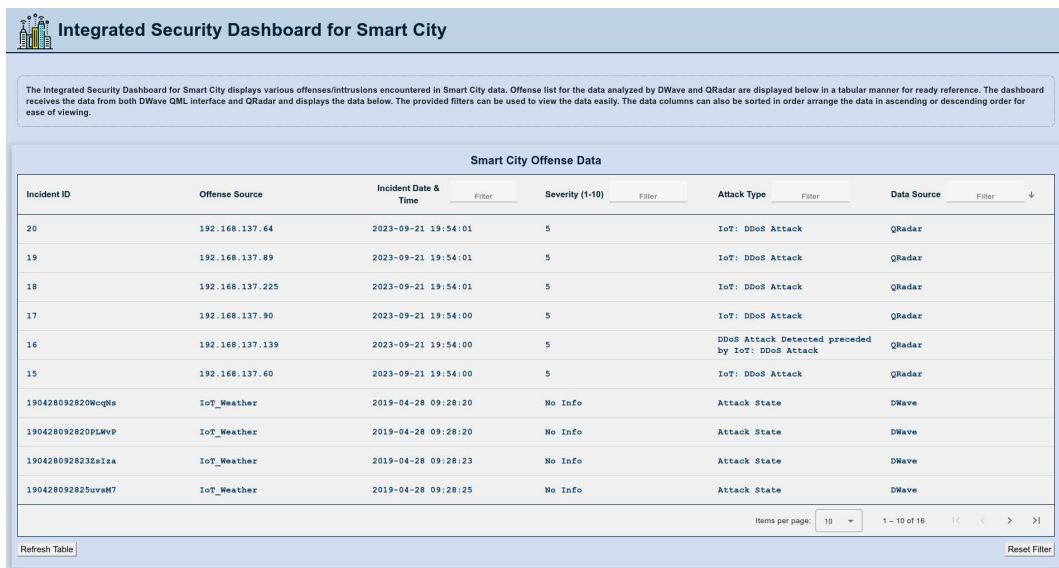
The proposed architecture introduces a dynamic graphical user interface (GUI), as exemplified by the interactive dashboard featured in Figure 3.17.b. This GUI serves as a real-time platform for monitoring and responding to a wide range of security incidents and attacks unfolding across various sectors within the Smart City infrastructure. The dashboard is intelligently structured so that different incidents associated with each Smart City sector can be accessed through filters on the respective headers, presenting them in an accessible tabular format. In the spirit of immediate responsiveness, the dashboard promptly updates upon incident registration, furnishing comprehensive details regarding the encountered threat. Each threat or attack is identified by a unique identifier, date, and time of the incident. The dashboard data are loaded from RESTful API (Representational state transfer application programming interface), which fetches data from the database tables. This API functions as the GUI's primary data conduit, presenting information in a format that allows security experts

to swiftly evaluate and counter threats.

Moreover, the dashboard supports the review of historical incidents, enabling security specialists to reference prior occurrences for in-depth analysis and informed decision-making. Figure 3.18 serves as an illustrative sample of the dashboard, a pivotal component within our proposed architecture, spotlighting real-time alerts stemming from incoming Smart City data. The dashboard systematically organizes attack incidents in rows, each tagged with a distinct incident ID and supplemented with relevant particulars such as incident timestamp, severity (when available), category, and attack type (also when available).

In addition to this, the GUI allows security experts to make decisions based on the attacks they receive. For example, considering the dashboard in Figure 3.18, the *Incident IDs* from 15 to 20 correspond to the threats identified by IBM QRadar SIEM. In this case, a security analyst can use the **Offense Source** IP to block the attacker and avoid the Distributed-Denial of Service (DDoS) attack (Attack Type column). On the other hand, for the attacks identified by the IDS, the security analysts could obtain the source IP from DWave (based on the prediction) and reconstruct the attack chain to identify the attacker IP and block the ones sending malicious content. In this preliminary work, the dashboard only shows information about the attack prediction with the **Offense Source** using a group-based typology of the type of attack, for instance, *IoT_Weather* in the dashboard screenshot (Figure 3.18), owing to the limitation of the dataset in consideration. In the future, we plan to connect the dashboard to a real IoT device that provides IP and other detailed information.

The dashboard was developed using Angular (Version - 15.1.6) and Node.js (Version - 18.14.1).



The Integrated Security Dashboard for Smart City displays various offenses/intrusions encountered in Smart City data. Offense list for the data analyzed by DWave and QRadar are displayed below in a tabular manner for ready reference. The dashboard receives the data from both DWave QML interface and QRadar and displays the data below. The provided filters can be used to view the data easily. The data columns can also be sorted in order arrange the data in ascending or descending order for ease of viewing.

Smart City Offense Data					
Incident ID	Offense Source	Incident Date & Time	Severity (1-10)	Attack Type	Data Source
20	192.168.137.64	2023-09-21 19:54:01	5	IoT: DDoS Attack	QRadar
19	192.168.137.89	2023-09-21 19:54:01	5	IoT: DDoS Attack	QRadar
18	192.168.137.225	2023-09-21 19:54:01	5	IoT: DDoS Attack	QRadar
17	192.168.137.90	2023-09-21 19:54:00	5	IoT: DDoS Attack	QRadar
16	192.168.137.139	2023-09-21 19:54:00	5	DDoS Attack Detected preceded by IoT: DDoS Attack	QRadar
15	192.168.137.60	2023-09-21 19:54:00	5	IoT: DDoS Attack	QRadar
190428092820WcQn	IoT_Weather	2019-04-28 09:28:20	No Info	Attack State	DWave
190428092820PLMVP	IoT_Weather	2019-04-28 09:28:20	No Info	Attack State	DWave
190428092823sIxa	IoT_Weather	2019-04-28 09:28:23	No Info	Attack State	DWave
190428092825uvaM7	IoT_Weather	2019-04-28 09:28:25	No Info	Attack State	DWave

Items per page: 10 1 - 10 of 16

Refresh Table Reset Filter

Figure 3.18: Integrated Smart City Security Dashboard for analysis and investigation.

Principal Application

At the core of the proposed architecture, as depicted in Figure 3.17.c, lies the main application, a critical component responsible for processing incoming incidents from various Smart City sectors. These sectors encompass areas such as government, mobility, living, environment, and people. Employing a well-structured Intrusion Detection System (IDS) pipeline, the application methodically sifts through this influx of data to pinpoint potential security incidents, a process illustrated in Figure 3.19. Following the prescribed IDS methodology, the data undergoes initial scrutiny and cleansing, shedding irrelevant fields that hold no significance. Subsequently, a sequence of operations extracts and prioritizes the most pertinent features for further analysis. The system then embarks on the training of machine learning models, subjecting them to rigorous validation before entering the prediction phase. During this predictive stage, test data is meticulously processed and subjected to classification by the trained ML models. The outcomes of these predictions are then compared against existing data to gauge the efficiency of the models, a task often assessed through popular quality metrics such as accuracy, precision, recall, or the F1-score.

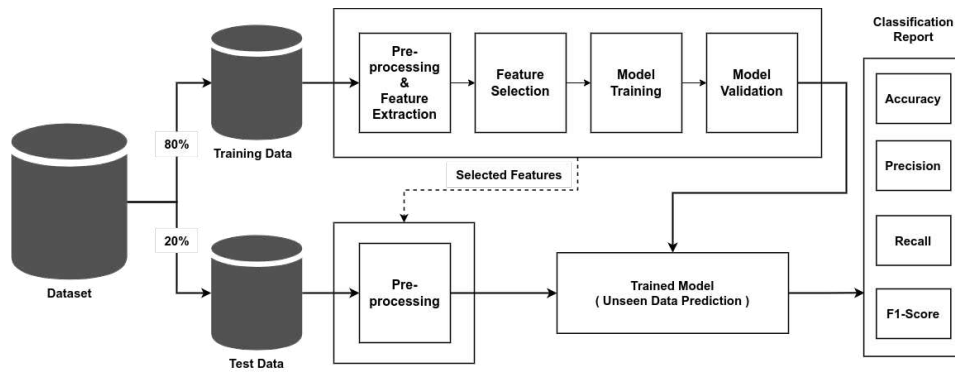


Figure 3.19: IDS Pipeline.

The principal application runs the core program that carries out all the functionalities of the proposed architecture. It curates the Smart City data, as mentioned before.

Within the proposed architecture, the training and testing of data are conducted within the D-Wave Leap Quantum Cloud (DLQC)²² (Figure 3.17.e), which is external. The application further forms secure connections to DLQC instances through token-based native API. For DLQC, D-Wave Python libraries are harnessed to format data suitably for Quantum Machine Learning (QML), with a focus on employing the QBoost [47] algorithm. After the training and validation stages, QML models are preserved locally ready for use in real-time predictions. When the main application initiates real-time predictions, the prediction model is loaded into the Quantum Cloud.

The main application is capable of processing single or batch incident predictions

²²<https://cloud.dwavesys.com/leap/>

received from data sources. Quantum Cloud assumes responsibility for incident-type predictions, leveraging QML algorithms to distinguish between normal and malicious incidents. Classification results are relayed back to the main application via the token-based API, subsequently updating the database with the most recent attacks, threats, or security incidents.

To maintain the model's effectiveness, the proposed system consistently trains QML models using a diverse array of datasets, including those sourced from reputable organizations or institutions. The best-performing model, determined through multiple runs, is retained for real-time predictions. Ensuring secure and efficient data access, our system employs a RESTful API interface, enabling seamless communication with the database instance. This API features distinct endpoints for data retrieval from various database tables and sources, converting the data into the JSON format for easy data fetch by the dashboard.

The proposed application was developed using Python 3.8, while Flask 2.2.2 was employed to establish the RESTful API for the retrieval of database and file data. The API's capabilities can be seamlessly expanded to encompass additional POST, PUT, PATCH, and DELETE operations in the future, allowing for flexibility to accommodate specific database needs.

Database

Situated at the core of the proposed architecture, the database (DB) depicted in Figure 3.17.d serves as the basis of our data management infrastructure. It acts as the repository for the information streamed from our diverse data sources. Each sector of the Smart City contributes its unique data, which is meticulously organized into dedicated DB tables. These tables, akin to specialized vaults, are tailored to accommodate the distinctive features and information inherent to their respective sectors. This structured approach ensures not only efficient data management but also facilitates quick retrieval.

In order to facilitate seamless data retrieval, we have implemented a RESTful API that adeptly selects data from incident tables associated with each Smart City sector. Serving as a route between the user interface and the database, this API allows for the rapid and secure extraction of data. It acts as a vital link, enabling the GUI to provide real-time insights into incidents occurring across the Smart City's diverse sectors. Furthermore, the main application actively interacts with these DB tables, continuously updating them whenever an incident unfolds within a specific sector. This dynamic interaction ensures that the database remains dynamic, updating in real-time to reflect the continuously evolving events in Smart City.

PostgreSQL 15.2 was chosen as the database technology for the system, which offers the scalability and reliability required for the storage and retrieval of critical information within our Smart City security architecture.

Quantum Cloud services

The D-Wave Leap Quantum Cloud (DLQC) Service, as delineated in Figure 3.17.e, occupies a pivotal role within the proposed architecture affording access to quantum computing capabilities via a secure native API mechanism, fortified by robust token-based encryption for authentication. While DLQC operates as an external service, its logical placement within the application tier is firmly rooted in its role as a strategic enabler of decision-making processes. Within this architectural context, DLQC takes receipt of meticulously pre-processed data originating from the principal application, sending those to a quantum computer encompassing data training and predictive analysis. DLQC harnesses the computational prowess of Quantum Unconstrained Binary Optimization (QUBO) algorithms, a cornerstone of quantum computing. This utilization extends not only to the training of quantum models but also to predicting outcomes, thus harnessing the computational supremacy intrinsic to quantum paradigms. Subsequently, DLQC returns the results of its training and prediction endeavors to the main application, which proceeds with further processing, thereby culminating in a cohesive decision-making framework.

The strategic deployment of QBoost [47], a reputed quantum algorithm, supports the foundation of architectural design. This algorithm is deployed on the QPU Hybrid Solvers residing on the Advantage_system5.3 platform (DWave QPU solvers for the European region). Additionally, the Leap Quantum Cloud service is available in diverse geographical locations, ensuring access to strategically positioned quantum hybrid solvers on a global scale. The selection of the most appropriate solver is executed judiciously, accounting for regional distinctions and the availability of solvers.

Notably, the solver possesses distinctive characteristics deeply rooted in the principles of quantum annealing, marked by its formidable ensemble of 5615 operational qubits. These qubits operate at an exceedingly low-temperature threshold of 16.4 ± 0.1 mK (millikelvin), a prerequisite for harnessing the unique computational capabilities in quantum mechanical phenomena. In summation, DLQC represents the veritable quantum computational powerhouse within our architectural expanse, endowing our infrastructure with advanced capabilities in data training and prediction, all emanating from the vantage point of quantum computational paradigms.

QRadar - SIEM

IBM QRadar (Figure 3.17.f) forms another vital part of the main application, where it effectively incorporates external threat intelligence data to bolster its threat detection and incident response capabilities. It receives streams of data from a multitude of external sources, including commercial threat intelligence providers, open-source feeds, and customized data repositories. It also maintains reference data sets based on this external threat intelligence, which are continually updated to reflect the latest threat information, ensuring that the system operates with current knowledge. It correlates external threat intelligence with internal security events and logs, identifying patterns, anomalies, or matches that signify potential threats. Custom correlation rules trigger alerts

when QRadar detects a potential threat or matches it with external threat indicators.

Risk scores and severity levels are assigned to incidents based on this data, helping security teams prioritize their responses effectively. The pattern and rules-based risk and threat identification are shared with the users over a secure API local to the QRadar implementation site. The use of the community edition of QRadar enables the proposed architecture to exploit the API-based data from both JSON dumps and local API.

QRadar's API empowers users to leverage the platform's rich security data and capabilities in a flexible and customized manner to meet their organization's specific security needs. In the purview of the proposed architecture, the cured and normalized data are fetched from the API to adapt the data to be displayed in the dashboard. The same functionality can also be implemented by taking JSON dumps at regular intervals and updating the dashboard via architecture-side API.

3.2.4 Education Sector

The creation of secure software presents significant challenges, despite the existence of numerous standards pertaining to secure software development, as it necessitates a high level of expertise, comprehensive knowledge, and substantial temporal investment. Practitioners and evaluators within the industry cannot acquire such expertise instantaneously and, therefore, require a structured framework and systematic training to facilitate development and informed decision-making throughout the Software Development Life Cycle (SDLC).

Emerging technologies have ventured into the realm of Quantum Computing (QC) as a feasible and accelerated alternative to Machine Learning (ML), with empirical evidence demonstrating its capability to address intricate problems within a reasonable timeframe [260, 90]. Drawing upon this premise, we have developed a vision model that incorporates Quantum Artificial Intelligence (QAI) to establish a Computer Science education and training framework for Developers and Testers, thereby equipping these professionals to make more informed and judicious decisions throughout the Software Development Life Cycle (SDLC), with the adaptability to be customized and executed across various sectors or academic institutions.

Industrial Cyber Security Education with Quantum Artificial Intelligence (ICSEQAI)

ICSEQAI is a high-level abstraction of a learning framework to support developers and testers during the SDLC (Figure:3.20). As per the ICSEQAI (Figure:3.20) vision model, artefacts including but not limited to Company Proprietary Security Guidelines, Industry Security Standards, previously analyzed

solutions or reports, and vulnerable component data can be processed and repurposed to support decision-making by the developers and testers during the SDLC. These artefacts can be translated into specific rules necessary for secure software development [245]. The rules can be mapped using industry standards, for example, automotive industry standards or global application standards like OWASP Application Security Verification Standard²³ to generate a robust model for decision-making. Quantum Artificial Intelligence (QAI) can be trained for such a constrained model to provide an optimal solution, which can be used to update existing security guidelines or standards and educate the developers and testers regarding CS during SDLC.

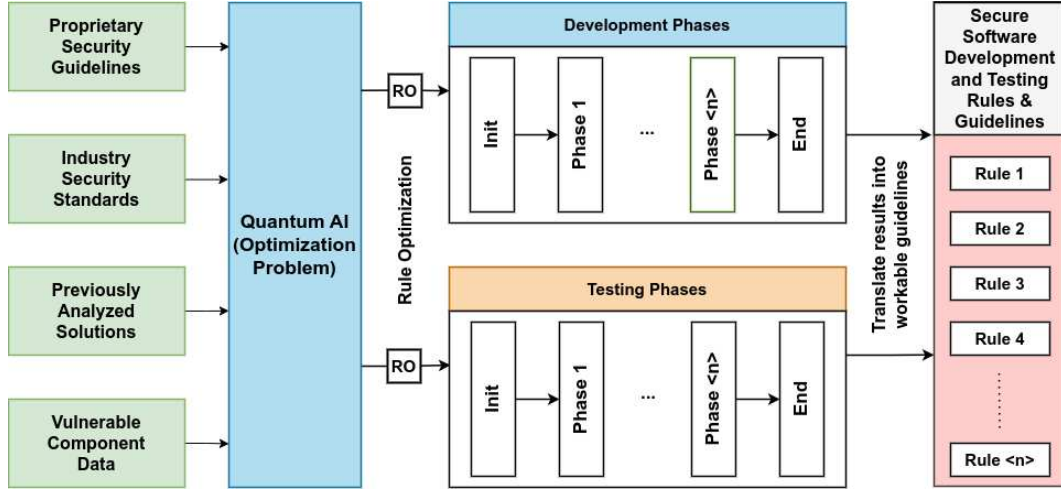


Figure 3.20: ICSEQAI Vision Model to support Developers and Testers during SDLC

The Industry Standard for Road vehicles — Cybersecurity engineering, ISO/SAE 21434:2021 [261], a high-level guideline for security solutions in the automotive domain, does not provide concrete design ideas, making assimilation into SDLC difficult. Therefore, the guide can be split into multiple rules for SDLC support. Vulnerable Electronic Control Units (ECU), building blocks of a car’s internal network, can provide invaluable information about specific attacks and details about the vulnerabilities. Historical security analyses, solutions, and fixes for previous ECU attacks and vulnerabilities are equally important as they contain precious information about specific drawbacks and implemented fixes.

To understand the model, we present, as a case study, a secure software development process to fix a vulnerable automobile component. A constrained optimization problem can be formulated from the dataset of similar rules, standards, data, and constraints based on ISO/SAE 21434:2021, OWASP, and SDLC, which can be trained on QAI algorithms to perform Rule Optimization (RO in Figure:3.20), generating optimal secure development rules for vulnerability fixes. For example, ICSQAI would suggest the best K strategies among N different actions suited to a specific developmental phase, where K is the

²³<https://owasp.org/www-project-application-security-verification-standard/>

constraint (rules to fix vulnerability). These rules can be translated to update company security guidelines, propose updated standards, and educate and train Developers and Testers to adjust to newer security requirements and standards during SDLC.

The proposed ICSEQAI model is designed to effectively train and continuously educate company personnel, including architects, developers, and testers, in cybersecurity skills. For developers, the model can provide automated, context-specific messages during the coding or unit testing phases, detailing secure coding practices relevant to the specific module being worked on. For testers, it offers comprehensive, on-screen guidance to facilitate effective testing of newly implemented fixes.

Additionally, various QAI algorithms can be trained and applied in different contexts to meet the model's objectives. Quantum classification algorithms can assist in identifying and classifying vulnerable components within the system. Meanwhile, optimization algorithms, such as constrained quadratic models (CQM) or discrete quadratic models (DQM), can be leveraged to generate rules that guide training and education efforts. Moreover, reinforcement learning techniques using variational quantum circuits can be employed to adapt and integrate new development guidelines, ensuring the continuous improvement of existing protocols [262].

Embedding privacy and security measures throughout development phases is essential for maintaining trustworthiness. However, existing literature lacks formal guidance on quantum-specific bugs and vulnerabilities, and the quantum software development life cycle (QDLC) is insufficiently equipped for privacy-centric development. As quantum computing advances, development teams face challenges in implementing security measures, highlighting the need to translate guidelines into the quantum domain and provide appropriate training. Unique bugs from quantum characteristics, alongside classical bugs in hybrid systems, complicate the QDLC stages.

Hereafter, we present the solutions to integrate Privacy By Design (PbD) and Security by Design (SbD) principles into the QDLC, ensuring that privacy and security are prioritized from the initial stages of development. This integration not only enhances the robustness of quantum applications but also fosters trust among users by safeguarding sensitive information against emerging threats inherent in quantum computing environments.

3.2.5 Security and Privacy-Aware Quantum Software Development

As quantum computing (QC) gains traction, quantum development teams face steep challenges in implementing security and privacy measures. Therefore, it's necessary to translate security and privacy guidelines into the quantum domain, train and educate quantum development teams with the required skills, and seamlessly integrate privacy and security into new and existing systems.

Aligned with this objective, the research study attempts to confront these challenges by employing the following frameworks:

- (i) *Quantum Privacy Knowledge Base (QPKB)* which amalgamates five (5) key elements, Quantum Privacy by Design (QPbD), Quantum Privacy Design Strategies (QPDS), Quantum Privacy Patterns (QPP), Quantum Vulnerabilities [1, 2], and Quantum Hybrid Context as essential solutions;
- (ii) *Quantum Privacy-Oriented Software Development (QPOSD)* that supports quantum software development by integrating privacy and security during the quantum software development life-cycle;

3.2.5.1 Quantum Privacy Knowledge Base

In the Noisy Intermediate-Scale Quantum (NISQ) era, where we are looking towards Quantum Hybrid Software Development, it is imperative that Security and Privacy concerns are aligned with both classical and quantum elements of the system, encouraging us to propose Quantum Privacy Knowledge Base (QPKB). It is constructed by directly applying the fundamental structure of classical Privacy Knowledge Base (PKB) [263, 264] to quantum contexts, thereby incorporating quantum counterparts of classical elements. Starting from the discussion of the classical PKB elements in [245, 244], we translate them into the quantum domain by carefully investigating and inferring the underlying classical components into the quantum domain. QPKB revolves around five (5) key elements, which are Quantum Hybrid Context, Quantum Bugs and Vulnerabilities, Quantum Privacy by Design Principles, Quantum Privacy Design Strategies, and Quantum Privacy Patterns, aiding in decision-making, system development, and re-engineering processes. The relationship among these elements has been formally presented in Figure 3.21.

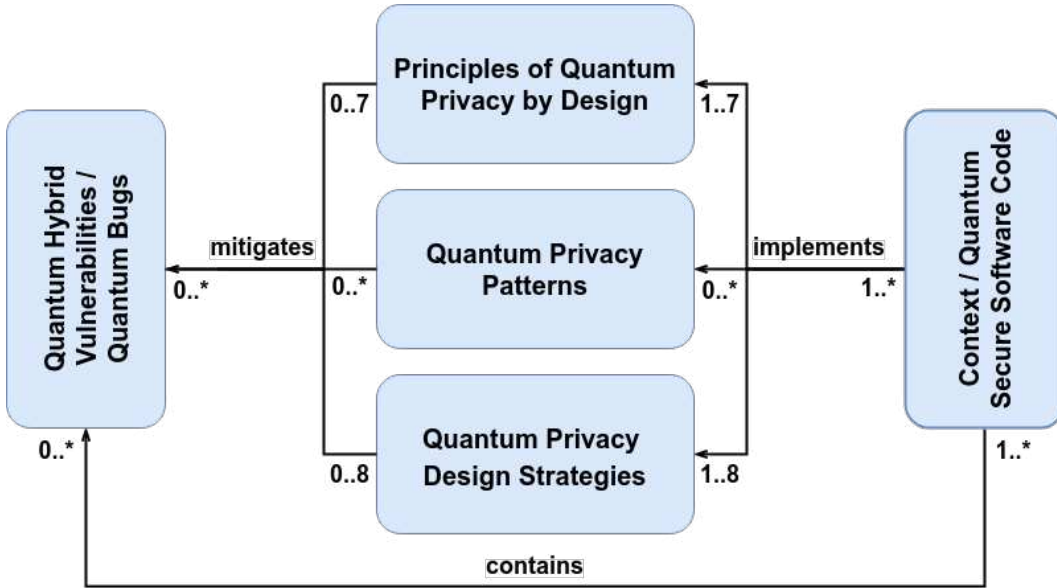


Figure 3.21: Quantum Privacy Knowledge Base Relationship

Quantum Hybrid Context:

At the core of quantum hybrid software development, maintenance, or re-engineering lies the context, serving as the principal element, which is the basis of the framework. It encompasses the below-mentioned key components that apply irrespective of the paradigm of software engineering:

1. *Architectural Requirements*. Define how the information flow is managed across the quantum hybrid system and its components, and define the roles and responsibilities regarding the data.
2. *Use Cases and Scenarios*. Present the elemental interactions within the system, leading to a thorough understanding of system architecture and thereby helping establish a secure software system.
3. *Quantum Hybrid Privacy Enhancing Technologies (QPETs)*. Collection of tools and technologies to safeguard and uphold the privacy and security of sensitive information throughout the quantum hybrid software's life cycle. Quantum computing offers technologies like *Quantum Encryption/Quantum Homomorphic Encryption*, where the user data is secured against eavesdropping and interception by leveraging the principles of quantum mechanics, *Quantum Key Distribution*, which uses qubits (quantum bits) to generate and distribute cryptographic keys securely, and *Quantum Random Number Generator (QRNG)*, which generates true random numbers essential for cryptography [265, 266].

Quantum Bugs and Quantum Vulnerabilities:

A necessary consideration for this element is to accommodate both classical and quantum vulnerabilities. Where vulnerability is defined as the loopholes in the code that an attacker can exploit to gain unauthorized access to the system, it can manifest in different ways in the quantum domain that are not code-dependent. In a hybrid scenario, where both classical and quantum bugs and vulnerabilities are considered, OWASP Top 10:2021 [267], a list of the top 10 vulnerabilities based on industry data and practices, serves as a critical knowledge base. In contrast, quantum bugs and vulnerabilities have not yet been standardized and formalized in the Quantum Software Engineering taxonomy. We rely upon the manually curated commit information from open-source quantum computing projects. Quantum Computing (QC) has been a research topic for decades, but major contributions in the area of Quantum Software Engineering started only in the latter part of the past decade. We know that there is a subtle connection between bugs and vulnerabilities, and errors in codes or bugs can lead to severe vulnerabilities jeopardizing the software system. Hence, analysis of both quantum bugs and vulnerabilities is deemed critical.

Quantum bugs represent anomalies inherent in quantum computing systems, arising from the idiosyncrasies of quantum mechanics. These aberrations encompass challenges such as qubit coherence, entanglement, or measurement inaccuracies, posing significant problems to the stability and reliability of quantum algorithms and computations. In [1], 17 quantum bug patterns were

summarized from among 223 bugs, of which 89 (39%) bugs were purely quantum. Figure 3.22 presents the bugs categorized into different types and patterns. The yellow-colored quantum-specific patterns are explicitly related to QC, whereas the other recurring patterns (blue-colored) are applicable to both paradigms.

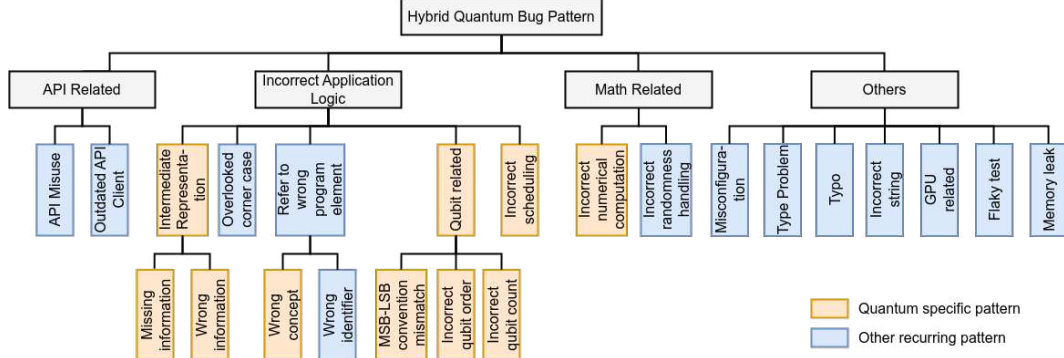


Figure 3.22: Recurring bug patterns in Quantum Hybrid Software Development [1]

Here, we discuss some of the important and recurring bugs in the quantum programs in publicly available repositories [1]:

- **API Related**

- *API misuse*: Passing incorrect parameters or using the wrong API altogether.
- *Outdated API client*: Using expired third-party APIs.

- **Incorrect application logic**

- *Intermediate representation*:
 - *Missing Information*: Error in integrating data into quantum data structures like qubo²⁴ or tape²⁵
 - *Wrong Information*: Failure to add attributes in quantum data structures.
- *Overlooked corner case*: Ignoring rare data handling scenarios such as semi-prime numbers during the development of quantum algorithms for factoring large composites.
- *Refer to Wrong Program Element*:
 - *Wrong concept*: Confusion among different programming elements like qubits and bits.
 - *Wrong identifier*: Mix-up using the same variable names in different contexts.

²⁴<https://support.dwavesys.com/hc/en-us/articles/360003684474-What-Is-a-QUBO>

²⁵<https://docs.pennylane.ai/en/stable/code/api/pennylane.tape.QuantumTape.html>

- *Qubit-related*:
 - *MSB-LSB convention mismatch*: Noncompliance with bit order standards.
 - *Incorrect qubit order*: Inaccurate mapping of qubit indices leading to issues like incorrect gate application and entanglement error.
 - *Incorrect qubit count*: Calculating the wrong qubit count leading to gate operation failure.
- *Incorrect scheduling*: Unwanted execution of quantum instructions at the wrong time.
- ***Math Related Bugs***:
 - *Incorrect numerical computation*: Uses incorrect mathematical calculations to implement quantum events.
 - *Incorrect randomness handling*: Errors in probability and randomness-related code.
- ***Others***:
 - *Misconfiguration*: Configuration parameter errors.
 - *Type problem*: Error in parameter or argument types.
 - *Typo*: General typing errors.
 - *Incorrect string*: Mismatched string literals or variables.
 - *GPU-related*: Memory allocation errors in Graphics Processing Units (GPU).
 - *Flaky test*: Inconsistent or unpredictable test cases.
 - *Memory Leak*: Improper memory management by the program.

Quantum vulnerability describes the susceptibilities found in quantum computing code or setups, which can arise from factors like hardware constraints, algorithmic issues, incorrect mathematical analysis, or security lapses, opening doors for potential attacks. Given their theoretical foundation, most existing quantum algorithms are not inherently vulnerable. Instead, potential attacks or security issues typically stem from physical breaches or implementation issues. Hence, its security is comparable to classical algorithms like Dijkstra's, the Euclidean, or any sorting algorithm that relies solely on mathematical principles for security [67].

The major security concerns for a general quantum computing ecosystem are presented in Figure 3.23 [2] at a high level, providing a comprehensive view of security issues and vulnerabilities summarized below in detail:

- ***Vulnerabilities at the technology level***

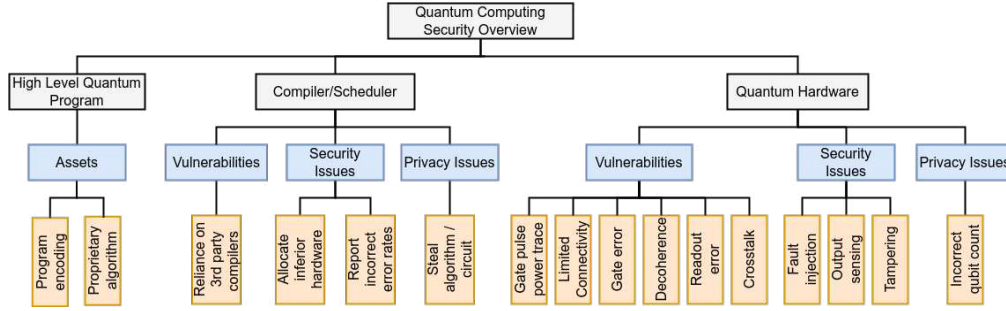


Figure 3.23: Quantum Hybrid Recurring Vulnerabilities/Security Patterns [2]

- Qubit lifetime and errors: *Decoherence* and *dephasing* [268] cause loss of information and phase in qubits, for example, *gate errors* and *readout errors*.
- Vulnerabilities in Trapped Ion (TI) & superconducting qubits: Increased trap vibration energy in TI qubits and crosstalk in superconducting qubits.
- ***Vulnerabilities at the device level***
 - Crosstalk: Unintended information transfer between qubits.
 - Hardware variability: Differing gate error rates and decoherence times due to inter-device differences.
 - Quantum Computer Power Side-Channels: Risks from controller electronics around the quantum core, for example, using RF pulse to execute gate operations, can be rigged or eavesdropped.
- ***Vulnerabilities at the circuit level***
 - Coupling constraint: Limited qubit connectivity impacting gate application.
 - Problem encoding: Translation of classical programs into quantum may expose sensitive information.
- ***Vulnerabilities at the system level***
 - Cloud-based access: Threats from rerouting traffic [219] and information theft.
 - Untrusted compilers & Untrusted hardware providers: Emergence of efficient but untrusted compilers and hardware.
- ***Program agnostic vulnerabilities***
 - Hybrid-classical algorithms: Vulnerable to adversarial attacks and third-party quantum services.

- Quantum machine learning (QML): Vulnerable to circuit performance, for example, *barren plateaus* and input data alterations or temporal quantum-noise variation.
- *Security issues in Quantum Computing*
 - Fault injection: Inducing failures via crosstalk or shuttle-induced fault injection.
 - Program rerouting to lower quality hardware: Redirecting quantum programs to inferior quality hardware.
 - Adversarial input/output manipulation: Insertion of malicious data or manipulation of output data, sometimes without touching the quantum circuit.
- *Privacy issues in Quantum Computing*
 - Readout sensing: Exploiting qubit readout errors to steal user data.
 - Reverse engineering: Exploiting the reversibility to attack the embedded functionality of quantum circuits.
 - Quantum Circuit Identification: Risks from physical and power-based side-channel attacks on cloud-based quantum platforms.

Quantum Privacy by Design Principles:

Quantum Privacy by Design Principles (QPbD) is inspired by the seven(7) foundational principles of Privacy by Design [64]. The principles present a high-level privacy framework for application on programming agnostic platforms, which are based on functionality and scope rather than the paradigm it is built in; thus, they can be safely translated into the Quantum Software Engineering domain. From the considerations about the impacts of quantum computing on classical software development life-cycle in [129], we categorize the QPbD principles in order of impact on Hybrid Software Development and list them below.

- **High Impact:**
 - **Proactive** not Reactive; **Preventive** not Remedial: Predict and address quantum threats by integrating quantum-safe security measures into systems and protocols, ensuring proactive defense against potential quantum attacks.
 - Privacy as the **Default Setting**: Design quantum systems with built-in privacy tools like quantum key distribution and quantum encryption, ensuring that privacy remains default during communication and computation.
 - End-to-End Security – **Full Lifecycle Protection**: Implement thorough security measures throughout the quantum data lifecycle, from generation to destruction, ensuring protection against quantum threats at every step.

- **Medium Impact:**

- Privacy **Embedded** into Design: Embed privacy considerations in quantum system design, ensuring features that protect privacy are foundational, not an afterthought, promoting user trust and security.

- **Low/No Impact:**

- Full Functionality – **Positive-Sum**, not Zero-Sum: Integration of privacy and security in quantum systems to improve capabilities and facilitate positive outcomes, avoiding constraints or compromises.
- **Visibility** and **Transparency** – Keep it **Open**: Promote transparency in quantum privacy and security, ensuring all stakeholders understand the measures in place and trust the reliability of quantum data exchanges.
- **Respect** for User Privacy – Keep it **User-Centric**: Emphasize user privacy in quantum security design, granting users control over data and privacy settings to promote autonomy.

Quantum Privacy Design Strategies:

Privacy Design Strategies serve as a platform-agnostic roadmap for navigating ambiguous legal guidelines and high-level privacy frameworks [269, 270], transforming them into clear design principles that transcend classical and quantum domains and focus on conceptual implementation. Therefore, these strategies can be safely translated into Quantum Privacy Design Strategies (QPDS), where the data-oriented strategies may differ, but process-oriented ones remain consistent. The eight(8) strategies are presented as Data and Process oriented strategies.

- **Data-Oriented Strategies:**

- **Minimize**: Reduce the processing of personal data to the minimum necessary. For instance, quantum secure communication channels can be used to minimize the exchange of classical data.
- **Separate**: Keep personal data processing separate whenever possible. Employing quantum secure multiparty computation, ensuring that personal data remains separate and inaccessible to unauthorized parties.
- **Abstract**: Process personal data with minimal detail, such as performing computations on quantum homomorphic encrypted data without revealing underlying information.
- **Hide**: Protect personal data from being public or known. For example, ensuring that data remains hidden from interception or eavesdropping during transmission using technologies like quantum teleportation.

- **Process-Oriented Strategies:**

- **Inform:** Timely and adequately inform data subjects about personal data processing.
- **Control:** Offer data subjects sufficient control over their personal data processing.
- **Enforce:** Commit to privacy-friendly data processing and enforce it effectively.
- **Demonstrate:** Provide evidence of privacy-friendly data processing practices.

QPDS serves as a strategy that guides developers in effectively distinguishing between process-oriented and data-oriented strategies, which helps break down the application of QPETs into manageable modules. This approach introduces modularity in the privacy-oriented development process, thereby ensuring that the developers tackle complex situations with ease. Therefore, it not only streamlines the development process but also ensures that privacy and security elements are integrated seamlessly and efficiently.

Quantum Privacy Patterns:

In the context of the Privacy Knowledge Base (PKB), privacy patterns enforcing privacy and security are adopted [271, 272] to address recurring privacy and security issues. This is the bedrock of Quantum Privacy Patterns (QPP), which are predefined templates within Quantum hybrid software development that offer standardized solutions for common privacy concerns. They guide decision-making processes regarding data security measures, such as encryption and access control, ensuring compliance with privacy regulations and data protection. By providing reusable solutions to recurring privacy challenges, privacy patterns streamline development efforts and mitigate the risk of privacy breaches. Eventually, they enhance the trustworthiness of quantum software systems while boosting adherence to ethical and regulatory standards in data-handling practices.

Contemporary literature does not contain any QPP that can directly support developers during the Quantum Software Development Life Cycle. Although the literature accounts for bugs and vulnerabilities [2, 1] in quantum hybrid systems, standardized QPPs are still missing. Quantum Bugs and Quantum Vulnerabilities discuss in detail the current state of recurring bugs and vulnerabilities in the quantum ecosystem. Starting from these annotations and definitions of publicly available bugs and vulnerabilities, we can establish QPP that can be followed during the Quantum Software Development Life Cycle (QDLC). For example, in the case of the vulnerability related to **Crosstalk-induced fault injection**, QPP can be represented as shown in Table 3.15.

3.2.5.2 Quantum Privacy-Oriented Software Development

Quantum Privacy-Oriented Software Development (QPOSD) presents a high-level framework for integrating Security and Privacy primitives and imperatives into Quantum Hybrid Software Development. Although security and privacy

Quantum Circuit Integrity in Multi-Tenant Computing Environment
Problem: Crosstalk, unintended and intermittent interaction between two or more qubits, can be exploited to induce failure in the system, causing an advantage to the adversary in a multi-tenant compute (MTC) environment. Solution: The quantum circuit errors should be monitored by <i>test points</i> [273]. It is crucial to know beforehand the expected outcome of the program, which increases the importance of test case selection. Two isomorphic sub-graphs of the device are subject to running two copies of the same circuit with test points to check the relative error rates. The results show unexpected error rates if there is an anomaly. Example: For an un-compute test, the expected output is the initial state of the circuit. When the copies of the circuit are run on the device, if there is an anomaly, the result would show different error rates for each circuit, and the output would not be in the initial state, confirming the claim.

Table 3.15: Quantum Privacy Pattern Template

are different concepts, implementing comprehensive privacy without security leads to data breaches and loss of data integrity. Therefore, QPOSD provides important operational and technical guidelines for incorporating both security and privacy components into the software system. We present the QPOSD framework in Figure 3.24 as a five (5) step sequential process that can be integrated easily into the Quantum Software Engineering (QSE) framework. Considering the current state of QSE, the proposed QPOSD framework can be incorporated into the Waterfall model or AGILE in QSE [274, 123].

The proposed QPOSD framework can be implemented in *forward* and *backward* modes, where the *forward* mode should be used while developing a new software system while in general, the *backward* mode is more aligned with re-engineering existing software systems or implemented with repetitions in the Quantum Agile Development framework [274]. While integrating the elements of Quantum Privacy Knowledge Base, we have presented the framework in *forward* mode in this study. In each step of QPOSD, we have identified the input and output elements, thereby defining the steps and how each step contributes to imbuing security and privacy in Quantum Hybrid Software Development. We use an example to propose the different steps of QPOSD, where we show how security and privacy components relevant to the context are identified and addressed with the help of this framework. Currently, in the Noisy Intermediate-Scale Quantum (NISQ) era, where quantum computers are not fault-tolerant, and the software-based evaluation tools for QSE are not standardized, we rely on open-source and literature-based quality and vulnerability measurement tools.

Analysis

During the analysis phase, we conduct a thorough Quantum Privacy Impact Analysis, incorporating QPbD principles, architectural requirements, and differ-

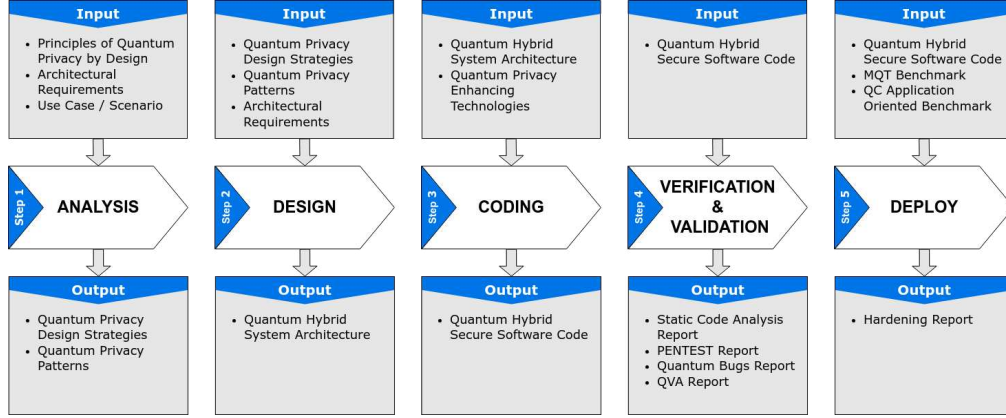


Figure 3.24: Quantum Privacy-Oriented Software Development Model

ent use cases/scenarios in the scope of Quantum Hybrid software development. This comprehensive process involves defining the development context and precisely outlining privacy-critical actors within the quantum software system. Additionally, we pinpoint architectural necessities essential for robust privacy and security measures.

By embedding QPbD Principles, we generate QPDS and QPP that are closely associated with the current context. More specifically, the analysis step investigates the QPbD principles that must be implemented to make the software system/application secure, and the output lists the QPDS applicable to the given scenario and the QPPs that are directly involved in the scenario. These strategic outcomes serve as key frameworks guiding the design phase of quantum software development. They guarantee that privacy and security considerations are recognized and intrinsically implemented by default into the software architecture.

Design

In this phase, the output of the *Analysis* phase, Quantum privacy Design Strategies (QPDS), and Quantum Privacy Patterns (QPP) are used as input elements along with the Architectural Requirements to achieve the Quantum Hybrid System Architecture as output. QPDS entails a higher level of abstraction, avoiding the imposition of specific structures or hard-binding frameworks while still aligning with predetermined objectives and goals. The designer maps the eight QPDS outlined in the subsection 3.2.5.1 to the appropriate areas of the context in order to identify the privacy and security necessities.

QPDS aligns with architectural needs, integrating quantum techniques like quantum homomorphic encryption or anonymization to **minimize** data exposure, maintain data **separation**, conceal or **abstract** personally identifiable information, and ensure data integrity by **hiding** sensitive information. Further, process-oriented strategies are imbibed to implement policies in a timely manner in the quantum hybrid application data flow. This helps the designer to pinpoint the areas in the current context requiring protection Figure 3.25.

The development teams use QPP on the existing architectural requirements to incorporate data protection features during the design phase, where each QPP effectively addresses common recurring issues encountered in existing software systems. These patterns provide tailored solutions to support privacy and security benchmarks, ensuring the development of robust and resilient software applications. The standardization of QPP depends on the maturity of Quantum Software Engineering and requires further investigation into quantum privacy vulnerabilities.

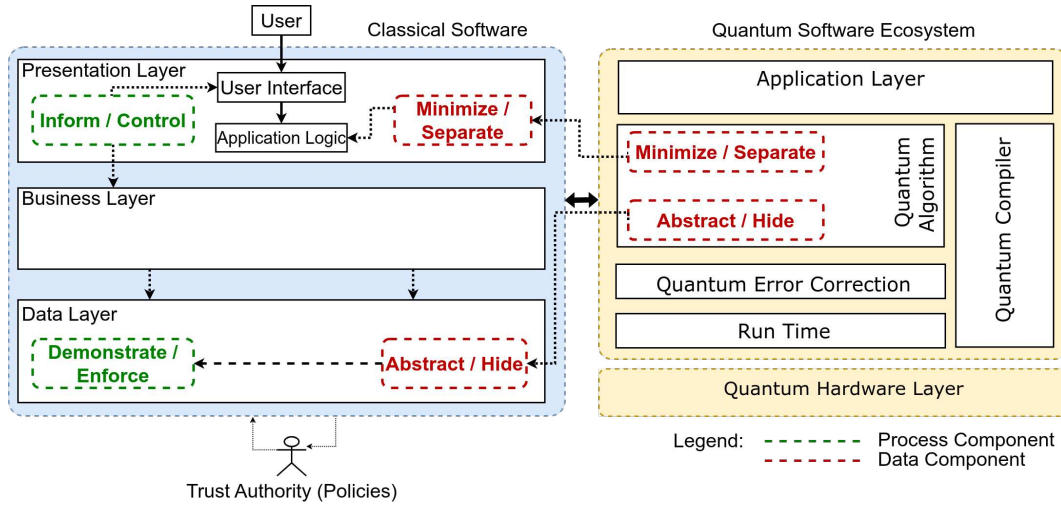


Figure 3.25: Quantum Privacy Design Strategies

Coding

Quantum Privacy Enhancing Technologies (QPET) are crucial tools in preventing identity theft, fraud, and discriminatory profiling in the upcoming Quantum Era. Integrating QPETs with the software system architecture makes it easier to create secure code, ultimately improving data protection. This phase receives Quantum Hybrid System Architecture and QPET as input elements and generates Secure Quantum Hybrid Software Code. The motive of calling the output Secure Quantum Hybrid Software Code is in reference to the Quantum Privacy by Design principle of End-to-End Security, which stipulates that *"without strong security, there can be no privacy"*. In classical software engineering, this step includes the application of various PETs based on the necessary privacy requirements like Anonymization, Encryption, Differential Privacy, Privacy Preserving Data Mining, and Homomorphic Encryption. Quantum Hybrid Software Development includes both classical as well as quantum components; therefore, it encompasses QPETs, like Quantum Key Distribution (QKD), Quantum Encryption (QE), Quantum Random Number Generation (QRNG), and Quantum Walk (QW), among others. Overall, QPET can significantly improve the coding phase by providing developers with tools and resources to enhance security, protect data, ensure compliance, and detect vulnerabilities related to privacy and data protection.

Extending from the design of Quantum Privacy-Oriented Software Development (QPOSD), we see that Quantum Homomorphic Encryption assists in

minimizing the use of sensitive information. Quantum data compression algorithms like Quantum Singular Value Decomposition (QSVD) and Quantum Principal Component Analysis (QPCA) can be used to compress the data while preserving useful information and complementing the minimization of sensitive data. Quantum secure multi-party computation (QMPC) helps computations on encrypted data without the need to reveal the raw data to each party collaboratively, ensuring the segregation of sensitive data. Quantum Differential Privacy (QDP), among others, adds controlled noise to quantum query results, protecting the privacy of individuals while allowing aggregate information to be extracted.

One of the most important pillars of quantum algorithms is the QKD, which uses quantum superposition to design public keys for encryption. QKD can use the QRNG to generate an actual random number and introduce real randomness in the system. The key used for encryption is secure against interception as the malicious listener, if at all, has access to the key, can only decipher a small part of the key, and cannot decipher any valuable information. Alternatively, quantum algorithms like quantum one-time pad (QOTP) or Quantum Vernam Cipher can also be used to encrypt plaintext messages using the shared cryptographic key.

Verification and Validation

During this phase, the Secure Quantum Hybrid Software code is subjected to both static and dynamic code analysis to detect any potential security vulnerabilities that may have arisen during the coding process. Similar to the *Coding* phase, classical and quantum components require diverse tools and techniques for verification and validation of the so-called secure code. For example, Penetration testing is used to ascertain the security of the overall classical system, whereas Quantum Bugs Report [1], or Quantum Vulnerability Analysis report²⁶ can be utilized to determine the quantum resiliency of the quantum hybrid code.

Deploy

The deployment step necessitates a hardening phase of the Secure Quantum Hybrid Software, where the major goal is to flush out any remaining defects and vulnerabilities that have not been caught in the tests until now, such as race conditions, deadlocks, compliance checks, and dependency analysis. The systems are run at full scale to induce failures, sometimes with the help of robot users. Important benchmarks such as CIS Benchmark [275] are used for classical software components, whereas MQT Benchmark, or QC Application Oriented Performance Benchmark are adopted for the quantum components.

3.2.6 Quantum Bug and Vulnerability Analysis in Quantum Software Development

Extending from the previous subsection, we present a methodology, Quantum Error (QError) to identify and analyze quantum software bugs that impact

²⁶www.qcve.org

privacy and security and potentially lead to vulnerabilities. By examining bug patterns and their alignment with Privacy by Design (PbD) and Security by Design (SbD) principles, we emphasize proactive mitigation to enhance robustness. QError enhances the QPOSD framework by focusing on Quantum Bugs and Vulnerabilities, providing insights for effective security guidelines. By analyzing code commits across QDLC phases, we illustrate how privacy and security measures can be seamlessly integrated into quantum software development. We follow the Quantum Development Lifecycle phases as proposed by [52], adopt the *7 Foundational Principles* as proposed by Ann Cavoukian (Table 3.10)[64] for Privacy by Design, and Security by Design principles as in OWASP’s ten principles (Table 3.11)²⁷.

The proposed QError methodology is designed to systematically identify, extract, and analyze code commits that represent actual bug fixes, particularly those related to privacy and security vulnerabilities. The methodology for analyzing quantum bugs in open-source projects can be summarized in the following key steps (Figure 3.26):

²⁷<https://github.com/OWASP/DevGuide/blob/master/02-Design/01-Principles%20of%20Security%20Engineering.md>

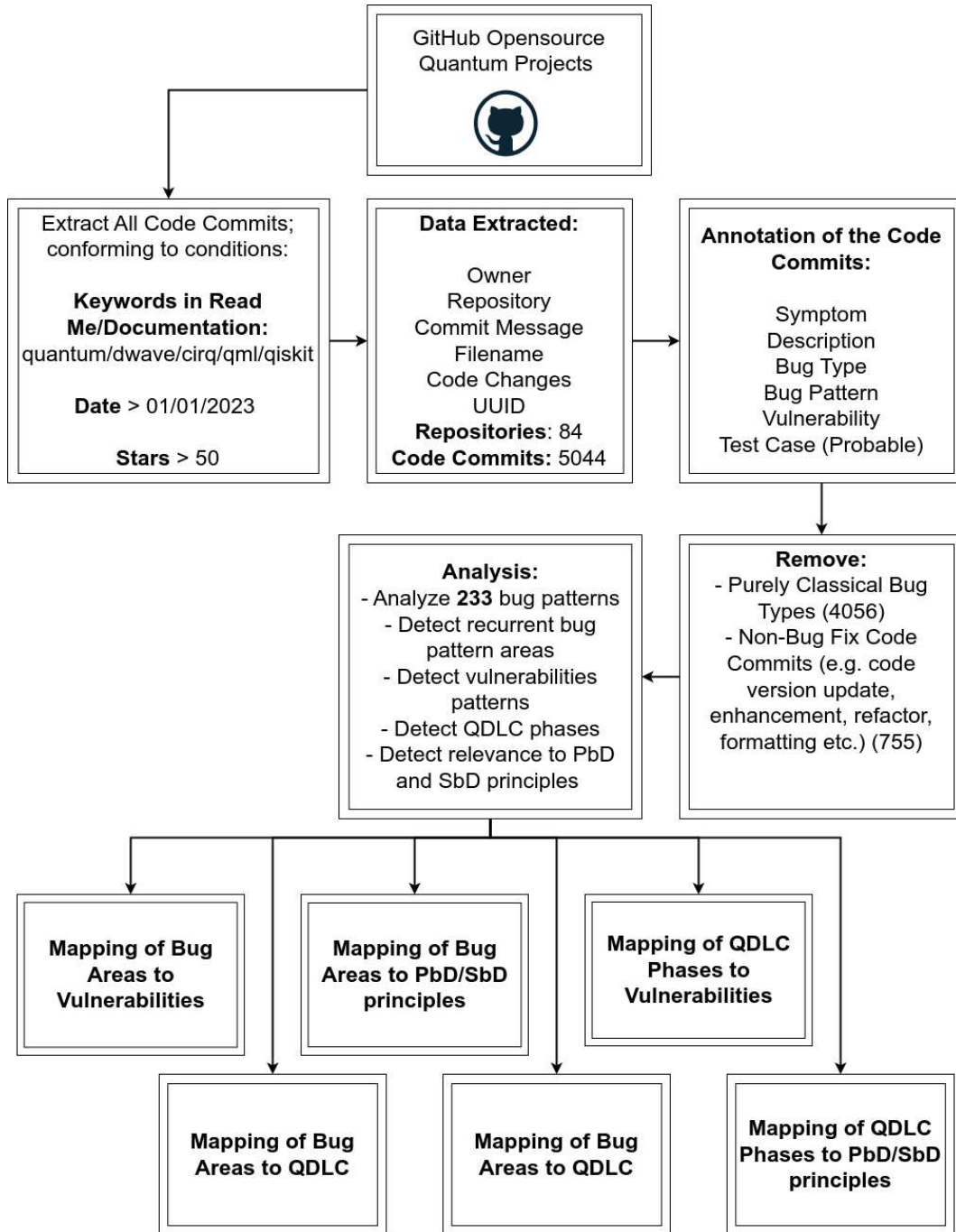


Figure 3.26: Methodology of Code Commit extraction and Analysis

Data & Code Commit Extraction

The data extraction process involved gathering code commits from several open-source quantum software repositories hosted on GitHub. The next step involved identifying and extracting all code commits from open-source quantum projects on GitHub. The projects selected for extraction conformed to the following conditions:

- *Keywords in ReadMe/Documentation*: Repositories were filtered to include only those that explicitly mentioned keywords like "quantum", "dwave", "cirq", "qml", and "qiskit" in their readme or other similar files for documentation, ensuring selection of only quantum related repositories.
- *Date Constraint*: Only commits from repositories updated after January 1, 2023, were considered, guaranteeing that the commits are from active repositories.
- *Popularity Constraint*: Projects with more than 50 stars were selected, indicating a reasonable level of community activity, relevance, and importance.

Data Collection

Thereafter, the data collection step extracted repositories from open-source quantum projects that matched the above criteria. The extracted data records consist of relevant information like *Owner*, *Repository*, *Commit Message*, *Filename*, *Code Change*, and *UUID*(Universal Unique Identifier), which helps to analyze the code commits and annotate the code commits efficiently.

Annotation of Code Commits

Each code commit was then annotated with the following details. The Bug Types were classified into Quantum and Classical.

- *Symptom*: Probable symptom of the problem that led to this code change.
- *Description*: Description of the code change.
- *Bug Type*: The nature of the bug - Quantum or Classical
- *Bug Pattern*: Identification of the underlying bug pattern.
- *Vulnerability*: Vulnerability introduced by the bug, if it exists.

Then, the code commits were filtered to remove *Purely Classical Bug Types*, commits classified as purely classical bug fixes, which were not relevant to quantum-specific analysis and *Non-Bug Fix Commits*, which are not related to bug fixes like code version updates, enhancements, refactoring, and formatting, which were excluded from further analysis.

Code Commit Analysis

The remaining code commits were then subjected to a thorough analysis to detect key bug patterns, QDLC phases, vulnerabilities, and their implications for privacy and security:

- *Recurrent Bug Patterns*: Identification of repeated issues across repositories.
- *Vulnerabilities*: Analysis to determine if bugs led to privacy or security vulnerabilities, identify recurring vulnerability patterns, and assess if similar issues can be resolved with analogous fixes.

- *QDLC Phase and PbD/SbD Relevance*: The commits were mapped to specific phases in the Quantum Software Development Lifecycle (QSDLC) and evaluated for relevance to Privacy by Design (PbD) and Security by Design (SbD) principles.

Mapping and Analysis

Finally, the identified bug patterns, vulnerability patterns, QDLC phases, and their relevance to Privacy by Design (PbD) and Security by Design (SbD) principles were systematically mapped to uncover insights into bug and vulnerability resolution within QDLC. These insights are invaluable for mitigating vulnerabilities proactively throughout the QDLC lifecycle.

- *Mapping Bug Patterns to Vulnerability Patterns*: Identifies how specific bug patterns can lead to potential vulnerabilities.
- *Mapping Bug Patterns to QDLC Phases*: Determines which QDLC phases are affected by specific bugs or identifies the stages where particular bugs originate.
- *Mapping QDLC Phases to Vulnerability Patterns*: Assesses how different phases of the QDLC contribute to various vulnerabilities.
- *Mapping QDLC Phases to PbD/SbD Principles*: Aligns each phase of the QDLC with relevant privacy and security guidelines.
- *Mapping Bug Patterns to PbD/SbD Principles*: Evaluates how bugs relate to Privacy by Design and Security by Design principles.

Tools and Techniques

The study leveraged several tools for data extraction, commit filtering, and bug analysis:

- *GitHub API*: Extracted commit data programmatically from quantum software repositories.
- *Keyword Matching and Filtering Scripts*: Python scripts were used to filter out irrelevant commits.
- *Manual Review*: Each commit was reviewed manually to annotate and confirm its relevance to privacy or security by design principles.
- *Bug Analysis Framework*: A custom framework was employed to analyze bug patterns and their impact on privacy and security.
 - *Bug Pattern Recognition*: Utilized manual and algorithmic approaches to detect and classify recurring bug patterns across code commits, highlighting common sources of errors.
 - *Privacy and Security Evaluation*: Evaluated each identified bug to determine its impact on privacy and security, incorporating Privacy by Design (PbD)(Table 2.1) and Security by Design (SbD)(Table 2.2) principles to assess potential vulnerabilities.

- *Mapping*: Categorized each bug pattern by their impact on various QSDLC phases, providing a comprehensive view of vulnerabilities in the development lifecycle. Sankey diagrams were used to present the mappings and relationships among the elements ²⁸.
- *Heatmap Visualization*: Generated heatmaps to illustrate the frequency of specific bug patterns across different QSDLC phase identifying areas that require stronger privacy and security measures.
- *Manual Review Integration*: Besides automated analysis, manual review plays a crucial role in ensuring the accuracy of bug classification, particularly for subtle or complex bugs that may significantly affect quantum algorithms and hybrid quantum-classical systems.

3.3 Agile in QSE

Agile software development practices is known for their iterative and incremental approach and therefore, have the potential to address many challenges in quantum software development. Agile methodologies promote team collaboration, quick iterations, and continuous delivery, which are beneficial for the evolving nature of QSE [276]. However, the adoption of agile practices in QSE faces several challenges, including a lack of quantum-specific knowledge and skills, the need for quantum-aware tools and technologies, and the difficulty of scaling agile practices sustainably and ethically in the quantum domain [276].

To effectively incorporate agile practices in QSE, there is a need for quantum-specific agile methodologies. These include developing agile toolkits tailored for quantum software, establishing quantum domain-specific expertise, and creating a technical infrastructure dedicated to quantum software development. Presently, QSE processes are often based on hybrid (quantum and classical) concepts. Agile methodologies are being adapted to fit the unique requirements of quantum software development, with a focus on continuous quantum-classical hybridization and the development of quantum-enabled simulators [123]. The integration of agile practices in QSE has significant industrial and regional impacts.

Having said that, there is no dedicated literature regarding the integration of security and privacy in QSE in Agile development methodologies. Although there are emerging discussions around the importance of incorporating security measures from the outset, it remains critical to establish frameworks that effectively address these concerns within agile methodologies. Scrum, the most popular agile methodologies, is well-suited for fostering collaboration among cross-functional teams, enabling rapid iterations and adaptations to evolving security requirements throughout the quantum development lifecycle.

²⁸<https://sankeymatic.com/>

3.3.1 DevSecOps and MLOps enabled Quantum Scrum

We present each of the important concepts in brief here to understand the applicability of these in integrating privacy and security in Quantum Development Lifecycle [52].

Scrum

Scrum²⁹ is an agile project management framework designed to enhance team collaboration and efficiency, particularly in software development. It breaks down complex projects into manageable chunks, known as sprints, which are iterative and incremental work cycles that allow teams to deliver software continuously and adapt to changes quickly [277, 278]. The framework involves key roles: the product owner, who prioritizes tasks; the scrum master, who facilitates the process; and the development team, which executes the tasks [279, 280].

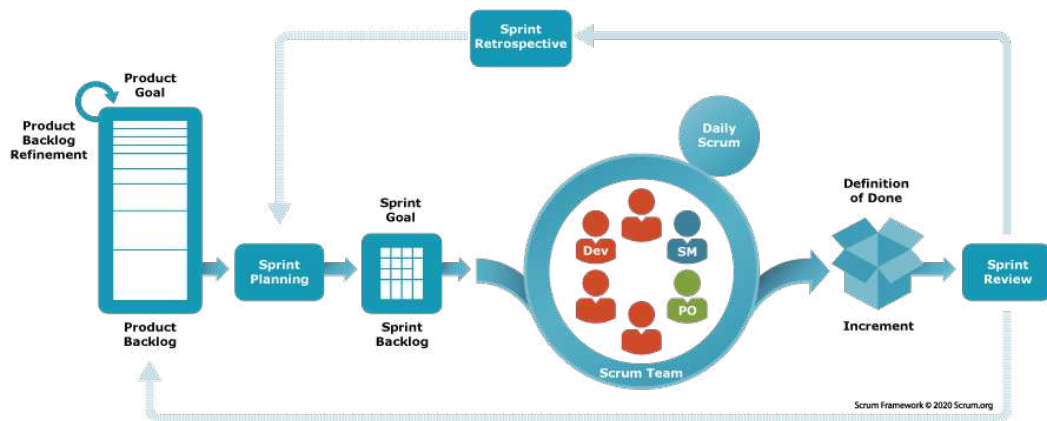


Figure 3.27: Scrum Methodology (From *scrum.org* website)

Scrum's structure includes a product backlog, sprint backlog, and regular meetings such as daily scrums, sprint reviews, and retrospectives to ensure continuous improvement and alignment with project goals [280] (Figure 3.27³⁰). The methodology is widely adopted across various domains beyond software, including education and IT development, due to its ability to handle uncertainty and tight deadlines while fostering a creative and productive work environment [279]. Effective implementation of Scrum requires disciplined adherence to its principles and clear role definitions to avoid overloading responsibilities on individuals, which can enhance team motivation and customer satisfaction [281].

DevSecOps

DevSecOps framework (Figure 3.28³¹) integrates security practices into the

²⁹<https://www.scrum.org>

³⁰<https://www.scrum.org/resources/what-scrum-module>

³¹<https://www.pagerduty.com/blog/devsecops-ops-guide/>

DevOps pipeline, ensuring robust and secure applications without compromising the speed and agility of development processes. This methodology emphasizes breaking down silos between development, security, and operations teams, fostering a culture of collaboration and shared responsibility for security throughout the software development lifecycle [282, 283, 284]. By embedding security measures from the outset, DevSecOps aims to proactively identify and mitigate vulnerabilities, thus reducing the risk of security breaches that traditional DevOps practices might overlook [283, 284]. Automation plays a crucial role in DevSecOps, streamlining security processes through continuous integration and continuous deployment (CI/CD) pipelines and employing tools like SAST and DAST for early vulnerability detection [285].

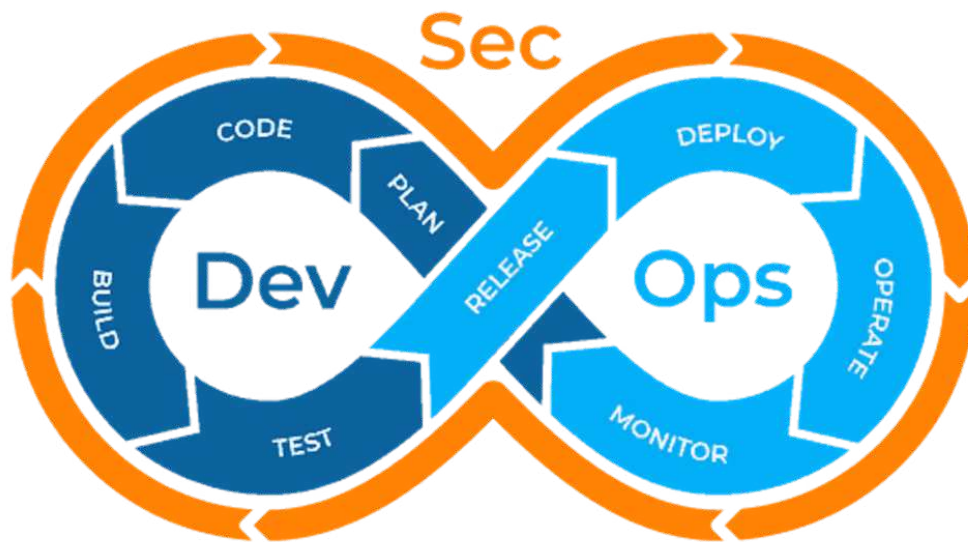


Figure 3.28: DevSecOps Framework (From <https://www.pagerduty.com/blog/devsecops-ops-guide/>)

This approach is particularly beneficial in dynamic environments such as cloud computing, which leverages cloud provider tools to enhance security measures. The adoption of DevSecOps not only accelerates delivery times and improves application quality but also aligns with regulatory compliance and builds customer trust by ensuring a security-first mindset across the organization [284]. As cyber threats become increasingly sophisticated, DevSecOps is paramount in maintaining resilient and secure software systems, making it an essential strategy for quantum software engineering.

MLOps

Machine Learning Operations (MLOps) (Figure 3.29³²) integrates machine learning model development with continuous software engineering processes, similar to DevOps, to streamline AI model deployment and management in production. It addresses challenges like scaling, reliability, and responsible AI

³²<https://ubuntu.com/blog/what-is-mlops>

development [286, 287]. MLOps includes data engineering, model development, CI/CD pipelines, and governance, which optimize the machine learning model lifecycle from data ingestion to monitoring [286, 288]. Implementing MLOps is crucial for organizations to leverage AI effectively, enhancing productivity and innovation while ensuring regulatory compliance. However, MLOps faces challenges such as talent shortages, interoperability issues, and the need for explainable AI (XAI) to improve model trust [286, 289]. By integrating XAI, MLOps can enhance error identification and model accuracy, critical for industrial applications [289]. Overall, MLOps represents the intersection of people, processes, and platforms, providing a structured approach to derive business value from machine learning through robust model deployment and operation [290].

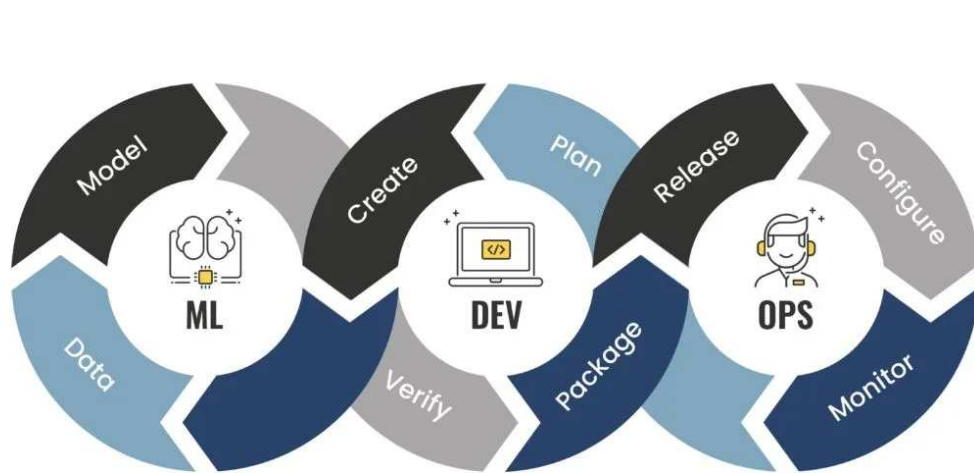


Figure 3.29: MLOps Framework (From <https://ubuntu.com/blog/what-is-mlops>)

3.3.2 Scrum for Iterative Quantum Development

We present an adapted Scrum framework that prioritizes security and privacy considerations throughout the development lifecycle of quantum algorithms and models.

Sprint Planning and Backlog Development

The application of Scrum methodology in the realm of quantum algorithm and model development necessitates a tailored approach to sprint planning and backlog creation. This adaptation involves the integration of security and privacy considerations directly into user stories, ensuring that these critical aspects are prioritized from the outset of the development process [291]. By embedding security and privacy requirements within the backlog, teams can effectively address potential vulnerabilities and privacy concerns throughout

the iterative development cycle.

Daily Standups and Sprint Reviews

Daily standup meetings serve as a vital mechanism for maintaining alignment among the various components of quantum development, including quantum algorithms, machine learning models, and security tasks. These brief, focused gatherings facilitate the sharing of progress related to the implementation of quantum features, particularly those that involve privacy validations [292]. Furthermore, sprint reviews provide an opportunity for teams to collectively assess the outcomes of each sprint, ensuring that all aspects of quantum development are cohesively advancing towards the project goals while adhering to security protocols.

Retrospectives

The retrospective phase of the Scrum process is crucial for evaluating the effectiveness of quantum iterations, especially from a security and privacy perspective. This reflection allows teams to identify areas for improvement in their development practices and to implement strategies that enhance the privacy-oriented aspects of quantum software development [293]. By facilitating a culture of continuous improvement, teams can better adapt to the unique challenges posed by quantum technologies, ensuring that security and privacy remain integral to their development efforts.

Although it is the responsibility of the product owner to maintain the product backlog updated with the security and privacy requirements, it is imperative that the cross-functional team in concern is responsible for updating the product manager about the security and privacy compliancy of the product during sprint planning, daily standups, sprint reviews, and retrospectives. In summary, adopting Scrum for quantum development emphasizes the importance of integrating security and privacy considerations throughout the iterative process. By doing so, development teams can create robust quantum algorithms and models that not only meet performance standards but also safeguard user data and privacy.

Integrating DevSecOps in QSE

Integrating robust security and privacy practices into the Quantum Development Life Cycle (QDLC) emphasizes the incorporation of such measures at every stage of the development process. By embedding security and privacy considerations from the outset and throughout the lifecycle, developers can mitigate potential vulnerabilities that may arise during the quantum algorithm's lifecycle [65].

A critical component of this integration is the automation of security testing for quantum algorithms, circuits and gates. As quantum computing systems become increasingly complex, it is imperative to establish automated testing mechanisms capable of identifying vulnerabilities in quantum software. This includes assessing hybrid integrations that combine classical and quantum

computing elements, which can introduce unique security challenges [53].

Furthermore, establishing Continuous Integration/Continuous Deployment (CI/CD) pipelines is critical for the timely detection and mitigation of security and privacy vulnerabilities in quantum software. These pipelines should be configured to perform automated quantum security assessments, enabling developers to identify and address weaknesses in real-time before the deployment of quantum/hybrid softwares. Tools designed for automating noise resilience testing and validating quantum state security are essential for ensuring that quantum systems can withstand external interferences and maintain their integrity during computations.

Continuous monitoring and feedback mechanisms play a pivotal role in safeguarding quantum systems against evolving threats. Regular testing for side-channel attacks, decoherence, and entanglement vulnerabilities is necessary to maintain the security posture of quantum applications. Additionally, practices derived from DevSecOps can be employed to ensure data integrity throughout quantum computations, thereby enhancing the overall security framework of quantum software development.

In conclusion, the integration of security and privacy measures within the QDLC through SbD and PbD principles, automated testing, CI/CD pipelines, and continuous monitoring is crucial for addressing the security challenges posed by quantum computing. By adopting these practices, developers can create resilient quantum algorithms that are better equipped to withstand potential threats.

Integrating MLOps in QSE

Integrating Machine Learning Operations (MLOps) within Quantum Software Engineering (QSE) enhances the capabilities of quantum computing through the application of hybrid quantum-classical models. Therefore, contextually, machine learning algorithms facilitates various quantum computing tasks, like quantum-enhanced optimization, quantum feature mapping, and the implementation of variational quantum circuits.

One of the primary functions of MLOps in this domain is the management of the data lifecycle associated with quantum-generated data. This encompasses the automation of processes related to the collection, versioning, and monitoring of data, which is crucial for maintaining the integrity and usability of quantum datasets. Furthermore, MLOps ensures that privacy concerns are addressed during the training of machine learning models, especially when dealing with sensitive quantum data. This aspect is vital, as the unique nature of quantum data can pose significant challenges to conventional privacy-preserving techniques.

Additionally, automated deployment of machine learning models is a critical component of MLOps in quantum workflows, which integrates ML models into quantum computing processes through the implementation of automated

Continuous Integration/Continuous Deployment (CI/CD) pipelines. Such automation not only enhances efficiency but also reduces the potential for human error during model deployment.

Moreover, inclusion of security testing of machine learning models is essential to safeguard against adversarial attacks, particularly in hybrid quantum systems where vulnerabilities may be exploited. The proactive identification and mitigation of these risks would be crucial for maintaining the robustness and reliability of quantum-enhanced machine learning applications. For example, if a quantum classifier is being used to analyze encrypted data, DevSecOps practices ensure that data privacy remains intact while the model's robustness is verified through MLOps processes.

In summary, the convergence of MLOps and quantum computing offers a transformative approach to leveraging machine learning in the quantum realm, with significant implications for data management, model deployment, and security. This integration not only advances the field of quantum computing but also opens new avenues for research and application in various domains.

3.3.3 Unified Framework for Agile Privacy and Security in QSE

We present a unified framework combining DevSecOps, MLOps, and SCRUM for Quantum Software Engineering, which exploits the principle ideas of the aforementioned frameworks and proposes a framework that continuously integrates security and privacy requirements throughout the lifecycle of the application.

Figure 3.30 presents a high-level representation of the unified framework that shows how Agile principles, particularly Scrum, can be adapted to integrate DevSecOps and MLOps ideas into Quantum Software Engineering (QSE). The process begins with a comprehensive Product Backlog that explicitly includes security, privacy, and machine learning (ML) requirements alongside other functional tasks. Security concerns, such as threat modeling or compliance requirements, and ML-specific needs like model retraining or hyperparameter tuning, are treated as first-class priorities, ensuring their iterative development throughout the lifecycle.

During Sprint Planning, the team collaboratively selects items from the backlog to create the Sprint Backlog, defining clear sprint goals. This step emphasizes cross-functional collaboration among quantum developers, security experts, and ML engineers, enabling the seamless integration of quantum and classical components. As the Sprint progresses, the Cross-Functional Scrum Team works to deliver incremental improvements, embedding security features and refining ML models while maintaining the robustness of the hybrid quantum-classical system.

Once tasks are completed, the Sprint Increment undergoes rigorous validation through security testing, ML performance evaluations, and integration testing,

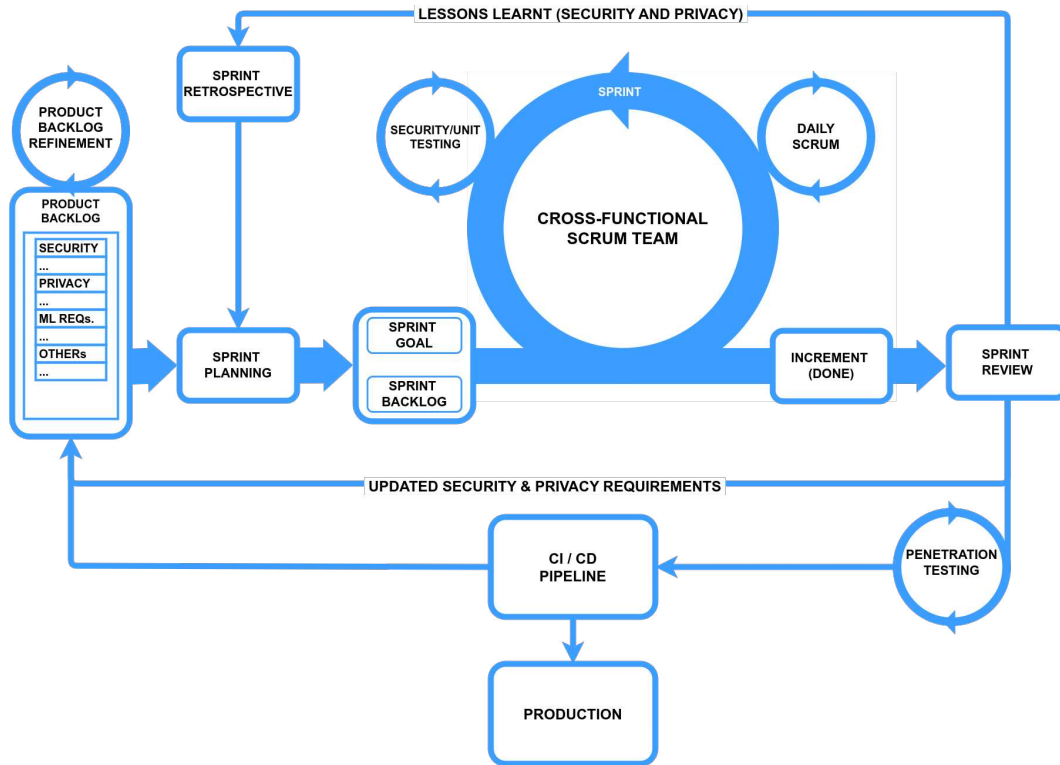


Figure 3.30: Unified Framework

ensuring that both quantum and classical components work cohesively. During the Sprint Review, stakeholders evaluate the deliverables, providing feedback on the implemented security measures and the performance of ML pipelines. This feedback informs the Sprint Retrospective, where the team reflects on successes and areas for improvement, including any overlooked vulnerabilities or inefficiencies in ML workflows. These insights feed back into the Product Backlog, refining future iterations.

The process seamlessly integrates into a CI/CD Pipeline, automating the deployment and validation of quantum applications. Security testing tools and ML model monitoring ensure that each deployment is robust and reproducible. Before production, the increment undergoes Penetration Testing to verify its security posture in real-world scenarios. Finally, the increment is deployed to Production, where runtime monitoring safeguards security and ML performance over time. Feedback from production use cycles back into the development process, ensuring continuous improvement in security, privacy, and ML integration.

This cyclical approach ensures that Agile principles adapt effectively to the unique needs of QSE, embedding security and privacy by design while leveraging MLOps to address the complexities of hybrid quantum-classical systems. Through iterative refinement, this framework fosters resilient, high-quality quantum software development

Shifting Security and Privacy Left

The contemporary approach to quantum software development is the traditional waterfall model, which is particularly inefficient in the domain of QSE as redesigning and redoing quantum software are extremely cost and resource-intensive. Therefore, it is necessary to incorporate security and privacy measures from the initial phases of the Quantum Development Life Cycle (QDLC). This proactive methodology, often referred to as "shifting left," emphasizes the importance of embedding security checks and privacy considerations into the developmental framework right from the onset. A principal aspect of this approach is the application of SbD and PbD principles, which advocate for the integration of security privacy considerations into the architecture of quantum data encoding and processing systems. To effectively manage security and privacy requirements, it is essential to update the product backlog at the commencement of the project and to continue this practice iteratively throughout the development process. This ensures that evolving security and privacy concerns are consistently addressed. Furthermore, security testing should be an integral part of the unit testing phase during each development sprint. This collaborative effort between developers and security teams fosters a culture of shared responsibility for security, thereby enhancing the overall robustness of the software.

Unified CI/CD Pipeline for Quantum, ML, and Security

A unified CI/CD pipeline represents a best practice for the integration of quantum software, machine learning (ML) components, and security protocols. This holistic approach ensures that security checks are seamlessly incorporated throughout the development lifecycle. Tools such as Qiskit, PennyLane, and TensorFlow serve as exemplary resources for implementing these practices, enabling teams to automate security testing within the CI/CD framework. With the automation of these security assessments, organizations can effectively identify vulnerabilities early in the development process, thereby mitigating potential risks before they escalate. For example, if a quantum machine learning model is used for quantum data classification, the model's updates and enhancements (e.g., retraining with new data) can be automatically validated within the pipeline without the need for manual intervention, consistent with Scrum's iterative approach and DevSecOps checks for security validation.

Threat Modelling and Risk Analysis

The identification and mitigation of privacy and security vulnerabilities are crucial in the context of hybrid quantum-machine learning models. Conducting thorough threat modeling exercises allows teams to systematically analyze potential security threats to both quantum and classical components of the system. This proactive analysis not only aids in recognizing vulnerabilities but also facilitates the development of a comprehensive risk assessment framework. To maintain an agile response to emerging threats, it is imperative to regularly update the product backlog with revised threat models and associated risks.

This iterative approach ensures that the development team remains vigilant and responsive to new security challenges, thereby fostering a resilient and secure software environment. By embedding these practices into the QDLC, organizations can better safeguard their quantum applications against evolving security threats.

The frameworks and models discussed in this section establish structured approaches for embedding security and privacy into quantum software engineering. These methodologies aim to strengthen resilience against evolving threats and enhance secure development practices. However, their practical implementation and effectiveness require thorough testing and validation. The next section focuses on the experimental setup, evaluation techniques, and results, providing a detailed analysis of how these models perform in real-world quantum software development scenarios.

Chapter 4

Experiments and Results

This chapter presents the experimental evaluation, results, and discussions on the proposed models and frameworks, assessing their effectiveness in addressing security and privacy challenges in quantum software engineering. The analysis is structured around key areas where specific models or frameworks have been introduced, ensuring a focused assessment of their impact. Systematic experimentation, validation techniques, and comparative analysis provide insights into their strengths and limitations, guiding potential refinements for practical adoption. In line with the goals outlined in section 1.2, this chapter aims to empirically evaluate the proposed models and their applicability in real-world quantum software development.

The chapter is organized into different domains where quantum technologies were applied as validation tools for integrating security and privacy into quantum software engineering. These include the automotive sector and smart cities where the proposed frameworks were assessed in distinct real-world contexts. Each domain highlights how quantum-enhanced methodologies contribute to securing software development processes and mitigating potential threats.

Beyond these sector-specific validations, the chapter also explores case studies focusing on privacy-oriented quantum software development, providing concrete examples of how privacy principles are embedded into quantum systems. Additionally, it examines quantum software bugs and vulnerabilities, analyzing their implications in Quantum Software Development. These discussions contribute to a broader understanding of the challenges and solutions in secure quantum software engineering, reinforcing the need for proactive security and privacy integration. We take a similar approach as before and present the experiments, results and the corresponding discussions related to all the sectors and areas mentioned earlier.

4.1 Automotive Sector

To evaluate the effectiveness of quantum-based intrusion detection in the automotive sector, a series of experiments were conducted comparing a hybrid quantum-classical approach with traditional machine learning models.

4.1.1 Quantum-based Intrusion Detection System

The initial study aimed to assess the applicability of quantum annealing for detecting malicious activity on the CAN bus, a critical component of in-vehicle communication. Using QBoost, a quantum ensemble learning algorithm, and a classical Support Vector Machine (SVM) implementation, the experiments analyzed their performance on two distinct datasets—DoS and Fuzzy attacks. The IDS pipeline employed in this study facilitated data preprocessing, feature extraction, model training, and prediction, ensuring a structured evaluation. The following present the experimental setup, results, and comparative analysis of these models, highlighting the advantages and challenges of integrating quantum computing into automotive cybersecurity.

We evaluate the performance of classifiers proposed in this sector by utilizing the state-of-the-art Car-Hacking Dataset [4, 5] for intrusion detection. This dataset was created by logging CAN traffic through the OBD-II port of a real vehicle, specifically a Hyundai YF Sonata. Two custom Raspberry Pi devices were employed: one for recording the traffic from the OBD-II port and another for injecting fabricated messages to simulate an attack node.

The dataset is divided into five sub-datasets based on the type of attack: (i) DoS, (ii) Fuzzy, (iii) Gear Spoofing, (iv) RPM Spoofing, and (v) Attack-Free. Each subset contains 300 instances of intrusions caused by injected messages. These intrusions were executed for durations ranging from 3 to 5 seconds, while the CAN traffic for each subset was logged over a period of 30 to 40 minutes [239]. The Car-Hacking dataset is structured as follows:

1. **DoS Attack:** Messages with ID "0x0000" are injected every 0.3 milliseconds.
2. **Fuzzy Attack:** Random IDs and payloads are injected every 0.5 milliseconds.
3. **Gear Spoofing Attack:** Messages corresponding to certain CAN IDs related to gear information are injected every one millisecond.
4. **RPM Spoofing Attack:** Messages corresponding to certain CAN IDs related to RPM information are injected every one millisecond.
5. **Attack-free:** Normal CAN traffic without any injected attack messages.

The structure of the messages within the dataset comprises the following five attributes:

- *Timestamp:* The time (in seconds) when the message was recorded.

- *CAN ID*: The identifier of the message in HEX format.
- *DLC*: The number of data bytes (ranging from 0 to 8) sent on the CAN bus.
- *DATA[0-7]*: The byte values expressed in HEX format.
- *Flag*: Indicates whether the message is injected (T) or normal (R).

Attack Type	Total number of messages	Number normal messages	Number of injected messages
DoS attacks	3,665,771	3,078,250	587,521
Fuzzy Attacks	3,838,860	3,347,013	491,847
Gear Spoofing	4,443,142	3,845,890	597,252
RPM Spoofing	4,621,702	3,966,805	654,897
Normal messages	988,871	988,871	-

Table 4.1: Car-Hacking Dataset [4, 5] overview for Intrusion Detection.

Table 4.1 provides a summary of the distribution across the four distinct subsets. Each row represents a subset associated with a particular type of attack, except for the subset containing normal messages, which does not include any attack-related data.

Data Pre-processing & Experiments

For this study, we used two sub-datasets: **DoS** and **Fuzzy**. A DoS attack comprises injection messages with a dominant arbitration ID (0000) and a null Data payload. The messages were injected every 0.3 milliseconds. This attack could flood the CAN bus because, in the arbitration phase, the dominant bit has a higher priority than the recessive bit (1-bit). Instead, in a fuzzy attack, spoofed random CAN ID and data values are injected every 0.5 milliseconds, which causes unusual behavior in the vehicle. We applied One-Hot-Encoding¹ on *CanId* and binary encoding² on *Data*. The one-hot-encoding technique creates a new binary variable for each unique category, extracting N features out of N categories, where 0 and 1 indicate the absence or presence of that category in each row. The total number of features was 43, including *Frequency*.

The Fuzzy database has a more complex structure than the DoS since malicious messages are randomly sent to the CAN bus using all the 2017 unique CanIds, 37 of which also transmit normal traffic. Binary encoding was applied to both *CanId* and *Data*, which returned 11 and 20 features, respectively, making 32 features in total with *Frequency*. The classifiers trained to detect a specific type of attack perform better than generic ones. Therefore, the two binary classifiers,

¹<https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html>

²https://contrib.scikit-learn.org/category_encoders/binary.html

SVC and QBoost, were trained for each dataset. The datasets were split into 80% training and 20% testing sets before training. To reduce bias and balance the ratio of CanIds between the two splits [228] [294], every dataset was shuffled.

DoS Dataset Results

The results of two binary classifiers, QBoost and SVC, are shown in Table 4.4. Both classifiers could distinguish between malicious and non-malicious messages, reaching 100% Accuracy, Precision, Recall, and F1-score in each of the 10 rounds of testing. As data shuffling before every round didn't affect the metrics, and the training and testing set metrics were always the same, we might safely assume these to be good indicators of low overfitting. The confusion matrices of SVC and QBoost in Figure 4.1 show that each attack (label 0) and normal message (label 1) in the testing set was correctly classified. The confusion matrices shown in Figure 4.1 refer to the results from the last run of the test set.

QBoost showed a considerable improvement in the execution times. Table 4.2 compares the performance in seconds between the two classifiers. We can see that we have a high gain in execution time in both the training and prediction phases, with gains of 62% and 99%, respectively.

Table 4.2: Average time performance of the Binary classifiers on DoS dataset. Grey shows the best performance

Classifier	Average training time (s)	Average prediction time (s)
SVC	68.6735	2.5609
QBoost	26.1415	0.0275
	-42.532	-2.5334

Table 4.3: Average time performance of the Binary classifiers on Fuzzy dataset. Gray shows best performance

Classifier	Average training time (s)	Average training time (h)	Average prediction time (s)	Average prediction time (m)
SVC	5933.2840	1.6481	431.4609	7.191
QBoost	25.7134	0.0071	0.1462	0.0024
	- 5907.5705	-1.641	- 431.3146	-7.1886

Fuzzy Dataset Results

The Fuzzy database has a more complex structure compared to the DoS since malicious messages are randomly sent to the CAN bus using all the 2017 unique CanIds, 37 of which also transmit normal traffic. Binary encoding was applied to both *CanId* and *Data*, which returned 11 and 20 features, respectively, making 32 features in total with *Frequency*. The results of two binary classifiers are shown in Table 4.4. Considering the recall metric, we can see that the

SVC - DoS Dataset		Qboost - DoS Dataset	
0	TN 16.16%	FP 0%	
1	FN 0%	TP 83.84%	
	0	1	
0	TN 12.27%	FP 0.85%	
1	FN 0.13%	TP 86.74%	
	0	1	
SVC - Fuzzy Dataset		Qboost - Fuzzy Dataset	
0	TN 12.92%	FP 0.2%	
1	FN 1.59%	TP 85.29%	
	0	1	

Figure 4.1: SVC and QBoost Confusion matrix on DoS & Fuzzy datasets. "0" malicious; "1" normal messages. TN-True Negative, FP-False Positive, FN-False Negative, TP-True Positive

SVC performed slightly better than QBoost. From the viewpoint of time performance, Table 4.3 shows the average results obtained from 10 runs, and it can be seen that QBoost clearly outperformed SVC in terms of execution time during both training and prediction phases, with 99% gain in both cases. The confusion matrices for SVC and QBoost shown in Figure 4.1 confirm the precision and recall metrics presented in Table 4.4. For the fuzzy attack, the confusion matrices also refer to the last run of the testing set. The training time for SVC was closer to that of QBoost for DoS dataset results. On the other hand, with the Fuzzy set being more complex in comparison to DoS, the acquired results prove that the quantum annealing-based algorithm successfully managed to optimize a complex problem.

Concluding, inclusion of quantum technologies in this study shows impressive improvements in time performance while maintaining similar quality for CAN bus attack classification.

4.1.2 Quantum Powered Developer Support for Automotive Security

The subsequent study in the same domain evaluates the performance of traditional and quantum machine learning algorithms in detecting cyber threats within automotive systems. Adaptive Boosting (AdaBoost) and Random Forest (RF) were compared against the quantum-based QBoost algorithm for attack detection on the CAN bus. Using the Car-Hacking Dataset, the proposed methodology classified various attack types and mapped them to a

(a) Binary classifier performance: Train & Test sets for **DoS** dataset.

Set Type	Classifier	Accuracy	Precision	Recall	F1-score
Train	SVC	1.0	1.0	1.0	1.0
	QBoost	1.0	1.0	1.0	1.0
Test	SVC	1.0	1.0	1.0	1.0
	QBoost	1.0	1.0	1.0	1.0

(b) Binary classifier performance: Train & Test sets for **Fuzzy** dataset. (Grey shows best)

Set Type	Classifier	Accuracy	Precision	Recall	F1-score
Train	SVC	0.9858	0.9945	0.9891	0.9918
	QBoost	0.9778	0.9988	0.9757	0.9871
Test	SVC	0.9716	0.9785	0.9892	0.9838
	QBoost	0.9756	0.9959	0.9760	0.9858

Table 4.4: Performance of Binary Classifiers

comprehensive taxonomy, aiding developers in mitigating vulnerabilities. Multi-class classification was implemented using the One-vs-Rest (OvR) strategy for QBoost. The following sections present the experimental setup, evaluation metrics, and results, demonstrating the role of quantum computing in enhancing automotive cybersecurity.

Data Pre-processing & Experiments

The experiments are carried out using an Intel(R) Core(TM) i7-1065G7, 1.30GHz Processor, and 12 GB RAM for AdaBoost [295] and RF [296]. Instead, for QBoost [47, 297], the program runs from the same system as mentioned before, but the QML part runs on the D-Wave Leap Quantum Cloud (DLQC) platform. AdaBoost or Adaptive Boosting [295] is a boosting technique that combines multiple weak classifiers to build one strong classifier. It is an ensemble learner where multiple weak classifiers are grouped with each other, progressively learning from past misclassifications to generate strong models. Random Forest [296] is also based on another ensemble learning called bagging. It consists of a large number of individual decision trees that operate as an ensemble, where each tree predicts a class, and the most voted one becomes the model's prediction.

To evaluate multiclass classifications, the Car-Hacking Dataset [238, 239] was used. The dataset was pre-processed by transforming the CAN ID and DATA from hexadecimal to decimal. The timestamp and DLC were removed as they did not contribute to the classification. A random under-sampling for the majority class was performed to account for the unbalanced dataXPERIMENTSet. The features were scaled using the standard deviation (MinMaxScaler³), and the Principal Component Analysis (PCA) was used to reduce feature dimensionality using four components. Finally, the dataset was split into 70% training and 30% testing parts. Considering Figure 3.14, the attack labels in the "Attack

³<https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html>

Prediction" phase are four (DoS, Fuzzy, Spoofing Gear, and Spoofing RPM) because the Car-Hacking Dataset contains typology for four attacks.

Results

It should be noted that where AdaBoost and RF are multi-class classifiers, QBoost is inherently a binary classifier. As mentioned before One-vs-Rest (OvR) methodology was adopted to transform it into a multi-class classifier, where one class is fitted against all the other classes to determine its efficiency. After the transformation of QBoost using OvR, it functions exactly similarly to any other multi-class classification algorithm. Therefore, we present the aforementioned quality metrics for each of the models. We also present the time efficiency of all those models to study the training and testing times required by each of them. The quality metrics of all the models are almost similar, with both AdaBoost and RF showing the same measures, whereas QBoost performed marginally lower, differing only in the thousandth place. Figure 4.2 clearly shows the quality metrics for the three ML algorithms on the Car-Hacking dataset. Although AdaBoost and RF perform slightly better than QBoost, the quality metrics for all these ML algorithms are very much similar. The accuracy scores for QBoost, RF, and AdaBoost are 0.994, 0.999, and 0.999, respectively. The scores for the other metrics are almost similar, for example, 0.997, 0.999, and 0.999 for Precision. We can clearly observe that the scores are very similar.

Figure 4.3 presents the time efficiency for training and prediction time for all three machine learning algorithms. AdaBoost outperforms both RF and QBoost in training and prediction times. AdaBoost, in general, outperforms RF since it consists of a forest of trees stumps rather than full-sized trees. Again, QBoost, inherently being a binary classifier, may not achieve the same effectiveness when it is implemented as a multi-class classifier.

In the mapping phase, after the classification of the attack typology as **DoS**, the result is mapped with level one of the Sommer et al. taxonomy [240]. This mapping, which is based on **Attack Class**, filters out other attack types and returns the list of different types of DoS attacks (12 DoS attacks). Level two of the taxonomy provides the probable attack types related to the identified attack, which are Distributed DoS and Jamming. Based on the **Attack Consequence**, which is **Engine doesn't start**, we pinpoint the exact attack to the description. From level three of the taxonomy, we get the attack term of the publication, and thus, we extract the Vulnerability (Improper Adherence to Coding Standards, Violation of the CAN standard), Attack Path (Access to CAN via OBD port, Continuously sending CAN messages with high priority), Interface (OBD), and Exploitability (CVSS 2.84). The **CWE-ID** provides a further mapping to the global Common Weakness Enumeration (CWE)⁴ scores, allowing detailed information on the vulnerability discovered. This generates guidelines for the developer, which suggests the developer to: **1)**

⁴<https://cwe.mitre.org/data/definitions/710.html>

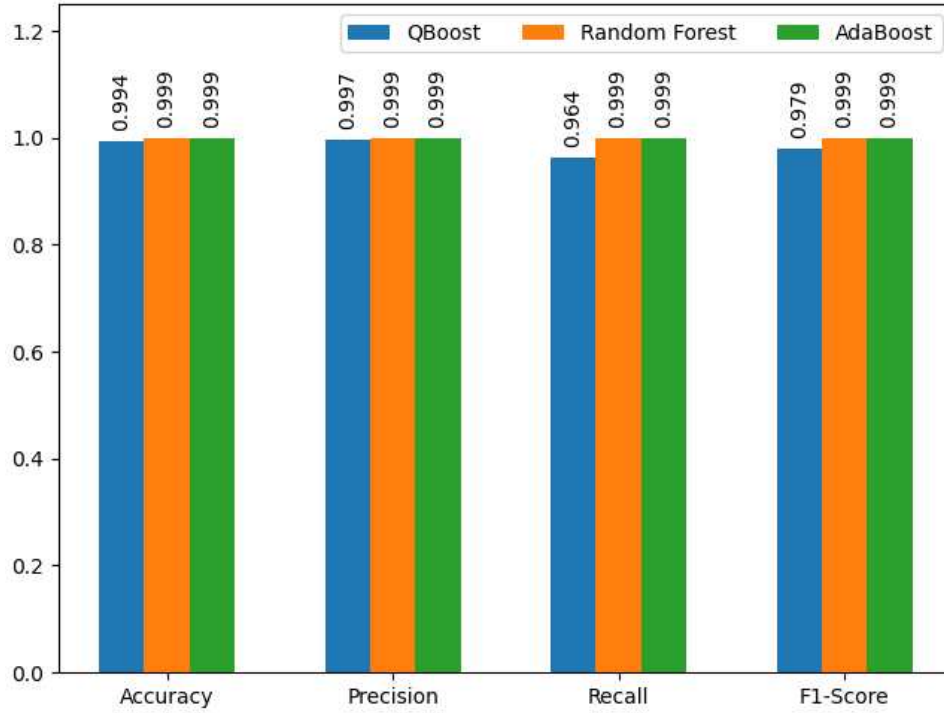


Figure 4.2: Quality of the ML models.

document the implementation and 2) closely follow coding standards during **Architecture and Design**, and **Development** phases. Additionally, it also suggests the **usage of automated tools to enforce the standards** during both the testing and implementation phases.

Summarizing the results above, the study presents clearly that quantum technologies can be used to boost security concerns in sectors like automotive, if implemented carefully, which however depends on the performance of quantum systems. Having said that, the results prove that quantum technologies perform with similar quality but with limited versatility, until now.

4.1.3 Quantum Artificial Intelligence in Extending Developer Support

This study evaluates the SeQuADE (Secure Quantum Automotive Development and Engineering) methodology Figure 3.16, which integrates quantum-based intrusion detection with a structured developer support system. Using QBoost for attack classification on the CAN bus and an extended taxonomy derived from Sommer et al. [240], the methodology maps detected attacks to relevant vulnerabilities and remediation strategies. A natural language processing (NLP) model retrieves additional vulnerability information from the NVD to support developers in mitigating security risks.

To evaluate QBoost in a multi-classification problem with the OvR technique, the state-of-the-art dataset called Car-Hacking Dataset [238, 239] was used. The dataset was pre-processed similarly to the initial study of the automotive domain.

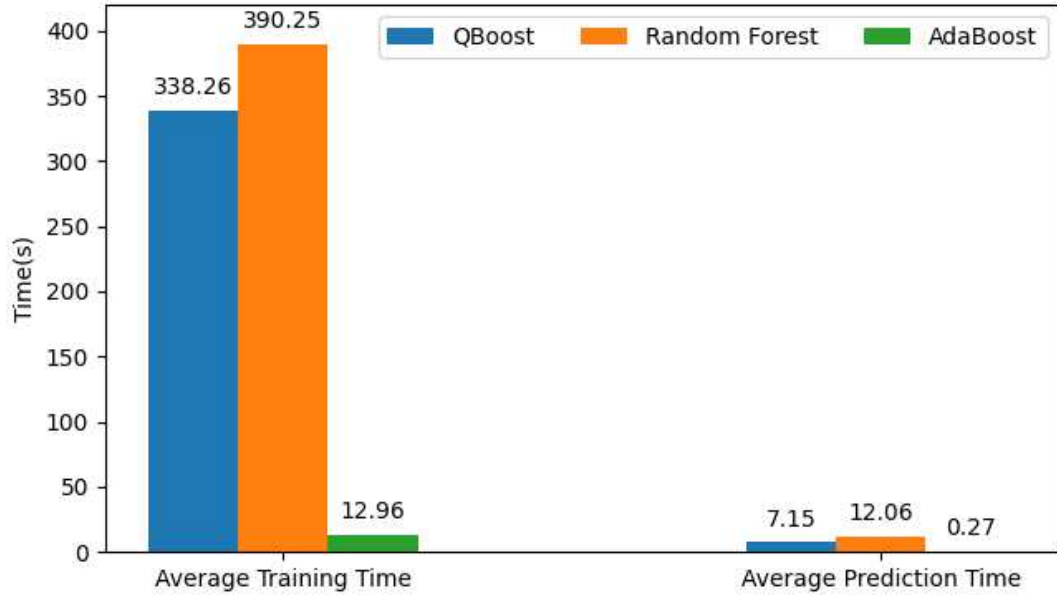


Figure 4.3: Time Efficiency of ML Classifiers.

The CAN ID and the DATA attributes were transformed from hexadecimal to decimal, and the timestamp and the DLC were removed because they did not contribute to the classification. Thereafter, a random under-sampling was performed for the majority class to balance the dataset. After the pre-processing phase, the features are scaled using the Min-Max scaler into a range of $[0,1]$. Then, the Principal Component Analysis (PCA) was used using four components. Finally, the dataset was split into 70% and 30% for training and testing data, respectively.

Finally, the DoS, Fuzzy, Spoofing RPM, and Spoofing Gear were concatenated to perform a multi-classification. Since QBoost performs only a binary classification, the OvR technique was used, where one class is fitted against the others. Since the numbers of the classes are four (Normal, DoS, Fuzzy, Spoofing Gear, Spoofing RPM), four QBoost classifiers are used. Then, to obtain a multi-classification, each calculates a score, and the QBoost model that obtained the best score was taken for the specific class. In the fitting phase, for each model, 35 weak classifiers were used with a tree depth of 3 and a lambda factor of 0.5. It is important to note that QBoost implements a Decision Tree as a weak classifier, so, in this case, 35 Decision Trees were fitted. For the Binary Quadratic Model, the following parameters were used *num_reads*: 3000, *auto_scale*: True, and *num_spin_reversal_transforms*: 10.

NLP Model

From NVD, 86 automotive vulnerabilities were obtained using the scraping process. In addition, the CVE ID, the description, references, CVSS score, CWE ID, and CWE description are also saved for each vulnerability. When one of the four QBoost models with the best score detects an attack (DoS, Fuzzy, Spoofing RPM, or Spoofing Gear), we can obtain the description for each attack

from the taxonomy using the attack class category. After this, the Sentence Transformers is used to get similar sentences using the cosine-similarity. The cosine-similarity is a score in an interval of $[-1, 1]$ where -1 and 1 signify dissimilarity and similarity between the two sentences, respectively.

The top three similar sentences, between the description obtained from the taxonomy and the NVD, were used, employing the cosine-similarity score. Finally, for each of these sentences, several fields are shown:

- *CVE ID*: the CVE identifier.
- *CVE Description*: the description about the CVE.
- *References*: the links that provides a PoC about the attacks.
- *CVSS Score*: the severity score about a vulnerability.
- *CWE ID*: a unique identifier about the CWE assigned to the CVE.
- *CWE Description*: the description about the CWE.

These reference fields are significant because they provide a PoC about the attack. In other words, they show how the attacks have been conducted and, in general, the research(es) that found the vulnerability usually provide information on how to exploit and fix it. On the other hand, the CWE ID and the CWE description are essential because they can support the developer in fixing the vulnerability and understanding the attack consequences.

```

"query": "Vehicle theft by using RollJam device to spoof the key fob",      Description presents into the taxonomy
"sentence": [
  Top-three similar sentences
  {
    "CVE": "CVE-2022-37418",
    "Description": "The Remote Keyless Entry (RKE) receiving unit on certain Nissan, Kia, and Hyundai vehicles through 2017 allows remote attackers to perform unlock operations and force a resynchronization after capturing two consecutive valid key fob signals over the radio, aka a RollBack attack. The attacker retains the ability to unlock indefinitely.",
    "References": "https://hackaday.com/2022/08/17/rollback-breaks-into-your-car/,https://medium.com/codex/rollback-a-new-time-agnostic-replay-attack-against-the-automotive-remote-keyless-entry-systems-df5f99ba9490,https://www.blackhat.com/us-22/briefings/schedule/#rollback---a-new-time-agnostic-replay-attack-against-the-automotive-remote-keyless-entry-systems-27185,https://www.pcmag.com/news/is-your-car-key-fob-vulnerable-to-this-simple-replay-attack,https://www.youtube.com/playlist?list=PLYody84oQL1gxwUrm13xRXxTQL9c05t",
    "CVSSV3": "5.2",
    "CWE": "CWE-294",
    "CWE Description": "Capture-replay attacks are common and can be difficult to defeat without cryptography. They are a subset of network injection attacks that rely on observing previously-sent valid commands, then changing them slightly if necessary and resending the same commands to the server. "
  },
  {
    "CVE": "CVE-2022-36945",
    "Description": "The Remote Keyless Entry (RKE) receiving unit on certain Mazda vehicles through 2020 allows remote attackers to perform unlock operations and force a resynchronization after capturing three consecutive valid key fob signals over the radio, aka a RollBack attack. The attacker retains the ability to unlock indefinitely.",
    "References": "https://hackaday.com/2022/08/17/rollback-breaks-into-your-car/,https://medium.com/codex/rollback-a-new-time-agnostic-replay-attack-against-the-automotive-remote-keyless-entry-systems-df5f99ba9490,https://www.blackhat.com/us-22/briefings/schedule/#rollback---a-new-time-agnostic-replay-attack-against-the-automotive-remote-keyless-entry-systems-27185,https://www.pcmag.com/news/is-your-car-key-fob-vulnerable-to-this-simple-replay-attack,https://www.youtube.com/playlist?list=PLYody84oQL1gxwUrm13xRXxTQL9c05t",
    "CVSSV3": "5.2",
    "CWE": "CWE-294",
    "CWE Description": "Capture-replay attacks are common and can be difficult to defeat without cryptography. They are a subset of network injection attacks that rely on observing previously-sent valid commands, then changing them slightly if necessary and resending the same commands to the server. "
  },
  {
    "CVE": "CVE-2020-15912",
    "Description": "*** DISPUTED ** Tesla Model 3 vehicles allow attackers to open a door by leveraging access to a legitimate key card, and then using NFC Relay. NOTE: the vendor has developed Pin2Drive to mitigate this issue.",
    "References": "https://cansecwest.com/post/2020-03-09-22:00:00-2020-Speakers,https://twitter.com/Kevin2600/status/1218892338182836224,https://www.carhackingvillage.com/speaker-bios#html3nrr,https://www.youtube.com/watch?v=VKSfgox-bs,https://www.youtube.com/watch?v=kQWg-Ywv3S4,https://www.youtube.com/watch?v=nn-3AbTEk1",
    "CVSSV3": "7.0"
  }
]

```

Figure 4.4: The top three sentences most similar to the description derived from the taxonomy. For brevity, CWE and their descriptions have been excluded from the final sentence. Highlighted terms, with yellow and green underlines, indicate similar words.

Results

The IDS was evaluated using the accuracy, precision, recall, and F1-Score metrics. In addition, the time performance of the training and testing were evaluated. Table 4.5 shows the results of QBoost using the OvR technique to obtain a multi-classification. It shows that the values of Precision are almost comparable, with Spoofing Gear and RPM showing the best metrics (1.0), followed by DoS (0.99), Fuzzy (0.99), and Non-attack (0.98) types. The Recall and F1-Scores are very much similar, mostly around 0.99 – 1.0. Only for the Fuzzy attacks, Recall dropped to 0.83 and F1-Score to 0.91, probably because of the randomness of the CAN IDs of the messages. For the Spoofing Gear and Spoofing RPM, QBoost reached 1.0 because the RPM injected by the CAN IDs are always *0316* instead of the RPM that the CAN IDs are *043f*. For the DoS attack, the CAN IDs injected are *000*. For this reason, it is simple for QBoost to identify these attacks. Only for the Fuzzy attacks, Recall dropped to 0.83 and F1-Score to 0.91 because of the randomness of the CAN IDs of the messages. The training time was about *338.26* seconds, whereas the testing time was about *7.15* seconds.

Figure 4.4 shows the results of the NLP process to support the developers. In the figure, some words have been underlined to show the similarity between the query (the description obtained from the taxonomy) and the description obtained from the NVD. In this particular case, the description from the taxonomy refers primarily to DoS, and also, Spoofing attacks. In addition, the query refers to an attack using the RollJam (a crafted hardware component) that consists of jamming the transmission between the key and the car to obtain the code to open the car door. RollBack is another attack that, unlike RollJam, is time-agnostic.

The obtained references can support the developer in understanding these attacks and what strategies can be applied to increase the vehicle's security or fix the vulnerability. For example, considering the CVE-2022-36945 and the Blackhat's reference link⁵ provides several mitigations and descriptions about not only for the RollBack attack but also RollJam.

The CVSS, instead, can help the developer to prioritize the attacks based on the score. The CWE and the CWE description add an additional detail to resolve the identified attack. For example, the CWE description *"Capture-replay attacks are common and can be difficult to defeat without cryptography."* suggests adding cryptography solutions to avoid this attack.

Finally, by using the taxonomy developed by Sommer et al. [240], it is possible to identify other similar attacks by using open-source vulnerability databases like the NVD. In addition, the SeQuADE can be used with other taxonomy that provides an attack class and a description.

The results demonstrate the potential of quantum technologies in enhancing

⁵<https://www.blackhat.com/us-22/briefings/schedule/#rollback---a-new-time-agnostic-replay-attack-against-the-automotive-remote-keyless-entry-systems-27185>

Table 4.5: Results obtained with QBoost in a multi-class strategy.

Label	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Normal	0.9941	0.9871	0.9907	0.9889
DoS		0.9999	1.0	0.9999
Fuzzy		0.9998	0.8291	0.9065
Spoofing Gear		1.0	1.0	1.0
Spoofing RPM		1.0	1.0	1.0

automotive cybersecurity by integrating quantum intrusion detection with structured developer support. By leveraging QBoost for attack classification and an extended taxonomy for vulnerability mapping, the proposed methodology provides actionable insights for mitigating security risks. The successful application of quantum computing in both threat detection and remediation highlights its viability as a tool for strengthening automotive security, ensuring proactive defense mechanisms, and supporting developers in addressing vulnerabilities effectively.

4.2 Smart City Sector

Building on the evaluation of quantum technologies in the automotive sector, this study extends the investigation to the smart city domain to assess the feasibility and accuracy of quantum-enhanced security solutions across different environments. The proposed Quantum as a Service (QaaS) architecture Figure 3.17(subsection 3.2.3) integrates quantum computing into smart city security through a hybrid quantum-classical methodology. It leverages Quantum Machine Learning (QML) algorithms for real-time threat detection, enhancing the capabilities of existing Security Operations Centers (SOCs) and Intelligent Operations Centers (IOCs). The system combines IBM QRadar, a rule-based SIEM, with the D-Wave Leap Quantum Cloud to accelerate incident classification and response. A multi-layered architecture processes smart city data sources, applying QBoost for intrusion detection and utilizing structured dashboards for security monitoring. The following sections detail the experimental setup, evaluation metrics, and results, demonstrating how quantum computing can enhance cybersecurity in interconnected urban infrastructures.

The validation of the proposed architecture was done through rigorous testing of the architecture using various state-of-the-art datasets.

4.2.1 Datasets

TON-IoT

A state-of-the-art publicly available telemetry IoT weather dataset was used for validation, which is a subset of the TON-IoT dataset, [298, 299, 300, 301, 302, 303, 304, 305]⁶. The Cyber Range and IoT Labs of the School of Engineering and Information Technology (SEIT), UNSW Canberra(ADFA), developed the datasets using realistic and large-scale networks. The attributes of the dataset (Table: 4.6) contain the date and time of the recorded data, respective temperature, atmospheric pressure, and humidity. The data are labeled as 0 and 1 for normal and attack/threat incidents and categorized among various attack categories such as normal, DoS, DDoS, Password, and backdoor attacks.

ID	Feature	Type	Description
1	date	Date	Date of logging IoT telemetry data
2	time	Time	Time of logging IoT telemetry data
3	temperature	Number	Temperature measurements of a weather sensor linked to the network
4	pressure	Number	Pressure readings of weather sensor linked to the network
5	humidity	Number	Humidity readings of weather sensor linked to the network
6	label	Number	Tag normal and attack records, where 0 indicates normal and 1 indicates attack
7	type	Number	Tag attack categories, such as normal, DoS, DDoS, and backdoor attacks

Table 4.6: IoT Weather Data Attributes

MQTTset

MQTTset [3] is a state-of-the-art dataset focusing on the widely used Message Queue Telemetry Transport (MQTT) protocol in IoT devices. MQTT serves as a lightweight, publish-subscribe protocol and facilitates machine-to-machine communication within a network. Specifically designed for environments with limited network bandwidth or devices with resource constraints, such as those found in the IoT, MQTT enables connections to remote locations⁷.

The MQTTset dataset consists of readings of eight IoT sensors connected to an MQTT broker based on Eclipse Mosquitto v1.6.2. The network was created to simulate a smart home environment (Figure 4.5). The sensors uniquely identified by IP addresses acquire information from the IoT sensors

⁶<https://research.unsw.edu.au/projects/toniot-datasets>

⁷<https://mqtt.org/>

like Temperature, Light intensity, Humidity, Motion sensor, CO-Gas, Smoke, Fan speed controller, Door lock, Fan sensor, and Motion sensor at different time intervals [306].

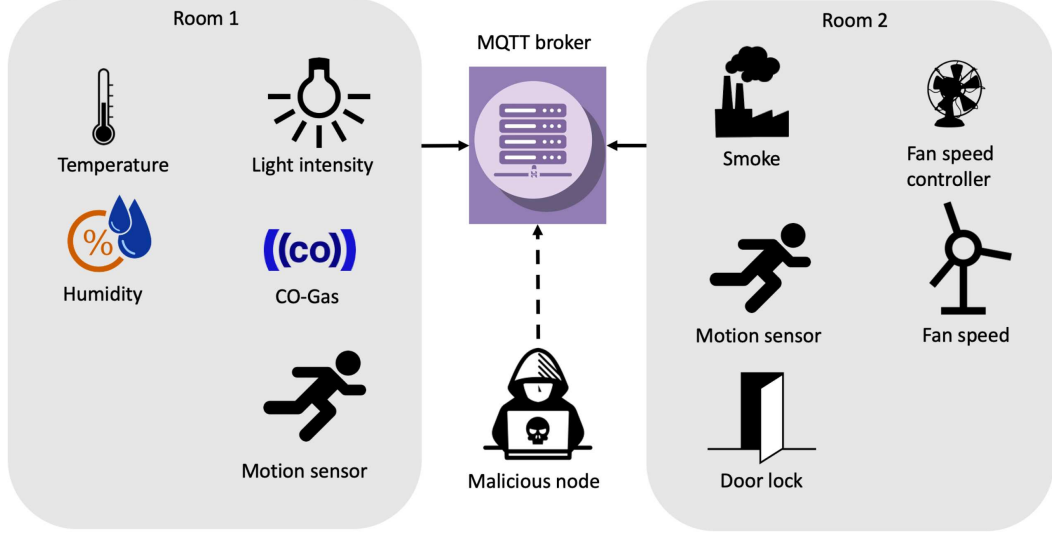


Figure 4.5: Sensors in MQTTset Scenario [3].

The attack on the MQTTset data includes different attack signatures, flooding denial of service, MQTT Publish flood, SlowITe, malformed data, and brute force authentication. For this research, we considered a reduced version of the dataset with 330936 rows and 34 data variables. The dataset consists of 33 features/independent variables and one target/dependent variable called *target*. For the sake of brevity, we have not provided the details of the features in this paper. Further details about the dataset can be acquired from the MQTTset paper.

CICIoT2023

The CICIoT2023 dataset [307] is an IoT dataset that consists of seven attack categories: **DDoS**, **DoS**, **Recon**, **Web-Based**, **Brute-Force**, **Spoofing**, and **Mirai**. For each of these categories, the authors have performed different types of attacks. For example, for DDoS, they performed attacks like *UDP Flood* and *HTTP Flood*. These attacks are generated with different tools like nmap, hping3, and golang-httpflood.

To perform these attacks, the researchers utilized an IoT platform comprising 105 IoT devices. Of these, 67 were directly involved in the attacks, while the remaining 38 Zigbee and Z-Wave devices were connected to five hubs [307]. The network topology is divided into two parts. In the first part, an ASUS router links the network to the Internet, and a Windows 10 machine shares this connection. A Cisco switch is placed between this computer and a VeraPlus access point, connecting seven Raspberry Pi devices responsible for the attacks in the experiments. Traffic is captured using Wireshark via the Cisco switch

that is connected to the second part of the network through a Gigamon Network Tap.

The CICIoT2023 has 46.686.579 rows of data with 46 independent variables (features). The dataset is 13.8 Gigabytes (GB) in size. The dataset is made available as a set of 169 different CSV files, which need to be concatenated before experimentation with the full dataset. The data are labeled with 33 attack types: *DDoS-ICMP_Flood*, *DDoS-UDP_Flood*, *DDoS-TCP_Flood*, *DDoS-PSHACK_Flood*, *DDoS-SYN_Flood*, *DDoS-RSTFINFlood*, *DDoS-SynonymousIP-Flood*, *DoS-UDP_Flood*, *DoS-TCP_Flood*, *DoS-SYN_Flood*, *Mirai-greeth_flood*, *Mirai-udpplain*, *Mirai-greip_flood*, *DDoS-ICMP_Fragmentation*, *MITM-ArpSpoofing*, *DDoS-ACK_Fragmentation*, *DDoS-UDP_Fragmentation*, *DNS_Spoofing*, *Recon-HostDiscovery*, *Recon-OSScan*, *Recon-PortScan*, *DoS-HTTP_Flood*, *VulnerabilityScan*, *DDoS-HTTP_Flood*, *DDoS-SlowLoris*, *DictionaryBruteForce*, *BrowserHijacking*, *SqlInjection*, *CommandInjection*, *XSS*, *Backdoor_Malware*, *Recon-PingSweep*, *Uploading_Attack*; and one normal data label, *BenignTraffic*.

In the context of ML algorithms, the dataset is extremely unbalanced, which means that the distribution of classes or categories is significantly skewed or imbalanced. For example in a more granular level the dataset has about 15.5% records labeled as *DDoS-ICMP_Flood*, 11.6% of records labeled as *DDoS-UDP_Flood*, and 9.7% records labeled as *DDoS-TCP_Flood*, whereas there are only 0.002%, 0.005% and 0.006% records labeled as *Uploading Attack*, *Recon-PingSweep*, and *Backdoor_Malware*. Further, normal messages (*BenignTraffic*) consist of only 2.3% of the dataset, which makes prediction difficult using ML classification algorithms.

4.2.2 Data Pre-processing & Experiments

All experiments were carried out on a Ubuntu SMP 20.04 server with 64-bit architecture, AMD Ryzen Threadripper PRO 3975WX CPU with a clocking speed of 4368MHz with 32-Cores and 64 GB of RAM. Quantum Annealing-based QBoost classifier algorithm was executed on D-Wave⁸ Leap Solvers. Whereas the IBM Qiskit-based QML algorithms were run on Ubuntu SMP with 8 cores CPU of speed 2.1 GHz and 32 GB RAM, the quantum parts were run on the Qiskit backend.

Model Accuracy, Precision, Recall, and F1-score have been considered as the quality metric to ascertain the quality of classification rendered by the different QML models and traditional ML models in the proposed architecture. Further, time efficiency for training and prediction was also taken into consideration for comparison of the models.

On TON-IoT Dataset

Some features of the **TON-IoT dataset**, date and time, were removed from

⁸<https://www.dwavesys.com/>

the IoT telemetry weather dataset since they did not contribute to the results, and temporal analysis was not carried out in this case. The attack typology feature(Type) was also removed owing to binary classification. Further, there were no NaN or null values in the dataset. As the Temperature, Pressure, and Humidity values were in different measurement units, those were scaled using *MinMaxScaler* [308] into the range of 0 and 1.

The TON_IoT dataset contains 650242 records, with 559718 normal records(label 0) and 90524 attack records(label 1). The dataset has only 16.17% attack records, making it unbalanced, which is undesirable as this might lead to biased predictions. To handle data imbalance, it was downsampled using *RandomUnderSampler*⁹ library based on the number of records in the minority class so that the train and test data have the same number of normal and attack data. Moreover, all 0 value labels(normal) were converted to -1 to fit QBoost requirements, as it can classify strictly between 1 and -1 only. Then the processed and scaled data were split into a ratio of 80% and 20% for training and testing, respectively, and consequently were fed to different quantum machine learning classification algorithms.

QBoost [47] and VQC [309] were used to validate the proposed architecture. Although the proposal discusses the usage of Quantum as a Service in the context of Smart City Security, for the sake of completeness and fairness, a comparison with Random Forest [296], a state-of-the-art traditional machine learning algorithm, was also carried out. After multiple rounds of testing hyperparameters considered for QBoost are $\langle NUM_WEAK_CLASSIFIERS: [35] \rangle$, $\langle TREE_DEPTH: [3] \rangle$, $\langle lmd(\lambda): [0.5] \rangle$, $\langle DW_PARAMS: num_reads = [3000], auto_scale = [True], num_spin_reversal_transforms = [10] \rangle$. For VQC, the hyperparameters considered are $\langle loss: [cross_entropy] \rangle$, $\langle optimizer: [COBYLA(maxiter = 30)] \rangle$, where the feature map was generated with a ZZFeatureMap using $[reps = 1]$ and the ansatz with RealAmplitudes using $[reps = 3]$. The hyperparameters for RF are $\langle n_estimators: [400] \rangle$ and $\langle max_depth: [30] \rangle$.

On MQTTset Dataset

The **MQTTset dataset** was checked for categorical features, missing values, NaN values, or infinity values. There were six categorical features in the dataset, which were removed since using any encoding schemes like one-hot encoding or binary encoding would greatly increase the number of features. For example, if the IP address is encoded using one hot encoder, it generates 32 new features. Keeping in mind that we are using QML algorithms, we try to keep the number of features to as minimum as possible, as the number of features impacts the number of qubits to be used, particularly in IBM Qiskit. The dataset had 12 features, where each had the same values for all records. Therefore, these features had no impact whatsoever on the data analysis and were also removed. The MQTTset did not have NaN, missing, or infinity values. Further, the

⁹<https://imbalanced-learn.org/stable/>

target variable has six different labels, *legitimate*, *DoS*, *malformed*, *bruteforce*, *slowite*, and *flood*, where legitimate was converted into -1, and the rest into 1 for classification. The dataset, after all these processings, is left with only 15 features. *MinMaxScaler* [308] is then applied to the dataset to scale the data into the range of 0 and 1. The dataset is balanced, having an equal number of attack and non-attack scenarios, so there is no need to balance the dataset. Finally, the dataset was split into a ratio of 70% and 30% for training and testing, respectively.

Moreover, to reduce the dimensionality of the data, Principal Component Analysis (PCA) was applied to the dataset [308]. The QML models were tested with different numbers of reduced features. QBoost works with a varied range of features, small or large, and does not have a large impact on the number of qubits for small changes in the number of features. On the other hand, IBM QMLs like VQC and PegaSoS QSVC use as many qubits (simulated) as there are features. Therefore, the complexity of the model increases exponentially and becomes nonviable for systems in laboratory settings with limited RAM or GPU RAM. Although there are no bars for the size of the dataset, the problem's scalability can rapidly deteriorate with the increasing number of features. Moreover, the Qiskit [310] QML algorithms cannot reap benefits from GPU RAM and are dependent on regular RAM for execution.

The proposed architecture was validated by QBoost [311, 312, 47], VQC [309], and PegaSoS QSVC [310, 313]. The experiments were carried out with two, four, six, and eight features for all QML algorithms. Although QBoost and PegaSoS QSVC finished code execution with all features (15), VQC has a serious limitation, and it requires extremely large RAM and a long running time for processing datasets with a large number of features. VQC exhausted the available memory (64GB) of the laboratory server while training the dataset with more than eight features, that is, with more than eight qubits, and stopped processing. Thus, experimentation with a higher number of features for VQC was not possible in this case.

The hyperparameters considered for QBoost are *Num Weak Classifiers*: 35, *Tree Depth*: 3, *lmd (lambda)*: 0.5. The DWave parameters are: *Num Reads*: 3000, *Auto Scale*: True, *Num Spin Reversal Transforms*: 10. For VQC, the hyperparameters considered are $\langle loss: [\text{Cross Entropy}] \rangle$, $\langle optimizer: [\text{COBYLA}(\text{maxiter}=30)] \rangle$, where the feature map was generated with a ZZFeatureMap using [reps=1] and the ansatz with RealAmplitudes using [reps=3]. For PegaSOS QSVC, the hyperparameters considered are $\langle Quantum Kernel=[\text{qkernel}] \rangle$, $\langle C: [1000] \rangle$, and $\langle num_steps: [100] \rangle$, where the qkernel was created with FidelityQuantumKernel using a feature_map, which is in turn generated with a ZFeatureMap with [Feature Dimension=num_qubits] and [reps=2].

On CICIOT2023 Dataset

The **CICIOT2023 dataset** underwent the following preprocessing steps to prepare it for machine learning analysis using the QBoost QML algorithm from

DLQC:

1. *Categorical Feature Removal*: Non-numeric, categorical features were excluded from the dataset to lower its dimensionality, as high-dimensional data can negatively impact the accuracy of prediction models.
2. *Redundant Feature Removal*: Features with identical values across all instances were eliminated. Since such features do not contribute meaningful information to the prediction process, their removal helped simplify the dataset.
3. *Handling Missing Values*: Instances containing Infinity or NaN (Not-a-Number) values were addressed by replacing them with the median of their respective feature values. This step ensured a complete dataset, suitable for analysis.
4. *Target Variable Transformation*: The dependent feature, serving as the target variable, was transformed. Non-attack messages were assigned a value of -1, while attack messages were assigned a value of 1. This transformation ensured compatibility with quantum machine learning algorithms.
5. *Data Normalization*: Independent variables were normalized using the *MinMaxScaler*¹⁰ from the *Scikit-Learn* library. This normalization scaled the values of the features to the range [0, 1], promoting consistency in the feature space.
6. *Balancing the Dataset*: Addressing class imbalance was critical. Techniques such as Synthetic Minority Over-sampling Technique (SMOTE)¹¹ and Random Under-Sampling, both implemented using the *imblearn* library, were employed. The optimal parameters for balancing were determined through *CrossValidation* using mean recall scores of the *DecisionTree* classifier. These methods ensured a more balanced representation of attack and non-attack classes. Post-balancing, the dataset was reduced to 10,257,385 rows.
7. *Dimensionality Reduction*: To address challenges posed by the dataset's high dimensionality (46 features), Principal Component Analysis (PCA)¹² from the *scikit-learn* library was applied. PCA reduced the feature set while retaining the most informative components. The threshold method, based on the explained variance ratio, retained the 13 most relevant features.
8. *Data Splitting*: The preprocessed and normalized data were divided into training and testing sets, with an 80% to 20% split [307]. This split was

¹⁰<https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html>

¹¹https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html

¹²<https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html>

essential for evaluating the model's performance and ensuring its ability to generalize to unseen data.

Within the proposed architecture, particular emphasis was placed on the deployment of the QBoost [47] QML algorithm as the primary model for validation purposes. The operationalization of QBoost involved an iterative process that entailed training the model on the meticulously preprocessed and balanced dataset, which had previously undergone dimensionality reduction via PCA. In the interest of comprehensiveness and impartiality, a comparative analysis was also conducted. To this end, the performance of QBoost was compared with that of Random Forest (RF) [296], a state-of-the-art traditional machine learning algorithm. This comparison aimed to provide a holistic view of the strengths and weaknesses of both quantum and classical approaches in the context of Smart City Security.

To ensure thoroughness and reproducibility, it is essential to record the hyperparameters used in the previously mentioned experiments. The relevant hyperparameters are detailed below.:

1. *SMOTE Hyperparameters:*

- (a) *sampling_strategy*: 0.1, specifying the preferred proportion of synthetic samples to produce for the minority class in relation to the majority class.
- (b) *k*: 7, specifying how many nearest neighbors are taken into account for generating synthetic samples.

2. *Random Undersampling Hyperparameters:*

- (a) *sampling_strategy*: 0.8, indicating the preferred proportion of instances to be kept for the majority class relative to the minority class.

3. *RepeatedStratifiedKFold Configuration:*

- (a) *n_splits*: 10, specifying the number of splits or folds in the cross-validation process.
- (b) *n_repeats*: 3, indicating the number of times the cross-validation process was repeated.
- (c) *n_jobs*: -1, allowing parallel processing when calculating cross-validation scores.

4. *PCA Configuration:*

- (a) *Threshold method*: Depends on the explained variance ratio.
- (b) *Threshold value*: 0.99, specifying the variance proportion that the selected principal components should collectively account for.

5. *QBoost λ value (*lam*):* 0.085, introduced as a means to manage model complexity and mitigate the potential for overfitting.

Finally, several key performance metrics, as mentioned before, were computed to evaluate and compare the effectiveness of the algorithm within the proposed architecture.

IBM QRadar SIEM Rules

The following subsection describes the rules created with QRadar to identify the considered attacks. This section brings out the novelty of the methodology regarding the adoption of hybrid quantum-classical approach.

DDoS

A Distributed Denial of Service (DDoS) attack is a malicious attempt to disrupt the normal functioning of a network, service, or website by overwhelming it with a flood of traffic. The goal of a DDoS attack is to make the target system or network unavailable to its intended users. A large number of compromised computers typically carry out DDoS attacks, often referred to as a "botnet", that are controlled by the attacker. In the CICIOT23 dataset, the authors utilized multiple Raspberry Pi devices in an SSH-based master-client configuration to execute these attacks [307].

To create the rules that identify a DDoS attack, we first use a series of building blocks, and then we create the main rule that aggregates the building blocks in order to identify the attack. A building block (BB) group commonly uses tests to build complex logic so that it can be used in rules.

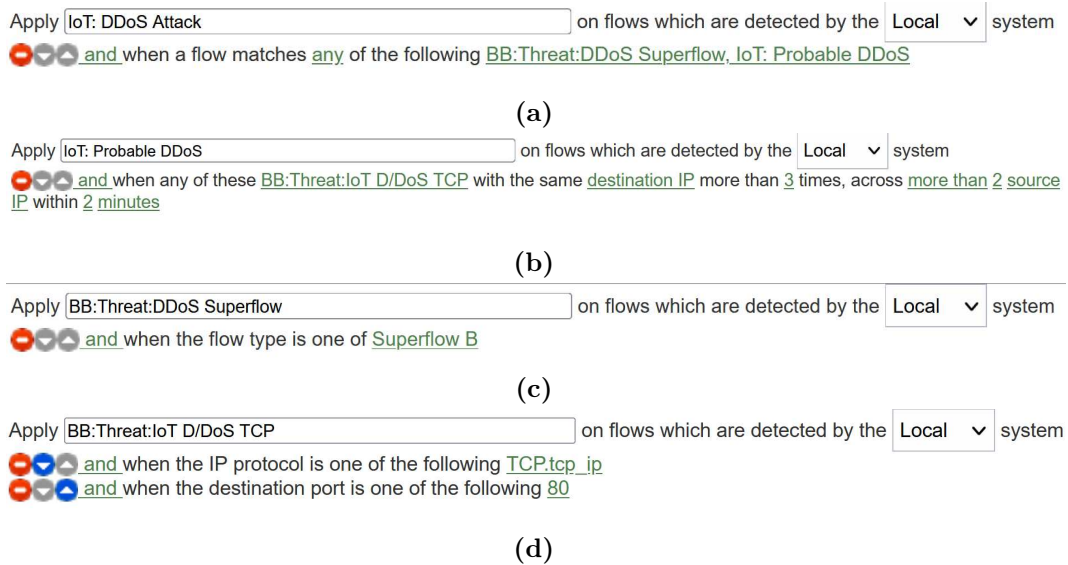


Figure 4.6: The figures depict the rules for identifying an HTTP Flood attack categorized as DDoS. Figure (a) displays the main rule, and if any of the conditions outlined in (a) are met, an offense will be triggered. Figure (b) presents the rules for two or more IP Sources. Figures (c) and (d) show the BB rules.

Figure 4.6 shows the rules created to identify an HTTP Flood attack with the

Description	Offense Type	Offense Source	Magnitude	Source IPs	Destination IPs	Us Lo So Events
IoT: DDoS Attack	Source IP	192.168.137.90		192.168.137.90	192.168.137.139	N/A Cus 113
IoT: DDoS Attack	Source IP	192.168.137.225		192.168.137.225	192.168.137.139	N/A Cus 104
IoT: DDoS Attack	Source IP	192.168.137.89		192.168.137.89	192.168.137.139	N/A Cus 118
IoT: DDoS Attack	Source IP	192.168.137.64		192.168.137.64	192.168.137.139	N/A Cus 102

Figure 4.7: Identification of DDoS attack offense.

category DDoS. Figure 4.6d and Figure 4.6c show the BB rules. In particular, Figure 4.6d is used to filter the flow network based on the TCP protocol and port 80 since the HTTP Flood is performed by sending a GET request on port 80. Figure 4.6c shows the Superflow characteristics. IBM QRadar groups the individual flows into a Superflow. There are three types of Superflow: A (Network Scan), B (DDoS), and C (Port Scan) [314]. QRadar looks for flows where many hosts send data to one destination host and flags this activity as Superflow B.

Figure 4.6b is used to identify two or more source IPs that send a high number of data to a single destination IP in two minutes. Finally, the rule *IoT: DDoS Attack* (Figure 4.6a) aggregates the Superflow and the "Probable DDoS". In particular, if one of those rules is true, an offense will be raised. An important consideration is that it is possible to modify the rule in Figure 4.6b to be a DoS and not a DDoS. For example, we could modify the source IP from two(2) to one(1).

Figure 4.7 shows the offense generated when IBM QRadar SIEM identifies an HTTP Flood attack with a DDoS category. In the generated offense, it is possible to see that we have multiple *Offense Source* and *Source IPs* that correspond to the Raspberry Pi that performs the attack to one *Destination IP*. The events column indicates the number of event flows where a DDoS attack was identified. Since the CICIoT2023 does not provide the list of IP devices but only the Media Access Control (MAC) addresses, we have used this information to understand the attack messages by analyzing the PCAP file. A PCAP (short for Packet Capture) file is a type of file format used to store network traffic data. It is commonly associated with network analysis and troubleshooting tools.

Cross-Site Scripting (XSS)

Cross-Site Scripting (XSS) is a web security vulnerability that arises when a web application permits users to inject malicious scripts into web pages, which are subsequently viewed by other users. XSS attacks are among the most common security vulnerabilities in web applications and can lead to severe consequences, such as data theft, session hijacking, and website defacement [315]. To carry out this attack, the researchers utilized DVWA [316], a deliberately vulnerable PHP/MySQL web application commonly used for educational purposes or to practice penetration testing techniques.

To create the rule that identifies an XSS attack, we first analyzed the PCAP file. We have seen that most of the messages are related to basic XSS injection

Apply on flows which are detected by the system
 and when the source payload matches the regex `(script|alert|onmouse[a-z]+|onkey[a-z]+|onload|onunload|ondragdrop|onblur|onfocus|onclick|ondblclick|onsubmit|onreset|onselect|onchange)`

Figure 4.8: XSS attack identification rule.

techniques, such as using the tag *script* with *alert*. Figure 4.8 shows the rule that matches several tags generally used to perform an XSS attack with a *regular expression*.

Description	IoT: XSS Detection	Offense Type	Source IP
		Event/Flow count	20 events and 0 flows in 1 categories
Source IP(s)	192.168.137.147	Start	Sep 20, 2023, 5:36:03 PM
Destination IP(s)	Local (3)	Duration	39m 47s
Network(s)	Net-10-172-192-Net, 192.168.0.0	Assigned to	Unassigned

(a) Offense information when XSS attack is detected

Description	Offense Type	Offense Source	Magnitude	Source IPs	Destination IPs
IoT: XSS Detection	Source IP	192.168.137.147	■■■	192.168.137.147	Local (3)

(b) More offense information

Figure 4.9: XSS attack and Offense detection information.

Figure 4.9a and Figure 4.9b show the generated offense when an XSS attack is identified by QRadar using the rule in Figure 4.8.

SQL Injection

SQL injection is a web application security vulnerability that occurs when attackers exploit input fields to inject malicious SQL queries into a web application's database. This exploitation can result in unauthorized data access, data manipulation, or the extraction of sensitive information from the database [317]. Like the Cross-Site Scripting (XSS) attack, the researchers conducted the SQL injection attack using DVWA.

Apply on flows which are detected by the system
 and when the source payload matches the regex `(UNION|ALL|SELECT|OR|AND|CONCAT|WHEN|SELECT|FROM|NULL|SLEEP)`
 and when the destination port is one of the following 80

Figure 4.10: SQL Injection attack identification rule.

The initial analysis of the PCAP file associated with the SQL Injection attack revealed that the attack was executed using fundamental SQL Injection techniques, such as UNION and SLEEP commands, among others. Based on these observations, a rule was created to identify SQL Injection attempts by incorporating the most frequently used SQL keywords associated with such attacks. Figure 4.10 shows the rule, including the regular expression designed for detection.

The offense generated is shown in Figure 4.11. The field *Local (4)* specifically refers to the number of destination IPs involved in the attack. By double-

Description	Offense Type	Offense Source	Magr	Source IPs	Destination IPs	Use	Log Sou	Events
IoT: SQL Injection	Source IP	192.168.137.178	■■■	192.168.137.178	Local (4)	N/A	C...	170

Figure 4.11: Offense generated by SQL Attack.

Description	IoT SQL Injection			Offense Type	Source IP		
Source IP(s)	192.168.137.178 (192.168.137.178)			Event/Flow count	170 events and 0 flows in 1 categories		
Destination IP(s)	Local (4)			Start	Sep 29, 2023, 5:09:00 PM		
Network(s)	Net-10-172-192-Net-192-168-0-0			Duration	48m 59s		
				Assigned to	Unassigned		

List of Local Destination IPs											
?	Destination IP	Magnitude	Location	Vulnerability	User	MAC	Weight	Offenses	Source(s)	Last Event/Flow	Events/Flows
	192.168.137.4	■■■	Net-10-172-192...	No	?	?	0	4	1	51m 10s	209
	192.168.137.217	■■■	Net-10-172-192...	No	?	?	0	2	1	4m 11s	103
	192.168.137.154	■■■	Net-10-172-192...	No	?	?	0	2	1	4m 19s	4
	192.168.137.86	■■■	Net-10-172-192...	No	?	?	0	2	1	32m 10s	44

Figure 4.12: SQL Injection with destination IPs involved in the attack.

clicking on the generated offense, as illustrated in Figure 4.12, these IPs and additional information can be accessed.

Apply IoT: Possible Port Scanning on flows which are detected by the Local system and when a flow matches any of the following Port Scanning: Over 10 Seconds

(a)

Apply Port Scanning: Over 10 Seconds on flows which are detected by the Local system and when all of these BB:Threat:IoT Port Scanning, in any order, from any source IP to the same destination IP, over 10 seconds

(b)

Apply BB:Threat:IoT Port Scanning on flows which are detected by the Local system and when the source byte/packet ratio is equal to 64 bytes/packet and when the IP protocol is one of the following TCP:tcp_ip

(c)

Figure 4.13: The images depict the rules used to detect a port scan. Figure (a) displays the primary rule, which triggers an offense if any of the rules outlined in (a) are met. Figure (c) illustrates a BB rule.

Port Scan

A port scan is a reconnaissance technique used to identify which network ports on a target system are open and listening for incoming connections [318]. While network administrators frequently use port scanning for legitimate purposes such as troubleshooting and security assessments, attackers may exploit it to detect vulnerabilities in a target system. For the port scan analysis, the researchers utilized the tool *Nmap* [319].

To create the rules (Figure 4.13) for detecting port scans in conjunction with XSS and SQL Injection attacks, the PCAP file corresponding to the port scan attack was analyzed. Specifically, it was observed that most messages sent using Nmap have a byte length of 64. Based on this finding, the first rule was developed (Figure 4.13c), designed to match only if the port scan targets the TCP protocol. Additionally, a second rule (Figure 4.13b) was created to detect

flow traffic where a device sends scanning packets (SYN, SYN-ACK, RST) to the same destination IP for over 10 seconds. Finally, the SIEM system (Figure 4.13a) generates an offense when the rule *Port Scanning: Over 10 Seconds* evaluates to true.

Description	Offense Type	Offense Source	Magnitude	Source IPs	Destination IPs	Us	Lo So	Events
IoT: Possible Port Scanning	Source IP	192.168.137.41		192.168.137.41	192.168.137.178	N/A	Cus	1,192

(a)

Description	IoT: Possible Port Scanning	Offense Type	Source IP
		Event/Flow count	1,192 events and 0 flows in 1 categories
Source IP(s)	192.168.137.41 (192.168.137.41)	Start	Sep 21, 2023, 7:36:00 PM
Destination IP(s)	192.168.137.178 (192.168.137.178)	Duration	1m 59s
Network(s)	Net-10-172-192.Net 192.168.0.0	Assigned to	Unassigned

(b)

Figure 4.14: Figure (a) shows the offense generated when a Port Scan attack is detected. Instead, Figure (b) shows the offense generated but with more details.

Figure 4.14 illustrates the offense generated upon detecting a Port Scan attack. The figure shows a single Source IP carrying out the attack on a single Destination IP.

4.2.3 Results

TON-IoT Outcomes

Figure 4.15 shows the quality results of the experiments on the dataset with Random Forest, QBoost, and VQC. Random Forest and QBoost achieved comparable and good classification quality, whereas the performance of VQC was lower. The low performance of VQC can be attributed to the limited number of qubits and noise in current quantum devices. We also have to take into consideration that the IBM quantum computer-based QMLs run on simulators rather than actual qubits. Although, running any complex algorithm on real qubits would result in very low-quality results owing to the probabilistic and noisy nature of the qubits.

Algorithm	Training Time (s)	Prediction Time (s)
QBoost	24.62	0.068
Random Forest	51.7	0.94
VQC	20406.25	179.67

Table 4.7: Time Performance based comparison of Classifiers

Table 4.7 presents the training and prediction times for the algorithms mentioned above. VQC took almost 5.6 hours for training, and about 3 minutes for prediction, whereas QBoost took 24.6 seconds and 0.068 seconds for training and prediction, respectively. On the other hand, Random Forest took 51.7 and

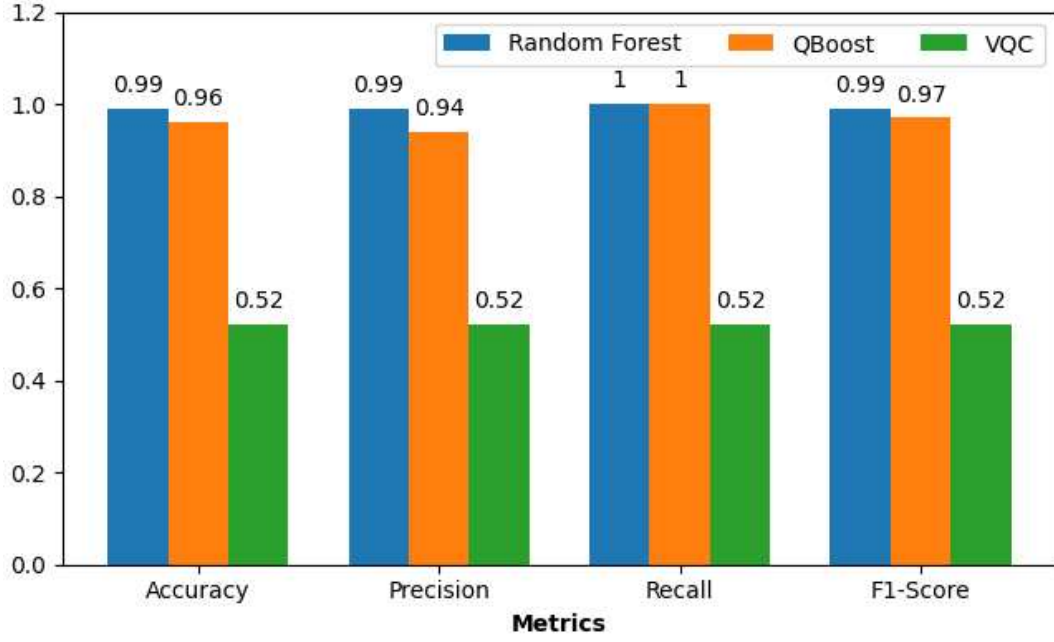


Figure 4.15: Quality metrics based comparison of Classifiers

0.94 seconds on average for training and prediction, respectively. In this case, QBoost emerged as the best classifier in this domain among other quantum-based classifiers, both with respect to quality and time.

MQTTset Outcomes

The comparison of the quality of the classification performed by the QML algorithms has been presented in Figure 4.16 for model training using PCA with the number of features as *two*, *four*, *six*, and *eight*, respectively. Evidently, QBoost, the quantum annealing-based classifier, outperforms the rest of the IBMC Qiskit's quantum gate-oriented simulator-based classifiers like VQC and PegaSoS QSVC. When the number of features/number of qubits is *two*, we can see that there is a substantial difference in the quality of the results between QBoost, PegaSoS QSVC, and VQC. With the increase in the number of features to *four* and subsequently, to *eight*, the difference between the results decreases. This is due to the fact that with more features VQC and PegaSoS QSVC learns more, and thus, the model's quality gradually increases. Whereas, in the case of QBoost there are no visible improvements with the increase in the number of features. It is interesting to notice that with $n = 4$, VQC presented the best quality metrics. For the sake of brevity, we only present the results of QBoost and PegaSoS QSVC, which are also available for VQC. For the sake of completeness, when the number of features was increased to 14 QBoost and PegaSoS QSVC only achieved 2.4% and 0% gains in quality.

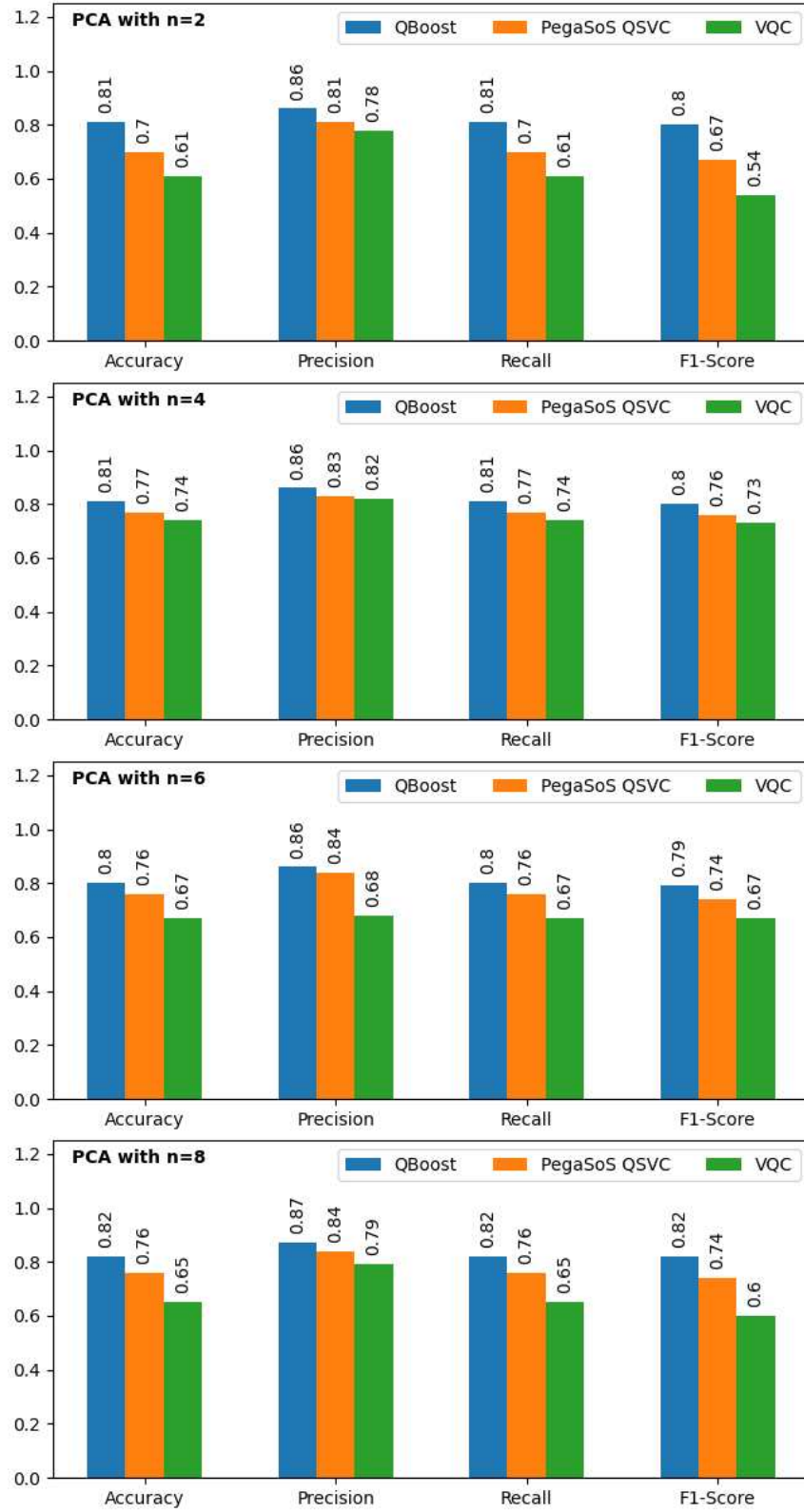


Figure 4.16: Quality metrics using PCA.

Table 4.8 presents the time taken by the QML algorithms for training and prediction of the MQTTset dataset in the proposed architecture. The prediction

Table 4.8: Time Performance (PCA $n=2,4,6,8$).

PCA (n)	Algorithm	Training Time (s)	Prediction Time (s)
n=2	QBoost	13.24	0.097
	PegaSOS	2.65	5405.17
	VQC	14075.14	194.18
n=4	QBoost	13.79	0.095
	PegaSOS	4.27	9347.38
	VQC	34478.57	460.9
n=6	QBoost	16.47	0.11
	PegaSOS	6.78	13781.99
	VQC	62182.18	859.68
n=8	QBoost	19.66	0.12
	PegaSOS	10.71	22589.22
	VQC	110296.37	1575.01

times for QBoost for any number of features (2, 4, 6, or 8) are the fastest among the three proposed quantum classifiers. PegaSoS QSVC's training times are the fastest among the classifiers, but unfortunately, the prediction is extremely slow and takes a lot of time, 1.5, 2.7, 3.8, and 6.28 hours for two, four, six, and eight features, respectively. The training times for VQC are the slowest among the three, with 3.9, 9.5, 17.3, and 30.63 hours for two, four, six, and eight features. The long training and prediction times for VQC can be attributed to the fact that VQC is a special variant of Neural Network Classifier with a SamplerQNN [310] and thus employs quantum neural networks for classification.

QBoost, in this case, has performed reasonably well over the other QML algorithms based on both quality and time efficiency. However, Quantum Machine Learning is in its nascent stage, and any advantage achieved thereof should be treated with caution. In this light, further investigation is always necessary to find out the general applicability of QML algorithms in the proposed architecture.

CICIoT2023

Figure 4.17 summarizes the experimental results obtained from the dataset, showcasing the performance of RF and QBoost classifiers. Both algorithms demonstrated similar and impressive classification performance. While RF achieved outstanding results, QBoost—a quantum machine learning algorithm—also performed competitively, highlighting its potential for data analysis and predictive tasks in the context of Smart Cities. The findings suggest that as quantum computing technology continues to mature and becomes more accessible, it could provide innovative solutions for addressing complex challenges associated with Smart City systems.

Table 4.9 details the training and prediction times for the two algorithms, providing valuable insights into their computational efficiency. Although the primary goal of this study is not to directly compare RF and QBoost, the

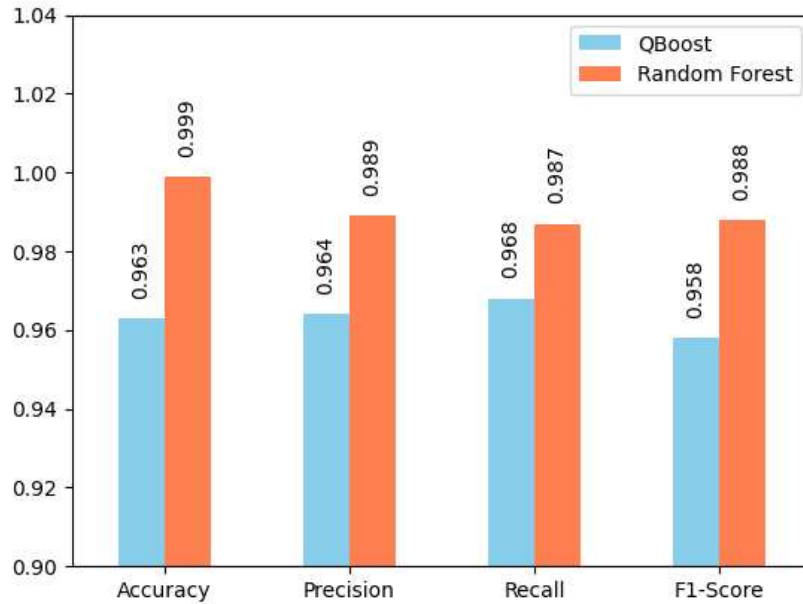


Figure 4.17: Quality metrics of Classifiers.

results confirm that QBoost is a feasible option for integration into Smart City frameworks. Both classifiers produced very similar results in terms of quality metrics, indicating that QBoost can serve as a credible alternative to RF. Specifically, RF achieved a classification accuracy of 99%, while QBoost attained an accuracy of 96%, resulting in a marginal difference of just 3% across all quality metrics.

Algorithm	Training Time (s)	Prediction Time (s)
QBoost	554.5	2.24
Random Forest	2667.5	7.83

Table 4.9: Time efficiency: QBoost and Random Forest.

However, what sets QBoost apart is its notable advantage in terms of computational efficiency. Specifically, we observed an impressive reduction of approximately 80% in training time and a substantial 71% decrease in prediction times when comparing QBoost with RF. This substantial improvement in efficiency is particularly evident in the case of prediction times, where QBoost achieved a remarkable processing speed of just 2.24 seconds. Even when dealing with datasets containing over a million data entries, the rapidity of QBoost's predictions remains a standout feature.

QRadar SIEM Outcomes

With IBM QRadar SIEM, we note that the time between the identification of an attack and the offense alert varies from about 2 to 3 minutes. This depends

on whether we are using a version deployed on a virtual machine and not in a cloud environment. By using more efficient hardware, this time could decrease.

The coexistence of QML (QBoost) models and QRadar SIEM suggests the possibility of hybrid approaches. Future research could explore the synergy between classical and quantum computing to harness the strengths of both paradigms for improved Smart City solutions. It is evident from the results of the experiments that a hybrid approach of using QML, QBoost in this case, IBMQ in couple of other scenarios, and IBM QRadar reinforces the security aspects of Smart City incidents by detecting offenses or security incidents in real time.

The results demonstrate that quantum technologies, particularly Quantum Machine Learning, offer promising enhancements for smart city security by significantly improving threat detection speed while maintaining comparable accuracy to classical methods. The successful integration of QBoost with IBM QRadar highlights the potential of hybrid quantum-classical architectures in augmenting real-time security monitoring and incident response. These findings suggest that quantum computing can not only strengthen existing security infrastructures but also pave the way for more resilient security frameworks in future quantum-powered smart city ecosystems.

4.3 Security and Privacy-Aware Quantum Software Development

The Quantum Privacy Knowledge Base (QPKB) and Quantum Privacy-Oriented Software Development (QPOSD) frameworks were proposed to systematically integrate security and privacy into quantum software engineering. These frameworks aim to address quantum-specific vulnerabilities, incorporate privacy-enhancing techniques, and provide structured methodologies for secure quantum software development. To assess their usability and feasibility, a case study was conducted, evaluating their practical integration into quantum software engineering. The results examine key aspects such as quantum vulnerabilities, privacy-enhancing techniques, and quantum software verification. Additionally, the study explores the implementation of Quantum Privacy by Design principles and Privacy Design Strategies, demonstrating how these frameworks can be applied to enhance the resilience of quantum software systems.

4.3.1 Results and Discussion

Let us visit a scenario where Quantum Privacy-Oriented Software Development (QPOSD) is implemented in a Quantum Hybrid Software Development scenario. We present an example scenario of a larger quantum hybrid application involving data transmission between organizations or between API (application programming interface) endpoints or other data transmission channels between software entities.

- *Context*: The data transmission between two software entities is secured by the asymmetric Advanced Encryption Standard (AES), which guarantees the privacy, security, and integrity of the data.
- *Problem*: Application of QPbD principles like *Proactive not Reactive*; *Preventive not Remedial*, and *Privacy as the Default Setting*, followed by the application of QPDS and QPP related to Encryption provides a clear guideline for the development of secure data transmission. Under classical computational circumstances, AES or RSA (Rivest-Shamir-Adleman) encryption is strong enough to protect the data. But, theoretically, quantum computers have the potential to break certain cryptographic algorithms, including AES (Advanced Encryption Standard), using quantum algorithms such as Shor's algorithm, when executed on a sufficiently powerful quantum computer.
- *Mitigation*: Data transmission is implemented over either quantum or classical using Quantum Key Distribution. QKD guarantees that there are no malicious listeners in the channel, and if found, aborts the connection at startup, thereby providing unconditional security.

Implementing the QPOSD methodology successfully identified vulnerabilities and implemented security measures to address them, leveraging their understanding of the issues. Additionally, it supports making design decisions related to privacy elements, even if there is a lack of specific privacy expertise. Importantly, the adoption of QPOSD did not disrupt the existing development processes by any means and integrated privacy-friendly directions by default.

The results of the experimental prototype generated a privacy-oriented quantum hybrid secure code that successfully handles intruders in data transmission. Figure 4.18a and Figure 4.18b show the functionality of the secure data transmission channel using Quantum Key Distribution (QKD). We can see that the prototype notifies if or note there is an intruder in the channel. If there are no intruders, the key exchange process is followed by data transmission. If there is an intruder, the current transmission channel, either quantum or classical, is discarded, and a new channel is constructed for the data transmission. Unlike classical eavesdroppings like *packet sniffing* or *man-in-the-middle* attacks, the data is transformed in QKD when an intruder intercepts the messages. This is attributed to the property of quantum entanglement and Heisenberg's uncertainty principle, which ensures that any attempt to read or measure, in quantum computing sense, disturbs the quantum state of the qubit (quantum bit), thereby invalidating the channel. So, it is evident that QPOSD can seamlessly complement the Quantum Software Development Life Cycle (QDLC) by integrating privacy and security considerations into the software development process from the start. This approach ensures that privacy and security requirements are addressed early on in the development of quantum software, leading to reduced cost overheads and a shorter time to market for the target software. By merging QPOSD with QDLC, developers can proactively mitigate potential risks and vulnerabilities, enhancing the overall robustness

```

$ python3 send_rec_msg.py
Enter 0 for 'no intruder' or 1 for 'with intruder': 0
INTRUDER_OPTION: 0
Enter the secret message (numeric/alphabetic/alphanumeric)[POC; keep reasonable length]: SECRET MSG
Sample secret message: SECRET MSG
Length of secret message: 10
Length of secret key: 400
Testing Channel with no Intruder.
Sending Messages with no Intruder
*****
Generated Random bits using QRNG
The bits Sender is going to send are: 1000111010...
The bases Sender is going to encode them in are: XXZZXZXZX...

The bases Receiver is going to decode them in are: XXXXZXZZX...
The first few bits Receiver received are: 1001100010...
The indices of the first 10 bases they share in common are: [0, 1, 7, 9, 10, 11, 13, 14, 15, 17]
Sender sent Bit 1 and Receiver received Bit 1: Bits are is the same
Sender sent Bit 0 and Receiver received Bit 0: Bits are is the same
Sender sent Bit 0 and Receiver received Bit 0: Bits are is the same
Sender sent Bit 0 and Receiver received Bit 0: Bits are is the same
Sender sent Bit 1 and Receiver received Bit 1: Bits are is the same
Sender sent Bit 1 and Receiver received Bit 1: Bits are is the same
Sender sent Bit 0 and Receiver received Bit 0: Bits are is the same
Sender sent Bit 0 and Receiver received Bit 0: Bits are is the same
Sender sent Bit 0 and Receiver received Bit 0: Bits are is the same
Sender sent Bit 1 and Receiver received Bit 1: Bits are is the same
No Intruders, Sender and Receiver seem to have the same bits!
The key is: 01111010100111011111100100001110000111000101001010111100000100100011000010010000110
The key is 113 bits long.
Using the QKD Secret Key for Encryption and Decryption
Encrypted Message: ³$¥'§' µ$
Decrypted Message: SECRET MSG
-----

```

(a) Data transmission - No Intruder.

```

$ python3 send_rec_msg.py
Enter 0 for 'no intruder' or 1 for 'with intruder': 1
INTRUDER_OPTION: 1
Enter the secret message (numeric/alphabetic/alphanumeric)[POC; keep reasonable length]: SECRET MSG
Sample secret message: SECRET MSG
Length of secret message: 10
Length of secret key: 400
Testing Channel with Intruder.
Sending messages with Intruder
*****
Generated Random bits using QRNG
The bits Sender is going to send are: 0010110010...
The bases Sender is going to encode them in are: ZZZXZXZZZ...

The bases Eve is going to decode them in are: ZZZZZXZZZ...
The first few bits Eve received are: 0010100110...
Eve creates new basis to send the bits to the Receiver.
The bases Eve is going to encode her decoy qubits in are: XZZXXZXZXZ...
The bases Receiver is going to decode them in are: XZZXZXZXZ...
The first few bits Receiver received are: 0010011110...
The indices of the first 10 bases they share in common are: [1, 2, 3, 4, 5, 6, 8, 15, 17, 18]
The first 10 bits they share in common are: ['0', '1', '0', '0', '1', '1', '1', '1', '1', '0']
Sender sent Bit 0 and Receiver received Bit 0: Bits are is the same
Sender sent Bit 1 and Receiver received Bit 1: Bits are is the same
Sender sent Bit 0 and Receiver received Bit 0: Bits are is the same
Sender sent Bit 1 and Receiver received Bit 0: Bits are different
Sender sent Bit 1 and Receiver received Bit 1: Bits are is the same
Sender sent Bit 0 and Receiver received Bit 1: Bits are different
Sender sent Bit 1 and Receiver received Bit 1: Bits are is the same
Sender sent Bit 1 and Receiver received Bit 1: Bits are is the same
Sender sent Bit 1 and Receiver received Bit 1: Bits are is the same
Sender sent Bit 0 and Receiver received Bit 0: Bits are is the same
Intruder Alert !!! at least one of the bits is different.
Communication compromised. Initiate new channel(Quantum/Classical) [Run program again].

```

(b) Data transmission - Intruder.

Figure 4.18: Data transmission security over quantum/classical channel

and security of quantum software applications while gaining useful insights and invaluable training.

Nevertheless, developing tools for Quantum Privacy Impact Analysis is one of the principal challenges for identifying privacy-critical actors and architectural necessities during system analysis. Integration of advanced Quantum Privacy Enhancing Technologies (QPET), such as Quantum Key Distribution (QKD) and Quantum Encryption (QE) into quantum software systems requires optimization for performance and compatibility. Further research is needed to devise novel verification and validation techniques tailored for quantum software, including classical and quantum components. Real-world implementations and case studies are necessary to validate quantum privacy-oriented software development methodologies across diverse application domains. Addressing these challenges will advance quantum privacy and develop secure, privacy-preserving quantum software systems.

4.3.2 Limitations/Threats to Validity

The present study has primarily three main limitations and threats to validity. Firstly, the QPOSD methodology has not yet been experimented with on real-world quantum hybrid applications and quantum development teams in any organization, which can be left as a future research endeavor. Secondly, the scenario presented above regarding data transmission was carried out on IBM Qiskit Simulator¹³, and not on the real IBM quantum backend, and also, the data transmission was not carried out on quantum channels. Practical data transmission in quantum channels might introduce noise and quantum decoherence-related issues in data quality, but the privacy and security integration in the software development process can be seen to have been established from applying QPOSD across the Quantum Software Development life cycle. Finally, the literature still does not have standardized Quantum Privacy Patterns, which is a major issue during the implementation of QPOSD, although QPOSD can still be implemented based on different bugs and vulnerability scenarios that have been proposed until now.

Having said that, universal quantum computing, relying on quantum gates and quantum annealing, operates differently in development and execution. While some patterns, bugs, and vulnerabilities may overlap between the two, their specific manifestation and mitigation strategies may differ. Therefore, where Quantum Privacy by Design Principles and Quantum Privacy Design Strategies apply seamlessly with paradigmatic changes, further investigation is necessary to assess the applicability of the proposed Quantum Privacy Patterns across diverse quantum technologies.

¹³<https://quantum-computing.ibm.com/>

4.4 Quantum Bug and Vulnerability Analysis in Quantum Software Development

The proposed methodology in subsection 3.2.6 systematically identifies and analyzes quantum software bugs that impact privacy and security, emphasizing proactive mitigation through the integration of Privacy by Design (PbD) and Security by Design (SbD) principles. By mapping bug patterns, vulnerabilities, and QDLC phases, this approach enhances the Quantum Privacy-Oriented Software Development (QPOSD) framework, providing structured insights into securing quantum software. To evaluate its applicability, five case studies were conducted, assessing the usability and feasibility of the framework across different quantum software development scenarios. The following section presents the results and discussions derived from these evaluations, highlighting key findings and implications for secure quantum software engineering.

4.4.1 Results and Discussion

The results provide a strong basis for evaluating the effectiveness of studying quantum software bugs and vulnerabilities and mapping bug patterns in implementing the QPOSD methodology. Empirical evidence from analyzed code commits supports integrating PbD and SbD principles throughout the QDLC. This section comprehensively reviews different bug types, areas of occurrence, patterns, and their alignment with vulnerabilities, QDLC phases, and PbD/SbD principles, presented via diagrams that enhance understanding of quantum software challenges and opportunities.

Figure 4.19 illustrates the distribution of bug types identified in the data extracted from GitHub open-source quantum software project code commits. Overall, 5044 code commits were extracted from the repositories based on the search keywords and applied conditions(subsection 3.2.6). Two main categories are represented as:

- *Classical Bugs*: These comprise 80% of the total, with 4056 instances. These are typical classical software system issues, such as logical errors, incorrect function calls, or incorrect input format.
- *Quantum Bugs*: Representing 20% of the total, with 988 instances, these bugs are specific to quantum computing. They include errors like quantum gate operations, improper use of quantum algorithms, and issues affecting the accuracy of quantum state manipulations.

Figure 4.20 provides a visualization of the number of bug instances across different areas of quantum software development before the filtering step. A total of 988 bugs were categorized into 32 bug patterns based on the typology and characteristics of the bugs as per the code changes. The categories are ranked by their frequency, as follows:

- *Enhancement* (213 instances): The most common category refers to

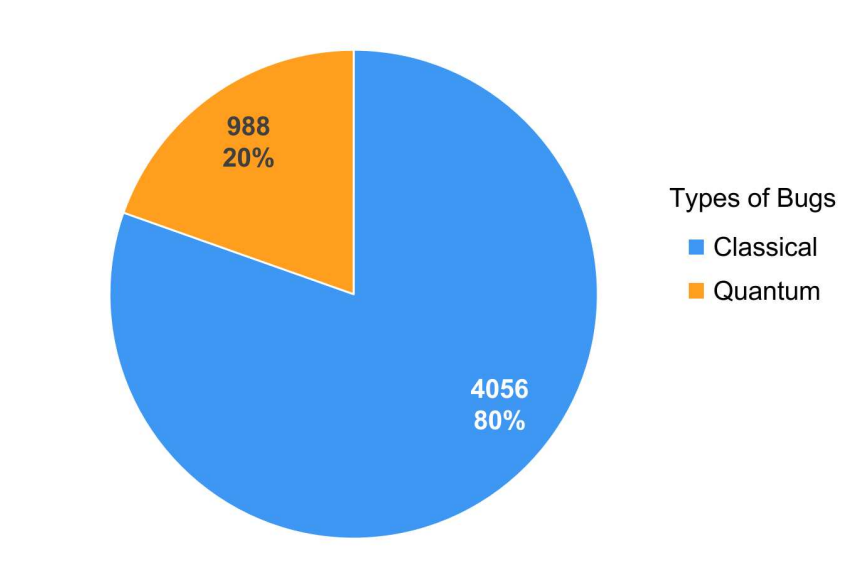


Figure 4.19: Data collected from Github

changes aimed at improving functionality or performance rather than correcting errors.

- *Documentation* (88 instances): These bugs relate to incomplete or incorrect information in project documentation.
- *Version* (70 instances), *Gate* (69), and *Operator* (61) are next concerning the frequency of Bug Patterns.

Less common but still noteworthy are API bugs (six instances), Observables (five instances), and Terminology (one instance).

After filtering and curation, Figure 4.21 depicts the distribution of 233 bug patterns, indicating which areas are most prone to issues. The bugs were analyzed and filtered based on the methodology, and only quantum-specific bug fixes were allowed. Key findings include:

- *Logic Bugs* (35 instances): The highest frequency, representing errors in coding logic that affect functionality.
- *Calculation and Circuit Bugs* (24 instances each): These are prominent, involving errors in mathematical calculations or circuit-level operations.
- *Gate Bugs* (20 instances): Errors related to quantum gates, which are fundamental to quantum algorithms, are frequent, underscoring their complexity.
- *Operator and Qubit/Qutrit Bugs* (16 and 13 instances, respectively): These relate to handling quantum operations or qubit management, which present significant challenges.

Bug patterns like state, test, backend, and measurement are less frequent but still notable. Issues related to Syntax, Configuration, and Compatibility are

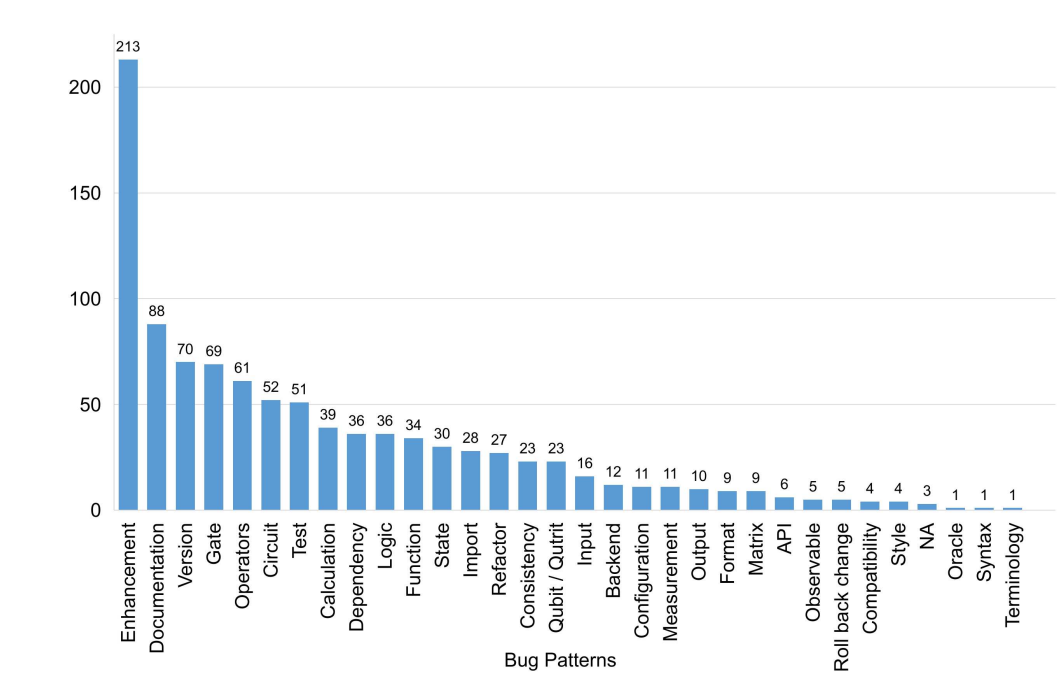


Figure 4.20: Code Commits before Filtering

rarer but critical, as they can affect the overall system functionality. This reveals that quantum-specific components like logic, calculations, circuits, and gates are more susceptible to errors, highlighting areas that need more rigorous debugging and verification.

The Sankey diagram Figure 4.22 illustrates the relationship between bug patterns and the vulnerabilities they can introduce, showing how different types of bugs contribute to weaknesses in quantum software systems.

- *Logic Bugs*: Contribute to multiple vulnerability patterns, particularly those affecting accuracy, states, gate, and circuit operations.
- *Gate and Circuit Bugs*: Significantly impact operational vulnerabilities and also impact vulnerabilities like code maintenance and configuration.
- *Calculation Bugs*: Impacts mostly mathematical vulnerabilities like the accuracy of operators, gradient and probability calculations, transformation accuracy, and Hamiltonian management.
- *Qubit/Qutrit Bugs*: Mostly leads to qubit management and handling vulnerabilities.

Some other patterns, such as operator, output, consistency, testing, and state, contribute to various important vulnerabilities, such as state and operator accuracy, initialization issues, configuration handling, and code efficiency and maintenance.

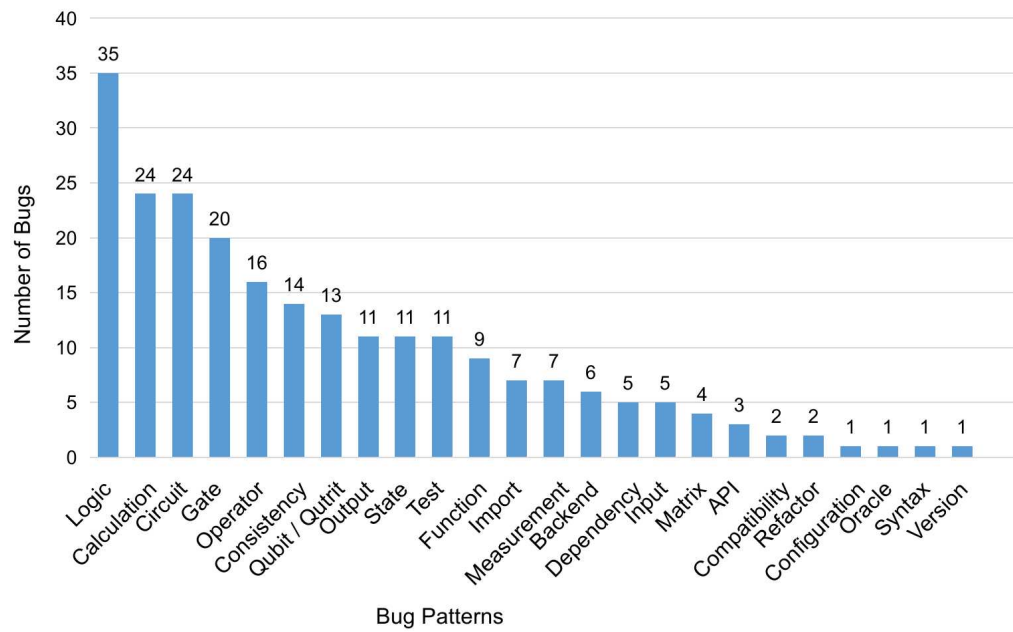


Figure 4.21: Bug Patterns

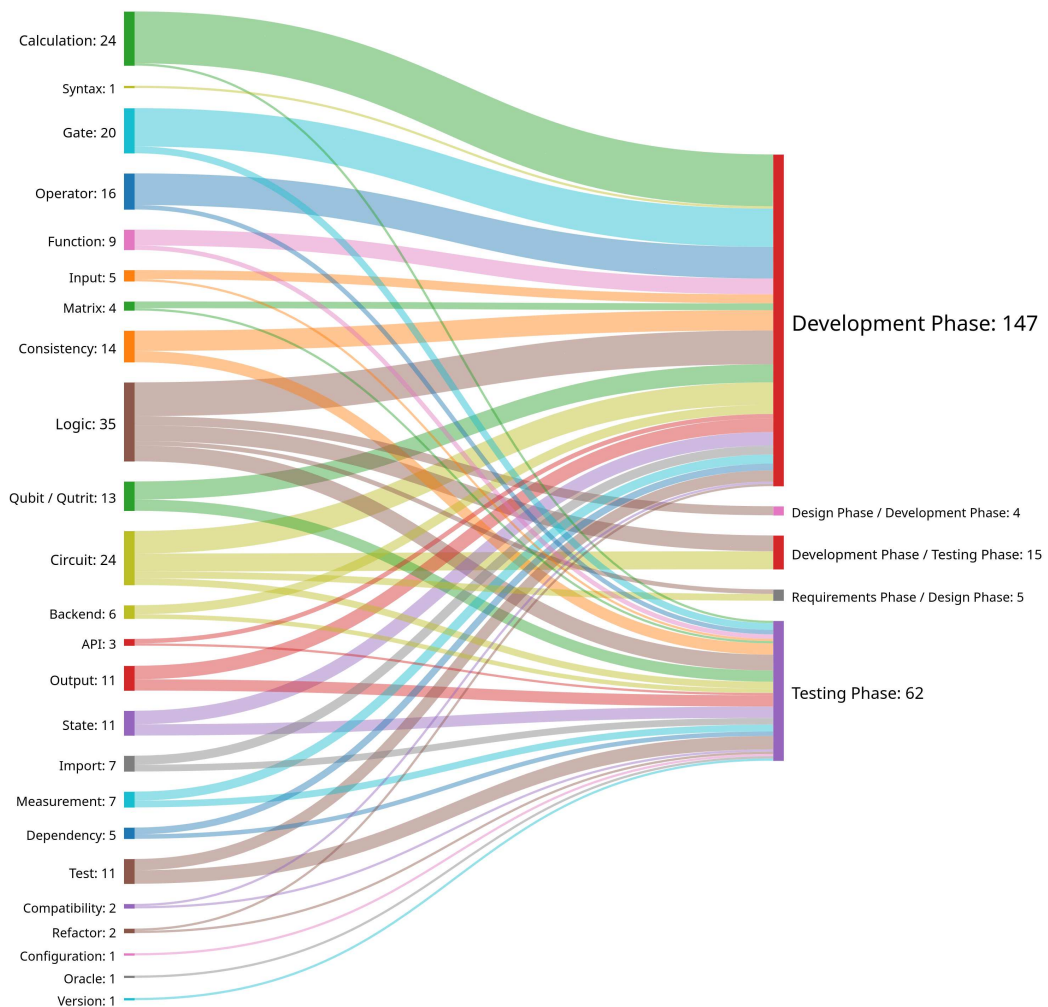


Figure 4.23: Mapping of Bugs Patterns with QDLC Phases

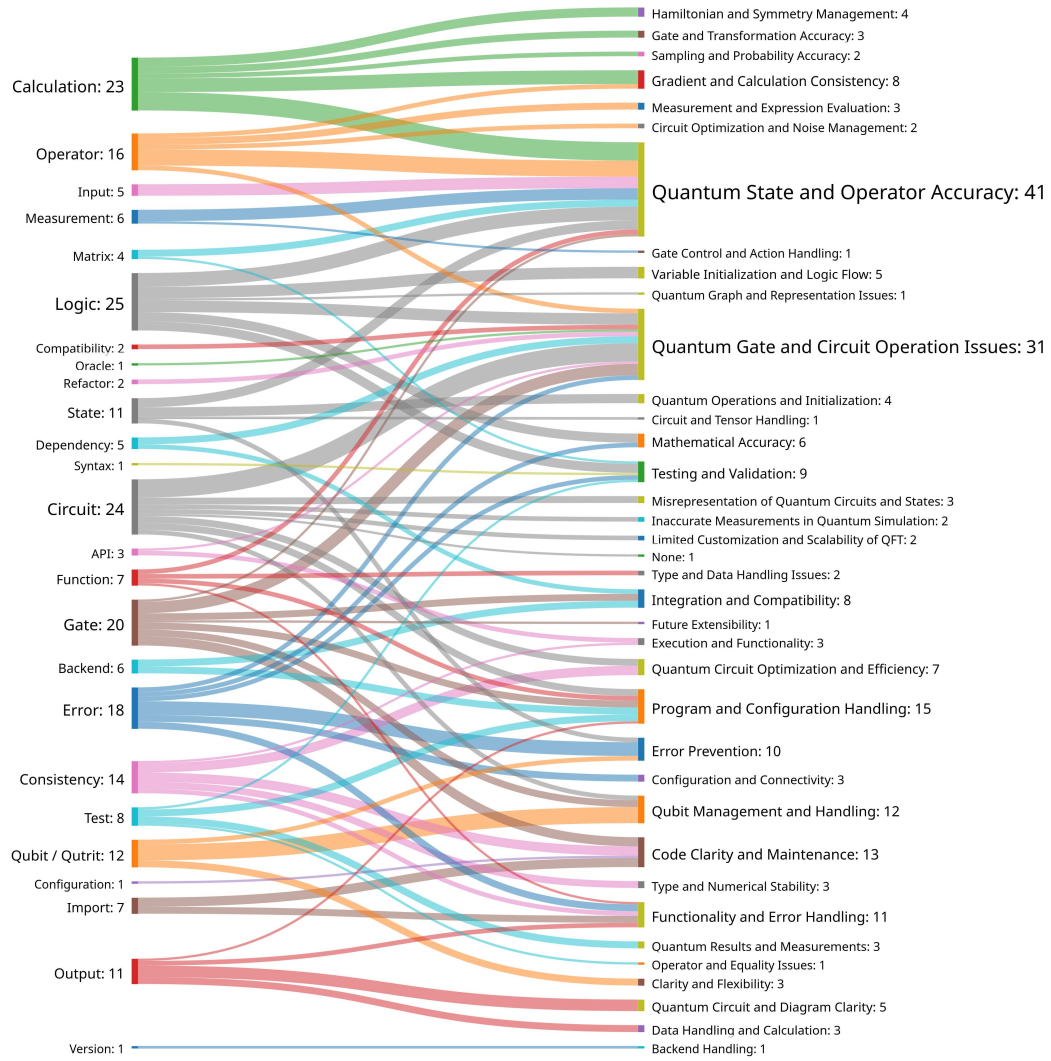


Figure 4.22: Mapping of Bugs Patterns with Vulnerability Patterns

The mapping between the bug patterns and the Quantum Software Development Lifecycle (QDLC) phases is presented in Figure 4.23. This mapping shows the life cycle phases during which the bugs denoted by the bug patterns are introduced into the code. The key findings include:

- *Development Phase*: Responsible for most bugs, indicating that most issues originate in the initial code construction. This suggests a need for better development practices to reduce vulnerabilities early on.
- *Testing Phase*: A substantial number of bugs, highlighting the importance of comprehensive testing and validation.
- *Requirements / Design Phase*: Although fewer, some bugs have also been attributed to these phases, suggesting the importance of privacy and security-oriented quantum software development.

This mapping helps pinpoint when specific bug patterns are most likely to be

introduced, allowing the development teams to focus their quality assurance efforts on the most vulnerable phases of the quantum software development lifecycle.

Figure 4.24 expands on the previous one, mapping QSDLC phases to the vulnerabilities in each phase. The key observations include:

- *Development Phase*: It is the primary contributor to quantum-specific challenges, accounting for 147 instances, contributing to issues related to quantum state and operator accuracies, gate and circuit operation issues, mathematical accuracies, code efficiency, and configuration issues, among others.
- *Testing Phase*: With 62 instances, these vulnerabilities are frequently linked to issues such as quantum gate accuracy, optimization challenges, clarity in quantum circuit representation, and quantum testing-related issues.

The interconnected nature of vulnerabilities is evident, with issues originating in the development phase cascading into other stages, such as testing and implementation. This highlights the importance of addressing vulnerabilities early to prevent them from dispersing throughout the lifecycle. Implementing stringent quality checks and integrating privacy and security measures from the design phase can help minimize quantum vulnerabilities throughout the QDLC.

Figure 4.25 provides a comprehensive view of how different phases of the QDLC align with specific Privacy by Design (PbD) and Security by Design (SbD) principles. It illustrates which phases are most involved in implementing these principles and highlights areas where more attention is needed to enhance privacy and security in quantum software development.

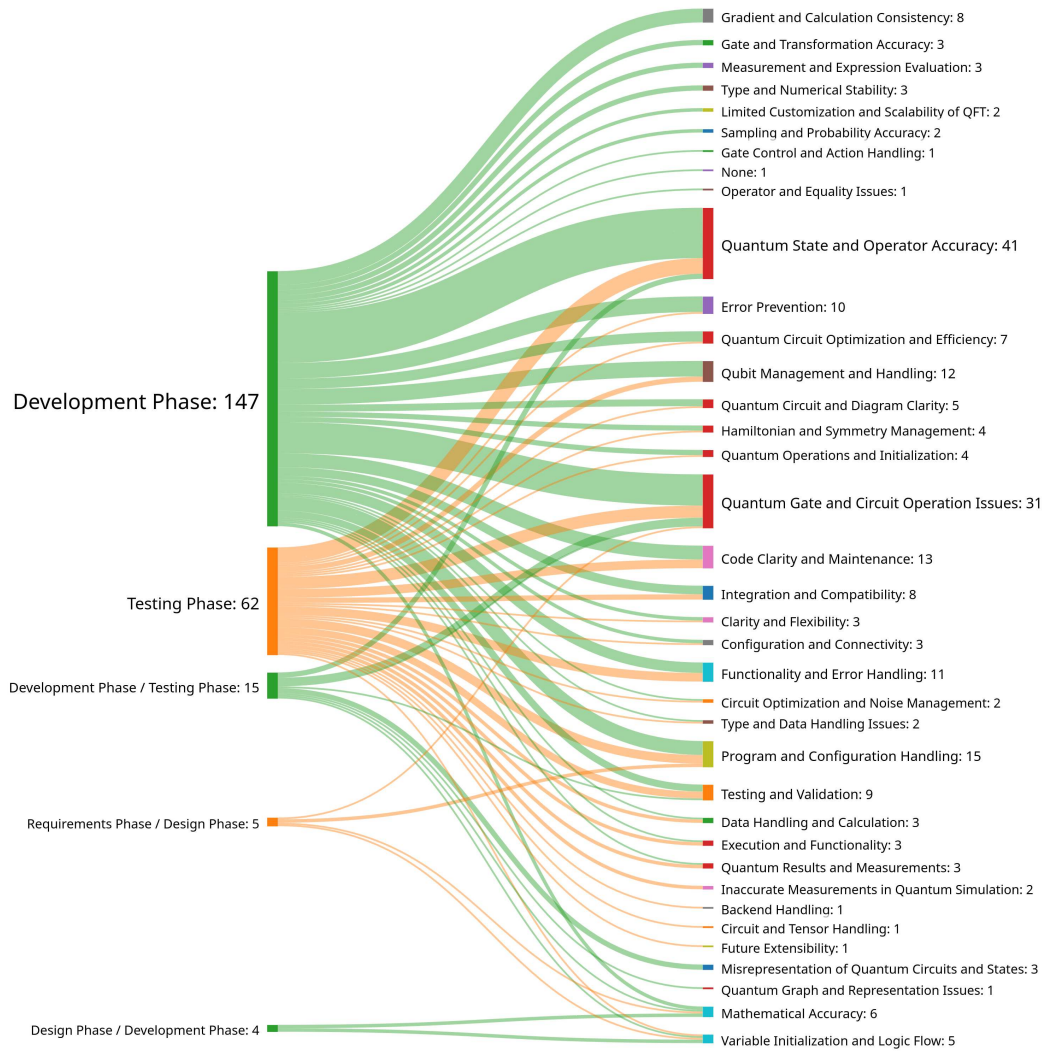


Figure 4.24: Mapping of QDLC phases with Vulnerability Patterns

The key insights include:

- *Development Phase*: The development phase is strongly connected to multiple PbD and SbD principles, with 147 instances indicating significant challenges in aligning with privacy and security best practices during code construction.
- *Testing Phase*: Accounting for 62 instances suggests that this stage is important in ensuring that privacy and security measures are properly validated and enforced throughout the QDLC lifecycle.
- *P5 (End-to-End Security)*: P5 or End-to-End Security – Full Lifecycle Protection is the most frequently represented principle, with 87 instances, highlighting the need for maintaining strong security from start to finish throughout the QDLC.
- *S2 (Establish secure defaults)*: The frequent representation of S2 em-

phasizes ensuring that all default settings are secure in the application, especially during phases like development, where changes are more frequent and could inadvertently introduce risks.

This analysis emphasizes integrating PbD and SbD principles throughout all QSDLC stages to ensure robust quantum software development. Both development and testing phases reveal inconsistent application of PbD and SbD principles, indicating that stronger adherence to these principles is required to improve privacy and security during development and testing.

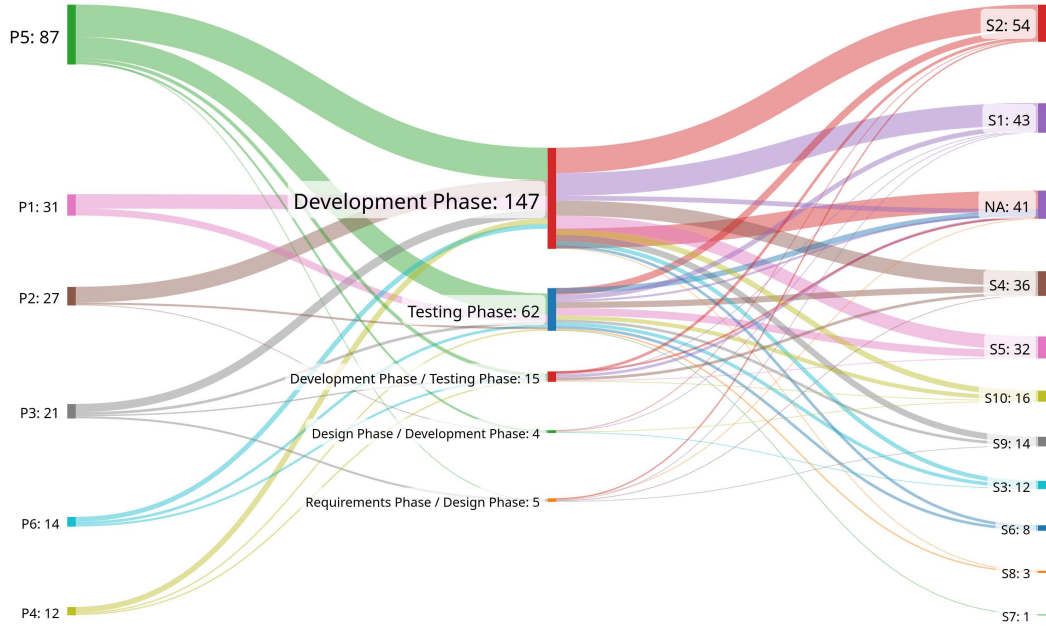


Figure 4.25: Mapping of QDLC phases with PbD and SbD Principles

The heatmap in Figure 4.26 illustrates the relationship between various quantum bug patterns and their violations of Privacy by Design (PbD) principles, offering insights into privacy compliance challenges in quantum software development. The principal points include:

- *Logic Bugs*: Logic Bugs show the highest frequency of violations, particularly against P5 (End-to-End Security) with 19 instances, suggesting the importance of addressing logic issues for maintaining strong security.
- *Gate Bugs*: Gate Bugs are frequent, especially linked to P5 and P2 (Privacy as the Default Setting), highlighting gate operations as a vulnerable area that needs attention for maintaining privacy and security.
- *Calculation Bugs*: Calculation Bugs also pose significant concerns, with 10 instances affecting P5, underscoring the need for secure implementation of quantum calculations.
- *Consistency Bugs*: Consistency Bugs frequently violate P1 (Proactive not Reactive; Preventive not Remedial), emphasizing the importance of

consistency in ensuring proactive privacy measures.

- *State Management Bugs* and *Test Bugs*: These bugs contribute notably to violations of P2 and P5, showing that improving state handling and implementing failsafe testing mechanisms are crucial for better privacy compliance.

The heatmap identifies which bug types pose the greatest risks to privacy compliance, providing actionable insights for enhancing quality across the software lifecycle. The findings from the PbD heatmap highlight areas where privacy measures need strengthening in quantum software development. By focusing on bug patterns that frequently violate P5 (End-to-End Security) and P2 (Privacy as the Default Setting), developers can ensure privacy protections are systematically integrated. Prioritizing gate operations, logic management, and calculation handling will help improve compliance with privacy principles, ultimately enhancing user trust and reducing the risk of privacy issues in quantum systems.

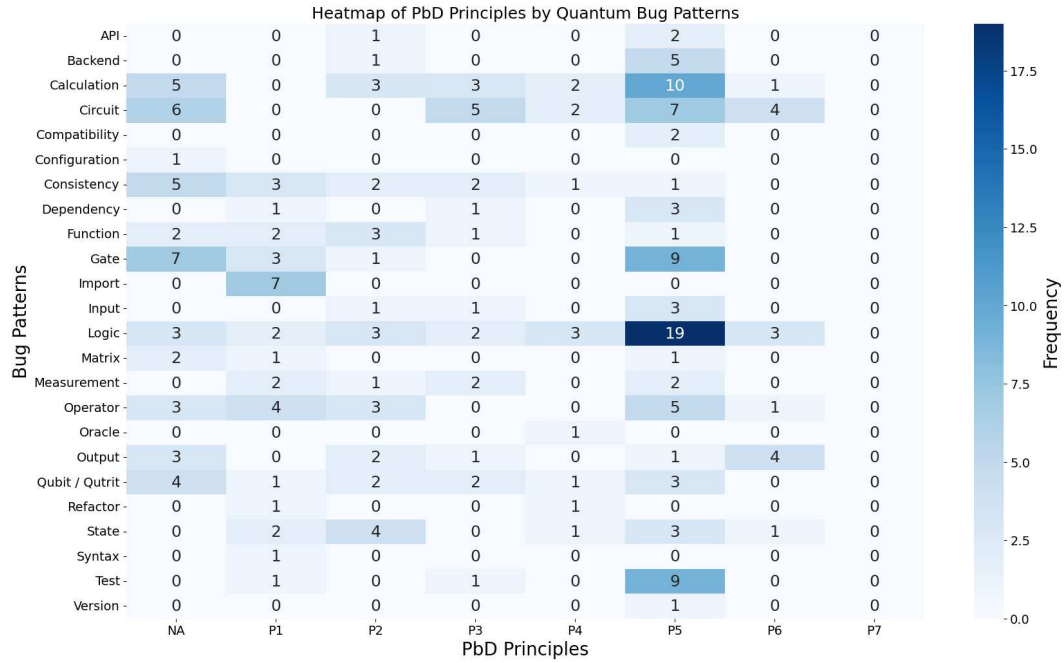


Figure 4.26: Heatmap of PbD in Bug Patterns

Figure 4.27 illustrates how different quantum bug patterns correspond to violations of Security by Design (SbD) principles. It offers insights into which bug types are most commonly associated with non-compliance, helping to prioritize areas for security enhancement. The key observations include:

- *Logic Bugs*: Logic bugs are highly associated with violations across several SbD principles, particularly S5 (Fail securely), with six instances, and S1 (Minimize Attack Surface Area), with 11 instances. This suggests strengthening defensive measures and reducing attack surfaces when handling logic.

- *Calculation Bugs*: These have eight instances linked to S1 and seven to S2 (Establish Secure Defaults), indicating a need for secure configurations and minimizing risks during quantum calculations.
- *Gate Bugs*: Gate-related bugs frequently violate S1, S2, and S4 (Defence in depth), with 5, 6, and 5 instances, respectively. This underscores the challenges associated with quantum gate operations and the importance of robust defenses.
- *Import Bugs*: 7 instances of import bugs violate S2, suggesting that improper imports can lead to insecure defaults, which need careful management.
- *Operator and Qubit Bugs*: These are often associated with S4 and S5 (Fail Securely), highlighting issues in safely managing quantum outputs and operations.
- *Testing Bugs*: Testing-related bugs have three instances tied to S1 and S5, indicating a need for secure testing processes to ensure resilience.

These findings have significant practical implications for quantum software development. Ensuring compliance with S1, S2, S4, and S5 at different stages of development can reduce the risk of security breaches and operational failures. Moreover, the emphasis on secure defaults, defense in depth, and secure testing helps facilitate a proactive security culture, ultimately leading to more robust quantum systems that can withstand evolving threats.

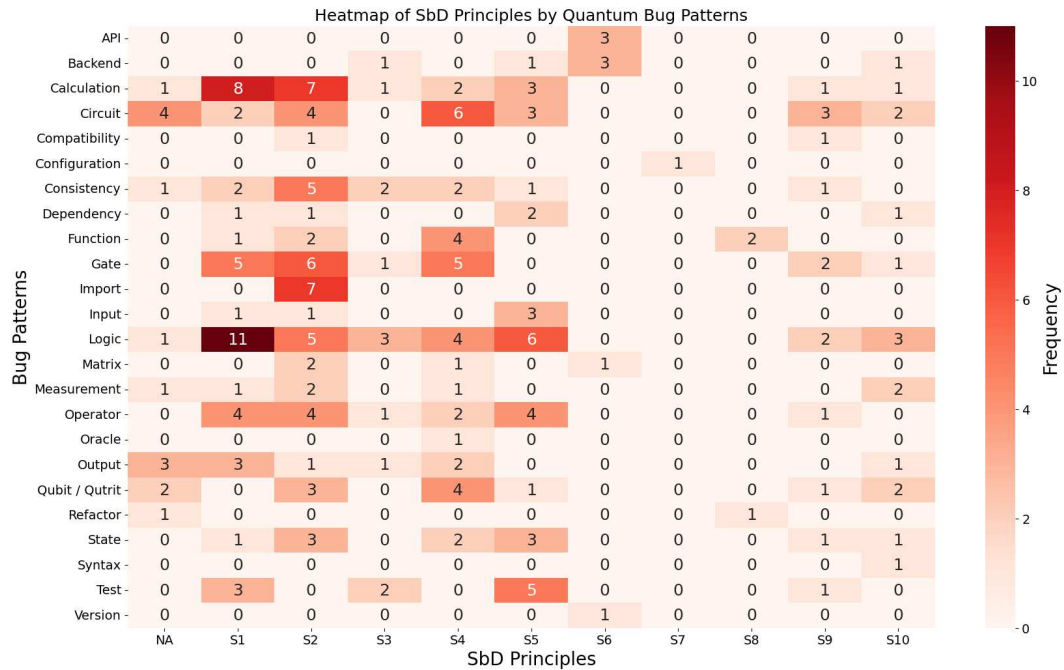


Figure 4.27: Heatmap of SbD in Bug Patterns

In Summary, the analysis reveals significant relationships between bug patterns, vulnerabilities, QDLC phases, and PbD/SbD principles. For instance, the

development phase contributes heavily to vulnerabilities, especially those tied to *End-to-End Security*, *Minimization of attack Surface Area*, or *Establishing secure defaults* underlining the need for early, proactive measures in the Quantum Development Lifecycle(QDLC). These findings offer crucial insights for development teams to preemptively address issues throughout the Quantum Development Life Cycle (QDLC), minimizing vulnerabilities before they become problems in the product. Moreover, these insights can help implement Quantum Privacy-Oriented Software Development (QPOSD) [65] by leveraging the interrelationships among quantum bugs, bug patterns, and vulnerabilities to build secure and privacy-conscious quantum systems.

4.4.2 Case Studies

In this section, we explore five case studies drawn from different quantum software projects on GitHub. Our goal is to showcase the practical implementation of privacy and security principles using real code examples. For each case, we created a testing scenario to assess the code both before and after applying the proposed fix. The case studies illustrate the value of integrating privacy and security considerations early in the development process, ultimately enhancing software reliability and trustworthiness.

Case Study 1: Logic check for unsupported orbital configurations

In the before-fix version, error messages were generic, failing to clarify unsupported scenarios specifically, which could lead to misinterpretation. The after-fix implementation introduces precise error messages that provide context, clearly indicating why certain configurations are invalid. We present a screenshot (Figure 4.28) demonstrating the enhanced validation in action, showcasing how the updated logic effectively flags unsupported orbital configurations during the development phase, providing developers with immediate and actionable feedback. (UUID - [https://github.com/qiskit-community/qiskit-nature/...](https://github.com/qiskit-community/qiskit-nature/))

Test Scenarios

- *Scenario 1:* Both particle counts are equal to `num_spatial_orbitals`, which should raise the first `ValueError` in the updated logic.
- *Scenario 2:* One of the particle counts exceeds `num_spatial_orbitals`, triggering the second `ValueError` in the updated logic.

Expected Output:

- *Before fix:* Scenario 1 might not raise an error or raise generic error as it doesn't handle fully occupied configurations explicitly, whereas Scenario 2 should raise an error due to the particle count exceeding the orbitals.
- *After fix:* Both scenarios should raise appropriate errors with specific messages according to the updated checks.

```

$ python3 commtest_4877.
py
Testing before fix (Original Logic):
Running Test: All particles less than spatial orbitals
Result: No errors.
-----
Running Test: Any particle exceeds spatial orbitals
Result: Caught ValueError - The number of spatial orbitals 5 must be greater than number of particles of
any spin kind [4, 6].
-----
Running Test: All particles equal to spatial orbitals
Result: Caught ValueError - The number of spatial orbitals 5 must be greater than number of particles of
any spin kind [5, 5].
-----

Testing after fix (Updated Logic):
Running Test: All particles less than spatial orbitals
Result: No errors.
-----
Running Test: Any particle exceeds spatial orbitals
Result: Caught ValueError - The number of spatial orbitals 5 must be greater than number of particles of
any spin kind [4, 6].
-----
Running Test: All particles equal to spatial orbitals
Result: Caught ValueError - UCC calculations for fully occupied alpha and beta orbitals is still not imp
lemented. The current system contains 5 orbitals for [5, 5] (alpha, beta) particles.
-----

```

Figure 4.28: Logic check for unsupported orbital configurations

PbD Principle P5: End-to-End Security

The fix ensures that error handling is robust and covers all possible configuration issues, which helps prevent issues from occurring at runtime.

SbD Principle S1: Minimize Attack Surface

By checking for unsupported configurations early, the fix prevents the system from reaching potentially insecure states due to invalid configurations. The code change fixes error handling by adding validation checks for specific configurations in the Unitary Coupled Cluster (UCC) circuit.

This change enhances code robustness during the development phase by adding precise validation to flag unsupported configurations with a clear *ValueError*. By proactively preventing incorrect orbital setups, it allows developers to efficiently identify and resolve issues, contributing to a more reliable UCC ansatz. These checks ensure the code handles all possible scenarios gracefully, improving security, functionality, and overall reliability.

Case Study 2: Dependency support fix for custom gate sets

The bug fix introduces a *basis_gates* parameter to limit allowed gates, preventing unsupported gates from being used. It also inlines initialization gates and converts control-flow operations into secure "mark and jump" instructions, enhancing configurability and security in circuit compilation. We provide a screenshot (Figure 4.29) showing the application of the *basis_gates* parameter in action, demonstrating how unsupported gates are filtered during circuit assembly. (UUID - [https://github.com/Qiskit/qiskit-aer/...](https://github.com/Qiskit/qiskit-aer/))

Test Scenarios

- *Scenario 1:* Basic compilation without any restrictions on gates or inline

initialization.

- *Scenario 2*: Compilation restricted to only h and x gates using *basis_gates*. The circuit contains a cx gate, so this should fail.
- *Scenario 3*: Compilation allowing h , x , and cx gates using *basis_gates*, which should succeed.
- *Scenario 4*: Full test with all features enabled, including inline initialization and control-flow handling, but with restricted gates to verify flexibility.
- *Scenario 5*: Full test with all gates allowed, verifying the functionality of inline initialization and control-flow conversion.

Expected Output

- Scenario 1: Succeeds with no gate restrictions.
- Scenario 2: Fails because cx gate is not in *basis_gates*.
- Scenario 3: Succeeds as all gates in the circuit are permitted.
- Scenario 4: Demonstrates inline initialization and control-flow handling, but will fail if unsupported gates are used.
- Scenario 5: Succeeds with all features enabled and no gate restrictions.

Generalized Fix

- *Basis Gate Filtering*: Added support for an optional *basis_gates* parameter to the *assemble_circuit* method. This enables defining an allowed set of gates for the circuit, which enhances control and ensures incompatible gates are filtered out. This results in more precise gate usage during compilation.
- *Inline Initialization and Control Flow Conversion*: The fix includes inlining gate initialization and converting high-level control-flow instructions into simpler ones. This approach allows the compiler to manage complex circuit features more effectively, making the circuit compatible with backends that do not support high-level features.

PbD Principle P3 - Privacy Embedded into Design

Introducing *basis_gates* and inline initialization helps embed privacy by allowing the restriction of circuits to necessary operations, reducing unnecessary exposure during compilation.

```

$ python3 commtest_3523a
.py
=== BEFORE FIX ===
Running Test: Before fix - No basis gate restriction
Compiling circuit with basis gates: set()
Result: Success
-----
Running Test: Before fix - Restricted to 'h' and 'x' gates
Compiling circuit with basis gates: set()
Result: Success
-----
Running Test: Before fix - All gates allowed
Compiling circuit with basis gates: set()
Result: Success
-----
Running Test: Before fix - Full test with restricted gates
Skipping inline initialization (not supported in before fix)
Skipping control flow conversion (not supported in before fix)
Compiling circuit with basis gates: set()
Result: Success
-----
Running Test: Before fix - Full test with all gates allowed
Skipping inline initialization (not supported in before fix)
Skipping control flow conversion (not supported in before fix)
Compiling circuit with basis gates: set()
Result: Success
-----
=== AFTER FIX ===
Running Test: After fix - No basis gate restriction
Compiling circuit with basis gates: set()
Result: Success
-----
Running Test: After fix - Restricted to 'h' and 'x' gates
Error: Gate 'cx' not in allowed basis gates: {'x', 'h'}
Result: Failure
-----
Running Test: After fix - All gates allowed
Compiling circuit with basis gates: {'barrier', 'initialize', 'x', 'cx', 'h'}
Result: Success
-----
Running Test: After fix - Full test with restricted gates
Inlining initialize gate in circuit 'circuit-158'
Error: Gate 'cx' not in allowed basis gates: {'x', 'h'}
Result: Failure
-----
Running Test: After fix - Full test with all gates allowed
Inlining initialize gate in circuit 'circuit-158'
Compiling circuit with basis gates: {'barrier', 'h', 'initialize', 'x', 'cx'}
Result: Success
-----

```

Figure 4.29: Dependency support fix for custom gate sets

SbD Principle S1 - Minimize Attack Surface Area

Limiting the allowed gates and simplifying control flow helps reduce the attack surface, decreasing vulnerability risks by focusing only on essential operations.

The updated functionality allows specifying a restricted set of basis gates when assembling quantum circuits, adding flexibility and accuracy to the process. Developers can now control which gates are used, ensuring compatibility with the Aer simulator. This change is crucial in the development phase to meet functional requirements and ensure robustness by controlling circuit configurations precisely. The added flexibility aids in avoiding unsupported gate setups, improving both the code's reliability and maintainability.

Case Study 3: Circuit handling fix for better measurement and reuse

The code commit resolved an issue related to quantum circuit handling in

the *SamplerQNN* class. Previously, the class copied the circuit and applied measurements if it lacked classical bits, which could lead to discrepancies, especially for reparameterized circuits. The fix introduces *_org_circuit* to preserve the original circuit, while a reparameterized version (*_circuit*) is created only when needed for calculations, ensuring consistency throughout the computation process. In the case study, a screenshot is provided to show how the *_org_circuit* remains intact, while *_circuit* is used for processing, effectively demonstrating the fix's impact on maintaining the original circuit's state (Figure 4.30). (UUID - [https://github.com/qiskit-community/qiskit-machine-learning/...](https://github.com/qiskit-community/qiskit-machine-learning/))

Test Scenarios

- Initial Circuit without Measurement: Display the original circuit to verify no modifications or measurements are present.
- Circuit After Measurement Addition: Check how measurements are added without affecting the original circuit.
- Circuit Modification: Apply a Hadamard gate to the circuit, verifying that modifications affect only the reparameterized circuit while the original remains intact.
- Output Shape Calculation: Confirm that the output shape calculation uses the correct circuit based on the number of qubits.

Expected Output:

- Initial Circuit without Measurement
 - Before Fix: The circuit does not preserve its original state before modifications.
 - After Fix: *_org_circuit* retains the original, unaltered circuit.
- Circuit After Measurement Addition
 - Before Fix: Measurements are added to the only reference available, altering the original circuit.
 - After Fix: Measurements are added to *_circuit*, but *_org_circuit* remains unchanged.
- Circuit Modification:
 - Before Fix: Modifications apply directly to the single reference available.
 - After Fix: Changes affect *_circuit* only, leaving *_org_circuit* unchanged.
- Output Shape Calculation: Before and After Fix: The output shape is calculated based on whether the circuit is measured or unmeasured.

PbD Principle P3 - Privacy Embedded into Design

The change enhances circuit handling by embedding privacy-conscious design, keeping the original circuit intact. Treating the original circuit as immutable minimizes risks of unintended manipulation, aligning with P3 by maintaining data integrity.

SbD Principle S2 - Establish Secure Defaults

Preserving the unmodified circuit as *_org_circuit* establishes a secure default, ensuring the original data is not inadvertently changed. This aligns with S2 by reducing the risk of unintentional modifications.

This update improves circuit handling in Qiskit Machine Learning by introducing a dedicated *_org_circuit* attribute, which preserves the original configuration while allowing changes to be made on a reparameterized version (*_circuit*). This separation is crucial during the Requirements Phase, where defining and preserving initial configurations is essential to ensure circuit integrity for later design and testing stages.

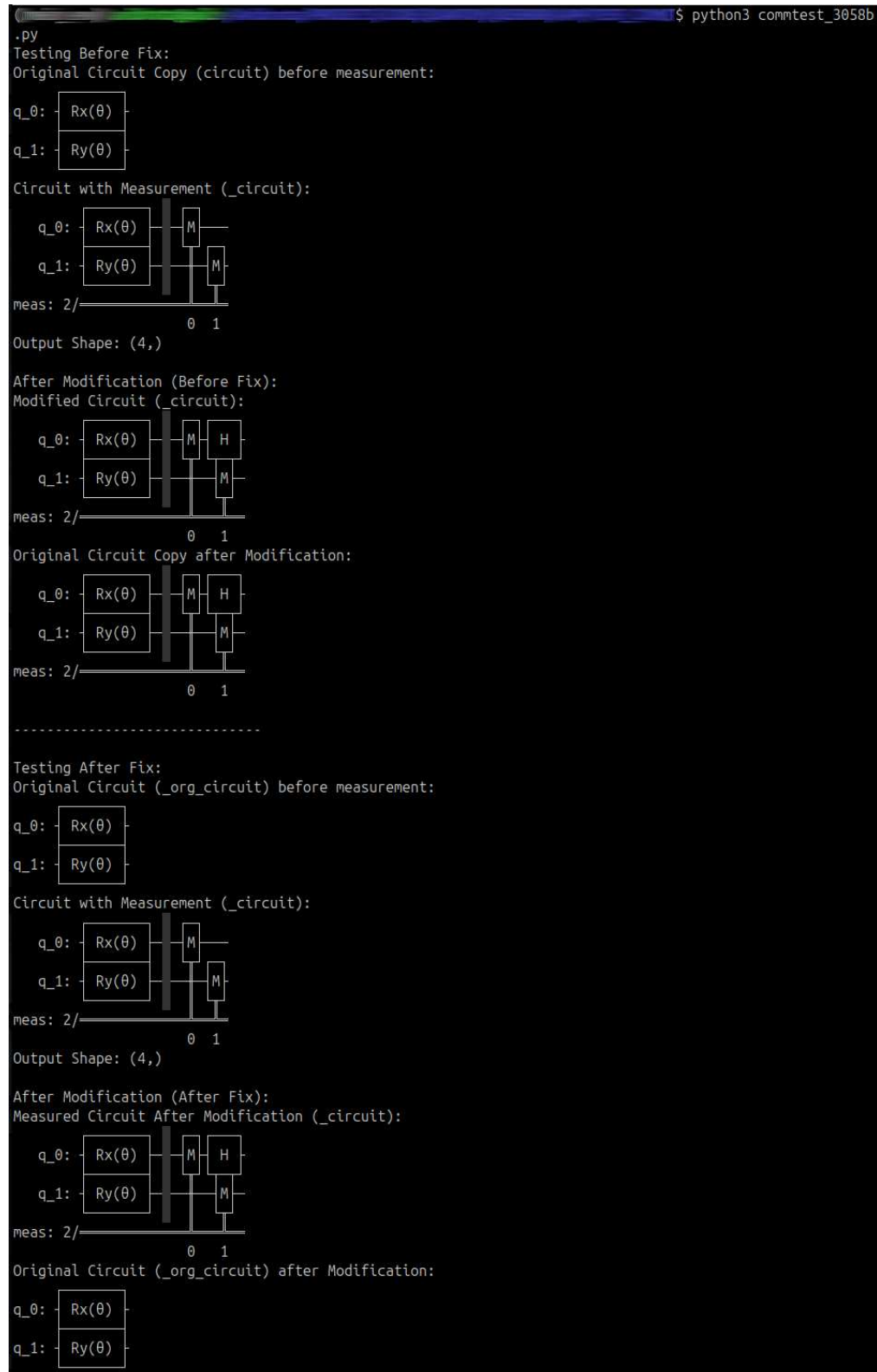


Figure 4.30: Circuit handling fix for better measurement and reuse

Case Study 4: Circuit optimization fix with consistent functional transforma-

tions

The changes involve updating the syntax for Cirq functions used to optimize and process quantum circuits. Specifically, the code now uses functions that return a modified circuit rather than modifying it in place. In the case study, a screenshot (Figure 4.31) is included to demonstrate the functional transformation of circuits before and after applying the fixes, showing the clear distinction in circuit handling. (UUID - [https://github.com/quantumlib/ReCirq/...](https://github.com/quantumlib/ReCirq/))

Test Scenarios

- Scenario 1: Initial Circuit without Optimization; display the original circuit before any optimizations.
- Scenario 2: Optimize the Circuit (Merge single-qubit gates, remove empty moments, align measurement at the end)
- Scenario 1: The circuit remains unchanged before any optimizations, both before and after the fix.
- Scenario 2: Gates are merged using a functional transformation, empty moments are removed, and measurements are aligned correctly; the circuit is fully optimized.

PbD Principle P5 - End-to-End Security – Full Lifecycle Protection

The updated functions ensure consistent circuit optimizations throughout the development process, minimizing errors during both compilation and runtime phases.

SbD Principle S2 - Establish Secure Defaults

The revised methods set secure optimization defaults during compilation, ensuring reliable circuit processing using secure configurations.

SbD Principle S4 - Defense in Depth

Switching from in-place modifications to functional transformations improves control over quantum circuit states. This aligns with a layered security approach, adding verification and reducing unexpected behaviors, making circuits more resilient to errors and improves robustness during execution.

The Testing Phase linked changes provide a structured framework for validating functions and ensuring circuit integrity before deployment. This approach reduces runtime errors and ensures consistent quantum gate operations and measurements, leading to more reliable quantum systems.

Case Study 5: Fix logic for improved circuit decomposition control

The fix enhances the compilation transformation process by replacing the expand method with the decompose function and refining the stopping criteria. In the case study, we provide a screenshot (Figure 4.32) demonstrating the circuit transformations before and after applying the changes, highlighting the difference in decomposition handling. (UUID - [https://github.com/PennyLaneAI/pennylane/...](https://github.com/PennyLaneAI/pennylane/))

Test Scenarios

```

5029.py
$ python3 commtest_5028_
=== BEFORE FIX ===
Original Circuit (Before Fix):
(0, 0): —X—@—M('q0')—
          |
(0, 1): —H—X—M('q1')—
Optimized Circuit (Before Fix):
(0, 0): —X—@—M('q0')—
          |
(0, 1): —H—X—M('q1')—
Measurements are at the end of the circuit

=== AFTER FIX ===
Original Circuit (After Fix):
(0, 0): —X—@—M('q0')—
          |
(0, 1): —H—X—M('q1')—
Optimized Circuit (After Fix):
(0, 0): —PhXZ(a=-1,x=1,z=0)—@—M('q0')—
                              |
(0, 1): —PhXZ(a=-0.5,x=0.5,z=-1)—X—M('q1')—
Measurements are at the end of the circuit

```

Figure 4.31: Circuit optimization fix with consistent functional transformations

- Scenario 1: Compile a Circuit Using a Custom Basis Set - Compile a circuit using a basis set that only allows certain gates (e.g., CNOT, RX, RY). Check if the compiler respects this basis set and the final circuit contains only the specified basis gates, with others decomposed appropriately, while undecomposable gates are left intact.

Expected Output

- Original Circuit Creation: Create the original circuit using $qfunc(x, y, z)$.
- Before Fix: The compiled circuit remains unchanged, reflecting no optimizations.
- After Fix: The compiled circuit shows an optimized structure, possibly with reduced or reordered gates.

PbD Principle P2 - Privacy as the Default Setting This fix ensures that circuit compilation respects privacy by restricting transformations to a defined basis set and stopping decomposition for operators that should remain unchanged, offering control and predictability over the process.

SbD Principle S2 - Establish Secure Defaults By introducing a stopping condition, the fix sets a secure compilation default, safeguarding operations that cannot be decomposed, thus restricting modifications to those that are safe.

During the Design Phase, this fix refines circuit compilation at a foundational level, influencing the behavior of all circuits compiled with this method. By adding conditions and constraints, it ensures that design requirements for controlled decompositions are consistently met.

These findings offer crucial insights for development teams to preemptively

```

$ python3 commtest_540.p
y
Original Circuit:
0: —H—RX(0.40)—X—RX(0.20)—X—| <Z>
1: —H—X—L—X—L—X—L—|
2: —H—RZ(0.40)—RZ(-0.40)—RX(0.30)—Y—|
Results: expval(Z(0))

=== Before Fix ===
Compiled Circuit:
0: —H—RX—X—RX—X—| <Z>
1: —H—X—L—X—L—X—L—|
2: —H—RZ—RZ—RX—Y—|
Results: [(0.0,)]

=== After Fix ===
Compiled Circuit:
0: —H—RX—| <Z>
1: —H—X—X—|
2: —H—X—RX—Y—Y—|
Results: [(0.0,)]

```

Figure 4.32: Fix logic for improved circuit decomposition control

address issues throughout the Quantum Development Life Cycle (QDLC), minimizing vulnerabilities before they become problems in the product. Moreover, these insights can help implement Quantum Privacy-Oriented Software Development (QPOSD) by leveraging the interrelationships among quantum bugs, bug patterns, and vulnerabilities to build secure and privacy-conscious quantum-hybrid systems.

Chapter 5

Conclusion, Limitation, and Future Work

This chapter presents the conclusions of the aforementioned models, frameworks and experiments based on them, and delineates its limitations and probable future research directions.

5.1 Conclusion

Quantum Software Engineering (QSE) represents an emergent and rapidly evolving discipline that leverages the foundational principles of quantum mechanics to offer fundamentally distinct and efficient resolutions to extant challenges, utilizing quantum computing through established software engineering methodologies for the development, validation, and maintenance of quantum software systems. Nevertheless, the swift evolution of these paradigms is accompanied by intrinsic security and privacy dilemmas. As a wholly novel paradigm, Quantum Computing is exceptionally susceptible to errors and hardware-related issues, such as side-channel energy leakage, shuttling, and decoherence. Additionally, the adaptation of quantum technologies to align with classical systems poses significant challenges, rendering them vulnerable to cyber threats. Consequently, this thesis seeks to investigate and critically analyze the current QSE frameworks concerning security and privacy while proposing innovative strategies to mitigate risks and threats associated with quantum and hybrid systems.

We presented the frameworks and models developed in the purview of the comprehensive investigation in the area of implementing QSE for security and privacy, we well as security and privacy in QSE following the Research Questions 1.1, 1.2, and 1.3. We presented an extensive systematic literature review investigating the existing empirical evidence regarding the definition and implementation of Quantum PbD and SbD that fulfills the guidelines of privacy and security provided in the Talavera Manifesto [59]. We analyzed the existing state-of-the-art Quantum PbD and SbD in the area of QSE and identified the challenges in the area. We collected empirical evidence from

all studies in Quantum Software Engineering published until now. The most popular empirical strategy was experimentation, the technology/methodology was Quantum Key Distribution, the software engineering implementation phase was design and development, and the applied research area was computer science. For researchers and practitioners, these observations likely stem from the need to test and validate hypotheses, methods, or technologies. The emphasis on the design and development phases of software engineering, especially within computer science, signifies the attention to the area where most practical and theoretical efforts are concentrated.

The results of this step led to establish QuIPS framework, a structured approach to integrating Privacy by Design (PbD) and Security by Design (SbD) in QSE, emphasizing their role in the Quantum Development Lifecycle (QDLC). QuIPS provides a roadmap for developing quantum-secure software that aligns with emerging security standards.

We followed the QuIPS framework to explore Research Questions 2.1, 2.2, and 2.3 by presenting methodologies for integrating security and privacy in QSE. In the automotive sector, we present a comparison of QBoost and SVC in terms of classification and time performance using an advanced automotive cybersecurity dataset. QBoost, a quantum boosting algorithm, significantly enhances training and testing times, even with complex datasets like Fuzzy, compared to SVC. The research addresses the real threat of fuzzing attacks on in-vehicle CAN bus systems and proposes a Quantum Intrusion Detection System (IDS) for rapid attack identification. QBoost employs One-vs-Rest techniques for multi-classification, and the IDS can query the automotive attack taxonomy, AAD, to retrieve attack descriptions using a pre-trained Sentence Transformers model. This model extracts key details from the National Vulnerability Database (NVD), aiding automotive developers in addressing vulnerabilities. Additionally, a vision model is introduced for classifying cyberattack types in the automotive sector, offering strategies to support developers throughout the software development lifecycle.

We introduce a Hybrid Quantum-Classical Architecture for Smart Cities that combines D-Wave Leap Quantum Cloud (DLQC) with IBM QRadar SIEM for enhanced security surveillance. The framework features an Integrated Security Dashboard for real-time insights on security incidents from both systems. Performance metrics show that the Quantum Machine Learning algorithm QBoost outperforms the Random Forest algorithm, detecting threats over 70% faster. The Intrusion Detection System (IDS) implemented here also demonstrated versatility across different urban scenarios. A comparative analysis of QML algorithms, including QBoost, Variational Quantum Classifier (VQC), and PegaSoS Quantum Support Vector Classification (QSVC), indicates QBoost as the most effective classifier within the proposed framework.

To bolster cybersecurity education, we presented ICSEQAI, a vision model where industry rules, standards, and information about existing vulnerable software components can be formulated as optimization problems and trained on

QAI models to support the development and testing phase in the organization.

Further, we also present two extremely important works on the proposal of security and privacy oriented quantum software engineering. We presented Quantum Privacy-Oriented Software Development (QPOSD), a method that integrates privacy and security into the Quantum Development Lifecycle (QDLC). QPOSD comprises five key components: QPbD Principles, Quantum Privacy Design Strategies, Quantum Privacy Patterns, Quantum Bugs/Vulnerabilities, and Context, which together form the Quantum Privacy Knowledge Base (QPKB) to guide development. Extending from that we also presented a methodology for analyzing code commits from major open-source quantum projects on GitHub is included, showing links between bug patterns, vulnerabilities, QDLC phases, and privacy/security principles. The results suggest that the development phase greatly affects vulnerabilities, particularly concerning End-to-End Security, Minimization of Attack Surface Area, and Establishing Secure Defaults, highlighting the need for early proactive measures in the QDLC.

Finally, we presented a unified framework combining DevSecOps, MLOps, and SCRUM for Quantum Software Engineering, which exploits the principle ideas of the aforementioned frameworks and proposes a framework that continuously integrates security and privacy requirements throughout the lifecycle of the application. The framework emphasizes the importance of collaboration among cross-functional teams, ensuring that security considerations are continuously integrated at every step of the developmental process.

5.2 Limitation

Despite the significant advances introduced by the QuIPS framework, there are inherent limitations in integrating Privacy by Design (PbD) and Security by Design (SbD) principles into Quantum Software Engineering (QSE).

Firstly, the constraints of current quantum hardware present considerable challenges. Quantum systems are still in their nascent stages, with instability, error-prone qubits, and limited quantum error correction capabilities making reliable implementation of PbD and SbD difficult. Issues such as qubit decoherence and side-channel vulnerabilities add additional layers of complexity, requiring further technological advancements to achieve stable security solutions.

Secondly, the integration of quantum and classical systems results in hybrid architectures that are susceptible to security vulnerabilities on both sides. Classical bugs can easily propagate into quantum operations, compromising the entire system. Maintaining consistent security measures across these disparate paradigms is an ongoing challenge that requires specialized tools and approaches.

Additionally, adapting classical security and privacy models to quantum contexts is not straightforward. The differences in data representation, quantum state

evolution, and the probabilistic nature of quantum algorithms complicate the direct application of classical frameworks, often necessitating extensive modifications or entirely new models.

Furthermore, the evolving quantum threat landscape means that security and privacy concerns must constantly adapt. Quantum systems are characterized by rapidly changing technology, which leads to continuous shifts in the associated vulnerabilities. As quantum hardware and algorithms evolve, security frameworks must be agile enough to evolve in tandem, which adds to their maintenance complexity.

Lastly, there is a lack of empirical validation and standardization in current QSE practices for integrating PbD and SbD. Without comprehensive validation across different quantum hardware platforms and diverse application domains, establishing consistent and effective security and privacy measures remains challenging.

These limitations highlight the need for continued research, adaptation of existing models, and the development of new frameworks to fully realize secure and privacy-preserving quantum software systems.

Moreover, some of the datasets utilized in this research contain synthetic data, which might have adverse impacts on the attained quality and time efficiency. Therefore, further research on real datasets is suggested for validation. Some experiments on quantum circuits and quantum machine learning algorithms were performed on simulated location machines, which might have given ideal solutions. However, real solutions on quantum backends are noisy and error-prone.

5.3 Future Work

The future research directions stem from the outcomes of the research carried out henceforth and from the limitations. Firstly, in the future a secure data communication between various data sources and the proposed Quantum as a Service (QaaS) architecture can be implemented using quantum key distribution (QKD). This advanced cryptographic technique leverages the principles of quantum mechanics to create secure communication channels that are theoretically immune to eavesdropping. Diving deeper into quantum security, it becomes increasingly pertinent to explore the potential for enhancing Intrusion Detection Systems (IDS) by incorporating a wider array of attack scenarios, such as zero-day attacks. By integrating diverse range quantum solutions into these systems, we can significantly bolster their resilience against sophisticated cyber threats, thereby creating a more robust security framework.

Moreover, by utilizing datasets from various domains such as mobility, environmental monitoring, economic trends, and governance, we can investigate whether quantum models can be generalized across these diverse fields. This interdisciplinary approach could unveil insights into the applicability of quan-

tum algorithms, particularly in classification and optimization tasks, across a spectrum of sectors and real-world situations.

Additionally, the development of Privacy Patterns for quantum software represents a critical area of focus. These patterns could encompass a range of strategies aimed at safeguarding user privacy within quantum systems. For instance, privacy-preserving optimizations of quantum circuits could ensure that sensitive data remains confidential during processing. Secure handling of quantum gate operations would mitigate risks associated with unauthorized access, while privacy-conscious measurement techniques could prevent the leakage of information during the quantum measurement process. By embedding these privacy and security measures into the fabric of quantum software, we can create a more secure and trustworthy environment for the deployment of quantum technologies.

Bibliography

- [1] M. Paltenghi and M. Pradel, “Bugs in quantum computing platforms: an empirical study,” *Proc. ACM Program. Lang.*, vol. 6, 04 2022.
- [2] S. Ghosh, S. Upadhyay, and A. A. Saki, “A primer on security of quantum computing,” 2023.
- [3] I. Vaccari, G. Chiola, M. Aiello, M. Mongelli, and E. Cambiaso, “Mqttset, a new dataset for machine learning techniques on mqtt,” *Sensors*, vol. 20, no. 22, 2020.
- [4] E. Seo, H. M. Song, and H. K. Kim, “Gids: Gan based intrusion detection system for in-vehicle network,” in *2018 16th Annual Conference on Privacy, Security and Trust (PST)*, pp. 1–6, IEEE, 2018.
- [5] H. M. Song, J. Woo, and H. K. Kim, “In-vehicle network intrusion detection using deep convolutional neural network,” *Vehicular Communications*, vol. 21, p. 100198, 2020.
- [6] E. Dijkstra, *Structured Programming*, pp. 41–48. USA: Yourdon Press, 1979.
- [7] J. D. Warnier, *Logical Construction of Programs*. Van Nostrand Reinhold Company, 1976.
- [8] J.-D. D. Warnier, *Logical Construction of Systems*. USA: John Wiley & Sons, Inc., 1981.
- [9] M. A. Jackson, *Principles of Program Design*. USA: Academic Press, Inc., 1975.
- [10] M. A. Jackson, *System development (Prentice-Hall International series in computer science)*. USA: Prentice-Hall, Inc., 1983.
- [11] “Manifesto for Agile Software Development,” oct 2024. [Online; accessed 25. Nov. 2024].
- [12] A. Cockburn and J. Highsmith, “Agile software development, the people factor,” *Computer*, vol. 34, no. 11, pp. 131–133, 2001.

- [13] L. Williams and A. Cockburn, “Agile software development: it’s about feedback and change,” *Computer*, vol. 36, no. 6, pp. 39–43, 2003.
- [14] G. Kim, P. Debois, J. Willis, and J. Humble, *The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations*. IT Revolution Press, 2016.
- [15] K. Beck, *Extreme programming explained: embrace change*. USA: Addison-Wesley Longman Publishing Co., Inc., 1999.
- [16] P. Duvall, S. M. Matyas, and A. Glover, *Continuous Integration: Improving Software Quality and Reducing Risk (The Addison-Wesley Signature Series)*. Addison-Wesley Professional, 2007.
- [17] J. Humble and D. Farley, *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*. Addison-Wesley Professional, 1st ed., 2010.
- [18] B. Vasilescu, Y. Yu, H. Wang, P. Devanbu, and V. Filkov, “Quality and productivity outcomes relating to continuous integration in github,” in *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2015*, (New York, NY, USA), p. 805–816, Association for Computing Machinery, 2015.
- [19] W. W. Royce, “Managing the development of large software systems,” in *Proceedings of IEEE WESCON*, pp. 1–9, 1970.
- [20] H. D. Mills, “Software development,” *IEEE Transactions on Software Engineering*, vol. SE-2, no. 4, pp. 265–273, 1980.
- [21] B. W. Boehm, “A spiral model of software development and enhancement,” *ACM SIGSOFT Software Engineering Notes*, vol. 11, no. 4, pp. 14–24, 1988.
- [22] K. Forsberg and H. Mooz, “The relationship of system engineering to the project cycle,” *INCOSE International Symposium*, vol. 1, no. 1, pp. 57–65, 1991.
- [23] J. Martin, *Rapid Application Development*. Macmillan Publishing, 1991.
- [24] C. Larman, *Agile and Iterative Development: A Manager’s Guide*. Addison-Wesley, 2003.
- [25] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland, and D. Thomas, “Manifesto for agile software development,” 2001.
- [26] K. Schwaber and J. Sutherland, “The scrum guide,” 2020.

- [27] G. Kim, P. Debois, J. Willis, and J. Humble, *The DevOps Handbook*. IT Revolution Press, 2016.
- [28] S. Shah, “Security as code: Devsecops,” *InfoQ*, 2018.
- [29] J. Klaise, A. Van Looveren, G. Vacanti, and A. Aguirre, “Monitoring and explainability of machine learning models in production,” *arXiv preprint arXiv:2103.13552*, 2021.
- [30] P. Benioff, “The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines,” *J. Stat. Phys.*, vol. 22, pp. 563–591, May 1980.
- [31] Y. I. Manin, “Computable and Non-Computable,” 1980.
- [32] R. P. Feynman, “Simulating physics with computers,” *Int. J. Theor. Phys.*, vol. 21, pp. 467–488, June 1982.
- [33] D. David, “Quantum theory, the Church–Turing principle and the universal quantum computer,” *Proc. R. Soc. Lond. A.*, vol. 400, pp. 97–117, July 1985.
- [34] P. W. Shor, “Algorithms for quantum computation: discrete logarithms and factoring,” in *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pp. 124–134, Santa Fe, NM, USA: IEEE, 11 1994.
- [35] L. K. Grover, “A fast quantum mechanical algorithm for database search,” in *STOC ’96: Proceedings of the twenty-eighth annual ACM symposium on Theory of Computing*, pp. 212–219, New York, NY, USA: Association for Computing Machinery, 07 1996.
- [36] E. Farhi, J. Goldstone, and S. Gutmann, “A quantum approximate optimization algorithm,” 2014.
- [37] M. Kjaergaard, M. E. Schwartz, J. Braumüller, P. Krantz, J. I.-J. Wang, S. Gustavsson, and W. D. Oliver, “Superconducting Qubits: Current State of Play,” *Annu. Rev. Condens. Matter Phys.*, vol. 11, pp. 369–395, March 2020.
- [38] C. D. Bruzewicz, J. Chiaverini, R. McConnell, and J. M. Sage, “Trapped-Ion Quantum Computing: Progress and Challenges,” *arXiv*, April 2019.
- [39] J. J. Pla, K. Y. Tan, J. P. Dehollain, W. H. Lim, J. J. L. Morton, D. N. Jamieson, A. S. Dzurak, and A. Morello, “A single-atom electron spin qubit in silicon,” *Nature*, vol. 489, pp. 541–545, September 2012.
- [40] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser, “Quantum computation by adiabatic evolution,” 2000.
- [41] T. Kadowaki and H. Nishimori, “Quantum annealing in the transverse ising model,” *Phys. Rev. E*, vol. 58, pp. 5355–5363, 11 1998.

- [42] J. Preskill, “Quantum Computing in the NISQ era and beyond,” *Quantum*, vol. 2, p. 79, August 2018.
- [43] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge, England, UK: Cambridge University Press, December 2010.
- [44] P. Kaye, R. Laflamme, M. Mosca, P. Kaye, R. Laflamme, and M. Mosca, *An Introduction to Quantum Computing*. Oxford, England, UK: Oxford University Press, November 2006.
- [45] S. Boixo, V. N. Smelyanskiy, A. Shabani, S. V. Isakov, M. Dykman, V. S. Denchev, M. H. Amin, A. Y. Smirnov, M. Mohseni, and H. Neven, “Computational multiqubit tunnelling in programmable quantum annealers,” *Nat. Commun.*, vol. 7, pp. 1–7, January 2016.
- [46] M. W. Johnson, M. H. S. Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson, R. Harris, A. J. Berkley, J. Johansson, P. Bunyk, E. M. Chapple, C. Enderud, J. P. Hilton, K. Karimi, E. Ladizinsky, N. Ladizinsky, T. Oh, I. Perminov, C. Rich, M. C. Thom, E. Tolkacheva, C. J. S. Truncik, S. Uchaikin, J. Wang, B. Wilson, and G. Rose, “Quantum annealing with manufactured spins,” *Nature*, vol. 473, pp. 194–198, May 2011.
- [47] H. Neven, V. S. Denchev, G. Rose, and W. G. Macready, “Qboost: Large scale classifier training with adiabatic quantum optimization,” in *Proceedings of the Asian Conference on Machine Learning* (S. C. H. Hoi and W. Buntine, eds.), vol. 25, pp. 333–348, PMLR, 04–06 Nov 2012.
- [48] M. Ying, “Foundations of quantum programming (extended abstract),” in *Programming Languages and Systems*, (Berlin, Heidelberg), pp. 16–20, Springer Berlin Heidelberg, 2010.
- [49] M. Ying, *Foundations of Quantum Programming*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1st ed., 2016.
- [50] S. Nagarajan, M. Malarvel, and T. J., “A framework for quantum based software development process,” in *2024 International Conference on Advances in Data Engineering and Intelligent Computing Systems (ADICS)*, pp. 1–6, 2024.
- [51] B. Weder, J. Barzen, F. Leymann, and D. Vietz, *Quantum Software Development Lifecycle*, pp. 61–83. Cham: Springer International Publishing, 2022.
- [52] N. Dey, M. Ghosh, S. S. Kundu, and A. Chakrabarti, “QDLC – the quantum development life cycle,” *arXiv*, no. arXiv:2010.08053, 2020.
- [53] J. Zhao, “Quantum software engineering: Landscapes and horizons,” 2021.

- [54] S. Ulyanov, A. Reshetnikov, O. Tyatyushkina, and V. Korenkov, “Quantum software engineering. Pt. I: Quantum Circuit (Gate) Model based Computing – education Lectures and pedagogical workshop,” *System Analysis in Science and Education*, pp. 129–201, Sept. 2020.
- [55] L. Zhou, G. Barthe, P.-Y. Strub, J. Liu, and M. Ying, “Coq: Foundational verification of quantum programs,” *Proc. ACM Program. Lang.*, vol. 7, Jan. 2023.
- [56] P. S. Kosobutskyy, “Physical foundations of quantum informatics: from quantum mechanics through quantum computing to quantum cryptography,” *Computer Design Systems. Theory and Practice*, vol. 4, pp. 33–47, Dec. 2022.
- [57] G. Bisicchia, J. Garcia-Alonso, J. M. Murillo, and A. Brogi, “From quantum mechanics to quantum software engineering: A historical review,” 2024.
- [58] M. A. Akbar, A. A. Khan, S. Mahmood, and S. Rafi, “Quantum software engineering: A new genre of computing,” in *1st ACM International Workshop on Quantum Software Engineering: The Next Evolution*, QSE-NE 2024, (New York, NY, USA), p. 1–6, Association for Computing Machinery, 2024.
- [59] M. Piattini, G. Peterssen, R. Pérez-Castillo, J. L. Hevia, M. A. Serrano, G. Hernández, I. G. R. de Guzmán, C. A. Paradela, M. Polo, E. Murina, L. Jiménez, J. C. Marqueño, R. Gallego, J. Tura, F. Phillipson, J. M. Murillo, A. Niño, and M. Rodríguez, “The talavera manifesto for quantum software engineering and programming,” in *Short Papers Proceedings of the 1st International Workshop on the QuANtum SoftWare Engineering & pRogramming, February 11-12, 2020* (M. Piattini, G. Peterssen, R. Pérez-Castillo, J. L. Hevia, and M. A. Serrano, eds.), vol. 2561 of *CEUR Workshop Proceedings*, (Talavera de la Reina, Spain), pp. 1–5, CEUR-WS.org, 2020.
- [60] C. H. Bennett and G. Brassard, “Quantum cryptography and its application to provably secure key expansion, public-key distribution, and coin-tossing,” in *Proceedings of IEEE International Symposium on Information Theory*, (St-Jovite, Canada), p. 91, 1983.
- [61] Y. Cao, Y. Zhao, Q. Wang, J. Zhang, S. X. Ng, and L. Hanzo, “The evolution of quantum key distribution networks: On the road to the qinternet,” *IEEE Communications Surveys & Tutorials*, vol. 24, no. 2, pp. 839–894, 2022.
- [62] Y. Tanizawa, R. Takahashi, H. Sato, A. R. Dixon, and S. Kawamura, “A secure communication network infrastructure based on quantum key distribution technology,” *IEICE Transactions on Communications*, vol. E99.B, no. 5, pp. 1054–1069, 2016.

- [63] T. Janani and M. Brindha, “A secure medical image transmission scheme aided by quantum representation,” *Journal of Information Security and Applications*, vol. 59, p. 102832, 2021.
- [64] A. Cavoukian *et al.*, “Privacy by design: The 7 foundational principles,” *Information and privacy commissioner of Ontario, Canada*, vol. 5, p. 12, 2009.
- [65] V. S. Barletta, D. Caivano, and A. Pal, “Integrating security and privacy in quantum software engineering,” in *Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering*, 2024.
- [66] N. Mousa and F. Shirazi, “A survey analysis of quantum computing adoption and the paradigm of privacy engineering,” *Security and Privacy*, vol. 7, p. e419, November 2024.
- [67] D. Arias, I. G. R. de Guzmán, M. Rodríguez, E. B. Terres, B. Sanz, J. G. de la Puerta, I. Pastor, A. Zubillaga, and P. G. Bringas, “Let’s do it right the first time: Survey on security concerns in the way to quantum software engineering,” *Neurocomputing*, vol. 538, p. 126199, 2023.
- [68] F. Qi, K. N. Smith, T. LeCompte, N.-f. Tzeng, X. Yuan, F. T. Chong, and L. Peng, “Quantum vulnerability analysis to guide robust quantum computing system design,” *IEEE Transactions on Quantum Engineering*, 2023.
- [69] P. Zhao, J. Zhao, Z. Miao, and S. Lan, “Bugs4q: A benchmark of real bugs for quantum programs,” in *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, (Melbourne, Australia), pp. 1373–1376, IEEE, 2021.
- [70] P. Zhao, J. Zhao, and L. Ma, “Identifying bug patterns in quantum programs,” 2021-03-16.
- [71] P. Zhao, X. Wu, Z. Li, and J. Zhao, “QChecker: Detecting bugs in quantum programs via static analysis,” in *2023 IEEE/ACM 4th International Workshop on Quantum Software Engineering (Q-SE)*, pp. 50–57, IEEE, 2023.
- [72] M. Paltenghi and M. Pradel, “MorphQ: Metamorphic testing of quantum computing platforms,” 2022-06-02.
- [73] M. R. E. Aoun, H. Li, F. Khomh, and L. Tidjon, “Bug characteristics in quantum software ecosystem,” 2022.
- [74] J. Luo, P. Zhao, Z. Miao, S. Lan, and J. Zhao, “A comprehensive study of bug fixes in quantum programs,” 2022-01-21.
- [75] P. K. Nayak, K. V. Kher, M. B. Chandra, M. V. P. Rao, and L. Zhang, “Q-PAC: Automated detection of quantum bug-fix patterns,” 2023.

- [76] S. A. Metwalli and R. Van Meter, “A categorization of bugs in quantum programs,” in *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pp. 346–347, IEEE, 2023.
- [77] K. Sentamilselvan, P. Suresh, G. K. Kamalam, H. Muthukrishnan, K. Logeswaran, and P. Keerthika, “Security Threats and Privacy Challenges in the Quantum Blockchain: A Contemporary Survey,” in *Quantum Blockchain*, pp. 293–316, Chichester, England, UK: John Wiley & Sons, Ltd., July 2022.
- [78] D. Leichtle, L. Music, E. Kashefi, and H. Ollivier, “Verifying bqp computations on noisy devices with minimal overhead,” *PRX Quantum*, vol. 2, October 2021.
- [79] C. Li, N. Kumar, Z. Song, S. Chakrabarti, and M. Pistoia, “Privacy-preserving quantum federated learning via gradient hiding,” *Quantum Sci. Technol.*, vol. 9, p. 035028, May 2024.
- [80] T. Patel, D. Silver, A. Ranjan, H. Gandhi, W. Cutler, and D. Tiwari, “Toward privacy in quantum program execution on untrusted quantum cloud computing machines for business-sensitive quantum needs,” 2023.
- [81] A. A. Saki, M. Alam, K. Phalak, A. Suresh, R. O. Topaloglu, and S. Ghosh, “A survey and tutorial on security and resilience of quantum computing,” in *2021 IEEE European Test Symposium (ETS)*, pp. 1–10, 2021.
- [82] P. W. Shor, “Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer,” *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1484–1509, 1997.
- [83] A. Pakhomchik and V. Vinokur, “Potential vulnerabilities of cryptographicalgorithms in the world of ideal quantum annealers,” in *The Role of Cybersecurity in the Industry 5.0 Era* (A. P. C. Kalloniatis, ed.), ch. 17, Rijeka: IntechOpen, 2024.
- [84] M. I. Garcia Cid, J. Álvaro González, L. Ortiz Martín, and D. Del Río Gómez, “Disruptive quantum safe technologies,” in *Proceedings of the 17th International Conference on Availability, Reliability and Security, ARES '22*, (New York, NY, USA), Association for Computing Machinery, 2022.
- [85] C. H. Bennett and G. Brassard, “Quantum cryptography: Public key distribution and coin tossing,” *Theoretical Computer Science*, vol. 560, p. 7–11, December 2014.
- [86] C. H. Bennett, “Quantum cryptography using any two non-orthogonal states,” *Physical Review Letters*, vol. 68, no. 21, pp. 3121–3124, 1992.

- [87] A. K. Ekert, “Quantum cryptography based on bell’s theorem,” *Physical Review Letters*, vol. 67, no. 6, pp. 661–663, 1991.
- [88] S. Pirandola, U. L. Andersen, L. Banchi, *et al.*, “Advances in quantum cryptography,” *Advances in Optics and Photonics*, vol. 12, no. 4, pp. 1012–1236, 2020.
- [89] N. Gisin, G. Ribordy, W. Tittel, and H. Zbinden, “Quantum cryptography,” *Reviews of Modern Physics*, vol. 74, no. 1, pp. 145–195, 2002.
- [90] V. S. Barletta, D. Caivano, M. De Vincentiis, A. Magri, and A. Piccinno, “Quantum optimization for iot security detection,” in *Ambient Intelligence—Software and Applications—13th International Symposium on Ambient Intelligence*, (Cham), pp. 187–196, Springer International Publishing, 2023.
- [91] P. R. K. Reddy and S. Bouzefrane, “Analysis and detection of dos attacks in cloud computing by using qse algorithm,” in *2014 IEEE Intl Conf on High Performance Computing and Communications, 2014 IEEE 6th Intl Symp on Cyberspace Safety and Security, 2014 IEEE 11th Intl Conf on Embedded Software and Syst (HPCC,CSS,ICSS)*, pp. 1089–1096, 2014.
- [92] M. De Vincentiis, F. Cassano, A. Pagano, and A. Piccinno, “Qai4ase: Quantum artificial intelligence for automotive software engineering,” in *Proceedings of the 1st International Workshop on Quantum Programming for Software Engineering*, pp. 19–21, 2022.
- [93] D. Caivano, M. De Vincentiis, F. Nitti, and A. Pal, “Quantum optimization for fast can bus intrusion detection,” in *Proceedings of the 1st International Workshop on Quantum Programming for Software Engineering*, QP4SE 2022, (New York, NY, USA), p. 15–18, Association for Computing Machinery, 2022.
- [94] M. S. Salek, P. K. Biswas, J. Pollard, J. Hales, Z. Shen, V. Dixit, M. Chowdhury, S. M. Khan, and Y. Wang, “A novel hybrid quantum-classical framework for an in-vehicle controller area network intrusion detection,” *IEEE Access*, vol. 11, pp. 96081–96092, 2023.
- [95] M. Wang, A. Huang, Y. Liu, X. Yi, J. Wu, and S. Wang, “A quantum-classical hybrid solution for deep anomaly detection,” *Entropy*, vol. 25, no. 3, 2023.
- [96] H. Suryotrisongko and Y. Musashi, “Evaluating hybrid quantum-classical deep learning for cybersecurity botnet dga detection,” *Procedia Computer Science*, vol. 197, pp. 223–229, 2022. Sixth Information Systems International Conference (ISICO 2021).
- [97] S. Y.-C. Chen, S. Yoo, and Y.-L. L. Fang, “Quantum long short-term memory,” in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8622–8626, 2022.

- [98] D. Herr, B. Obert, and M. Rosenkranz, “Anomaly detection with variational quantum generative adversarial networks,” *Quantum Science and Technology*, vol. 6, p. 045004, jul 2021.
- [99] V. S. Barletta, D. Caivano, A. Lako, and A. Pal, “Quantum as a service architecture for security in a smart city,” in *Quality of Information and Communications Technology*, (Cham), pp. 76–89, Springer Nature Switzerland, 2023.
- [100] D. Micciancio and O. Regev, *Lattice-based Cryptography*, pp. 147–191. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.
- [101] P. N. Vithalkar, “Cryptographic Protocols Resilient to Quantum Attacks: Advancements in Post-Quantum Cryptography,” *Comm. Appl. Nonlinear Anal.*, vol. 31, pp. 520–532, June 2024.
- [102] C. B. R and A. K. Koundinya, “2-d information encoding for lattice-based encryption scheme using entropy approach for futuristic applications,” in *2024 IEEE 9th International Conference for Convergence in Technology (I2CT)*, pp. 1–5, 2024.
- [103] V. S, P. S, C. Harshavardhana, and K. R, “Quantum-resistant cryptographic algorithms for secure communication,” *ICTACT Journal on Communication Technology*, 2024.
- [104] F. Nisha, J. Lenin, S. Saravanan, V. R. Rohit, P. D. Selvam, and M. Rajmohan, “Lattice-based cryptography and ntru: Quantum-resistant encryption algorithms,” in *2024 International Conference on Emerging Systems and Intelligent Computing (ESIC)*, pp. 509–514, 2024.
- [105] A. Tiwari, R. Chauhan, N. Joshi, S. Devliyal, S. Aluvala, and A. Kumar, “The quantum threat: Implications for data security and the rise of post-quantum cryptography,” in *2024 IEEE 9th International Conference for Convergence in Technology (I2CT)*, pp. 1–7, 2024.
- [106] D. J. Bernstein, A. Hülsing, S. Kölbl, R. Niederhagen, J. Rijneveld, and P. Schwabe, “The sphincs+ signature framework,” in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS ’19*, (New York, NY, USA), p. 2129–2146, Association for Computing Machinery, 2019.
- [107] P. Thanalakshmi, R. Anitha, N. Anbazhagan, C. Park, G. P. Joshi, and C. Seo, “A hash-based quantum-resistant designated verifier signature scheme,” *Mathematics*, vol. 10, no. 10, 2022.
- [108] Q. Yuan, M. Tibouchi, and M. Abe, “Quantum-access security of hash-based signature schemes,” in *Information Security and Privacy* (L. Simpson and M. A. Rezazadeh Bae, eds.), (Cham), pp. 343–380, Springer Nature Switzerland, 2023.

- [109] T. Hatanaka, R. Fushio, M. Watanabe, W. J. Munro, T. N. Ikeda, and S. Sugiura, “A quantum-resistant photonic hash function,” 2024.
- [110] S. Ghosh, M. Zaman, and S. Sampalli, “Quantum-Safe Asymmetric Cryptosystems : Current Solutions and Future Directions against Quantum Attacks,” in *Holistic Approach to Quantum Cryptography in Cyber Security*, pp. 99–120, Boca Raton, FL, USA: CRC Press, August 2022.
- [111] S. Dai, “Quantum cryptanalysis on a multivariate cryptosystem based on clipped hopfield neural network,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 9, pp. 5080–5084, 2022.
- [112] F. H. Makoui, T. A. Gulliver, and M. Dakhilalian, “Post quantum code-based cryptosystems with dual inverse matrix,” in *2023 13th International Conference on Information Technology in Asia (CITA)*, pp. 43–47, 2023.
- [113] A. Kichna and A. Farchane, “Secure and efficient code-based cryptography for multi-party computation and digital signatures,” *Computer Sciences & Mathematics Forum*, vol. 6, no. 1, 2023.
- [114] V. S. Barletta, C. Catalano, M. De Vincentiis, A. Pal, and M. Scalera, “Artificial intelligence for automotive security: How to support developers in automotive solutions,” in *2023 IEEE International Conference on Metrology for eXtended Reality, Artificial Intelligence, and Neural Engineering (MetroXRaine)*, pp. 1–5, IEEE, 2023.
- [115] D. Caivano, M. De Vincentiis, A. Pal, and M. Scalera, “Extending developer support: Quantum artificial intelligence for automotive security,” in *Proceedings of the 2nd International Workshop on Quantum Programming for Software Engineering*, 2023.
- [116] M. De Vincentiis, A. Pal, A. Ragone, and M. Scalera, “A Multi-class Intrusion Detection System for Cyber Security Education in Automotive Industry,” *CEUR Workshop Proceedings*, vol. 3408, June 2023.
- [117] M. T. Baldassarre, M. De Vincentiis, A. Pal, and M. Scalera, “Quantum artificial intelligence for cyber security education in software engineering,” in *IS-EUD 2023: 9th International Symposium on End-User Development, Cagliari, Italy*, vol. 3408, June 2023.
- [118] D. Caivano, M. De Vincentiis, A. Pal, and A. Ragone, “Securing smart cities: Unraveling quantum as a service,” in *Proceedings of the 2nd International Workshop on Quantum Programming for Software Engineering*, pp. 1–5, 2023.
- [119] V. S. Barletta, D. Caivano, M. De Vincentiis, A. Pal, and M. Scalera, “Hybrid quantum architecture for smart city security,” *Journal of Systems and Software*, vol. 217, p. 112161, 2024.

- [120] Z. A. Maher, A. Ghulam, M. Y. Koondhar, I. A. Mahar, and M. K. M. Z. Hyder, "Security based scrum process (subscrum) for secure agile development," *VFAST transactions on software engineering*, vol. 12, no. 3, pp. 138–146, 2024.
- [121] G. Koç and M. Aydos, "Trustworthy scrum: Development of secure software with scrum," in *2017 International Conference on Computer Science and Engineering (UBMK)*, pp. 244–249, 2017.
- [122] L. Marchesi, M. Marchesi, and R. Tonelli, "A survey on cryptoagility and agile practices in the light of quantum resistance," *Information and Software Technology*, vol. 178, p. 107604, 2025.
- [123] A. A. Khan, M. A. Akbar, A. Ahmad, M. Fahmideh, M. Shameem, V. Lahtinen, M. Waseem, and T. Mikkonen, "Agile practices for quantum software development: Practitioners' perspectives," in *2023 IEEE International Conference on Quantum Software (QSW)*, pp. 9–20, 2023.
- [124] G. J. Hernández González and C. A. Paradela, "Quantum agile development framework," in *Quality of Information and Communications Technology* (M. Shepperd, F. Brito e Abreu, A. Rodrigues da Silva, and R. Pérez-Castillo, eds.), (Cham), pp. 284–291, Springer International Publishing, 2020.
- [125] X. Zhao, T. Clear, and R. Lal, "Identifying the primary dimensions of devsecops: A multi-vocal literature review," *Journal of Systems and Software*, vol. 214, p. 112063, 2024.
- [126] N. Pakalapati, S. Venkatasubbu, and S. M. K. Sistla, "The Convergence of AI/ML and DevSecOps: Revolutionizing Software Development," *J. Knowl. Learn. Sci. Technol.*, vol. 2, pp. 189–212, August 2023.
- [127] N. G. Camacho, "Unlocking the Potential of AI/ML in DevSecOps: Effective Strategies and Optimal Practices," *JAIGS*, vol. 3, pp. 106–115, March 2024.
- [128] Kung, Johnny and Fancy, Murium, "A quantum revolution - report on global policies for quantum technology," 2021.
- [129] M. Piattini, M. Serrano, R. Perez-Castillo, G. Petersen, and J. L. Hevia, "Toward a quantum software engineering," *IT Professional*, vol. 23, no. 1, pp. 62–66, 2021.
- [130] S. S. Gill, A. Kumar, H. Singh, M. Singh, K. Kaur, M. Usman, and R. Buyya, "Quantum computing: A taxonomy, systematic review and future directions," *Software: Practice and Experience*, vol. 52, no. 1, pp. 66–114, 2021.
- [131] C. Chareton, S. Bardin, D. Lee, B. Valiron, R. Vilmart, and Z. Xu, "Formal methods for quantum programs: A survey," 2022.

- [132] A. A. Khan, A. Ahmad, M. Waseem, P. Liang, M. Fahmideh, T. Mikkonen, and P. Abrahamsson, “Software architecture for quantum computing systems – a systematic review,” 2022.
- [133] B. Sodhi and R. Kapur, “Quantum computing platforms: Assessing the impact on quality attributes and sdlc activities,” in *2021 IEEE 18th International Conference on Software Architecture (ICSA)*, pp. 80–91, 2021.
- [134] H. Nejatollahi, N. Dutt, S. Ray, F. Regazzoni, I. Banerjee, and R. Cammarota, “Post-quantum lattice-based cryptography implementations: A survey,” *ACM Computing Surveys*, vol. 51, no. 6, pp. 1–41, 2019.
- [135] D. S. Gupta, S. Ray, T. Singh, and M. Kumari, “Post-quantum lightweight identity-based two-party authenticated key exchange protocol for internet of vehicles with probable security,” *Computer Communications*, vol. 181, pp. 69–79, 2022.
- [136] L. Malina, P. Dzurenda, S. Ricci, J. Hajny, G. Srivastava, R. Matulevicius, A.-A. O. Affia, M. Laurent, N. H. Sultan, and Q. Tang, “Post-quantum era privacy protection for intelligent infrastructures,” *IEEE Access*, vol. 9, pp. 36038–36077, 2021.
- [137] K. B. A. and S. Charters, “Guidelines for performing Systematic Literature Reviews in Software Engineering,” *ResearchGate*, vol. 2, January 2007.
- [138] P. Brereton, B. A. Kitchenham, D. Budgen, M. Turner, and M. Khalil, “Lessons from applying the systematic literature review process within the software engineering domain,” *Journal of Systems and Software*, vol. 80, no. 4, pp. 571–583, 2007. Software Performance.
- [139] T. Dyba, T. Dingsoyr, and G. K. Hanssen, “Applying systematic reviews to diverse study types: An experience report,” in *First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007)*, pp. 225–234, IEEE, 09 2007.
- [140] M. T. Baldassarre, D. Caivano, S. Romano, and G. Scanniello, “Software models for source code maintainability: A systematic literature review,” in *2019 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pp. 252–259, Kallithea, Greece: IEEE, 08 2019.
- [141] O. Dieste and A. G. Padua, “Developing search strategies for detecting relevant experiments for systematic reviews,” in *First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007)*, pp. 215–224, Madrid, Spain: IEEE, 09 2007.
- [142] M. Ouzzani, H. Hammady, Z. Fedorowicz, and A. Elmagarmid, “Rayyan—a web and mobile app for systematic reviews,” *Systematic Reviews*, vol. 5, no. 1, p. 210, 2016.

- [143] T. Dybå and T. Dingsøy, “Empirical studies of agile software development: A systematic review,” *Information and Software Technology*, vol. 50, no. 9, pp. 833–859, 2008.
- [144] T. Dybå and T. Dingsøy, “Strength of evidence in systematic reviews in software engineering,” in *Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, ESEM ’08, (New York, NY, USA), p. 178–187, Association for Computing Machinery, 2008.
- [145] L. Yang, H. Zhang, H. Shen, X. Huang, X. Zhou, G. Rong, and D. Shao, “Quality assessment in systematic literature reviews: A software engineering perspective,” *Information and Software Technology*, vol. 130, p. 106397, 2021.
- [146] A. M. D’angelo, R. Dell’eva, H. Inamori, and A. Martinoli, “Towards practical quantum key distribution: The *eqcspot* system,” *Journal of Modern Optics*, vol. 48, pp. 1943–1956, 11 2001.
- [147] A. A. A. EL-Latif, B. Abd-El-Atty, S. E. Venegas-Andraca, and W. Mazurczyk, “Efficient quantum-based security protocols for information sharing and data protection in 5g networks,” *Future Generation Computer Systems*, vol. 100, pp. 893–906, 2019.
- [148] X. Tan, X. Zhang, and J. Li, “Big data quantum private comparison with the intelligent third party,” *Journal of Ambient Intelligence and Humanized Computing*, vol. 6, no. 6, pp. 797–806, 2015.
- [149] R.-H. Shi, “Quantum sealed-bid auction without a trusted third party,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 10, pp. 4221–4231, 2021.
- [150] T.-S. Lin, I.-M. Tsai, H.-W. Wang, and S.-Y. Kuo, “Quantum authentication and secure communication protocols,” in *2006 6th IEEE Conference on Nanotechnology*, vol. 2, pp. 863–866, 7 2006.
- [151] J. Park, S. Cho, T. Lim, S. Bhunia, and M. Tehranipoor, “Scr-qrng: Side-channel resistant design using quantum random number generator,” in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–8, 2019.
- [152] I. Capraro and T. Occhipinti, “Implementation of a real time high level protocol software for quantum key distribution,” in *2007 IEEE International Conference on Signal Processing and Communications*, pp. 704–707, IEEE, 2007.
- [153] R. Kaewpuang, M. Xu, D. Niyato, H. Yu, Z. Xiong, and X. S. Shen, “Adaptive resource allocation in quantum key distribution (qkd) for federated learning,” in *International Conference on Computing, Networking and Communications (ICNC)*, pp. 71–76, 2023.

- [154] P. Wang, X. Zhang, G. Chen, K. Pham, and E. Blasch, “Quantum key distribution for security guarantees over quantum-repeater-based qos-driven 3d satellite networks,” in *2014 IEEE Global Communications Conference*, pp. 728–733, 2014.
- [155] M. Q. Vu, T. V. Pham, N. T. Dang, and A. T. Pham, “Design and performance of relay-assisted satellite free-space optical quantum key distribution systems,” *IEEE Access*, vol. 8, pp. 122498–122510, 2020.
- [156] Q. Zhang, O. Ayoub, J. Wu, X. Lin, and M. Tornatore, “Ic-qkd: An information-centric quantum key distribution network,” *IEEE Communications Magazine*, pp. 1–7, 2023.
- [157] Y.-G. Yang, B.-X. Liu, G.-B. Xu, Y.-H. Zhou, and W.-M. Shi, “Practical quantum anonymous private information retrieval based on quantum key distribution,” *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 4034–4045, 2023.
- [158] K. E. Rumyantsev and A. P. Pljonkin, “Preliminary stage synchronization algorithm of auto-compensation quantum key distribution system with an unauthorized access security,” in *2016 International Conference on Electronics, Information, and Communications (ICEIC)*, pp. 1–4, 2016.
- [159] L. Zhou and M. Ying, “Differential privacy in quantum computation,” in *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pp. 249–262, 2017.
- [160] G. Li, D. Wang, C. Cao, H. Qin, L. Zhang, and G. Zhao, “Research on the fusion architecture and application mode of quantum cryptography and classic cryptography,” in *Proceedings of the 2017 International Conference on E-Society, E-Education and E-Technology*, ICSET 2017, (New York, NY, USA), p. 28–32, Association for Computing Machinery, 2017.
- [161] T. Ngoben, B. Kabaso, and A. Mukherjee, “A generic framework for the implementation of a secure quantum teleportation infrastructure,” in *International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD)*, (Durban, South Africa), pp. 1–6, IEEE, 2020.
- [162] H. Yong and F. Z. Xue, “Quantum growing hierarchical self organized map-based intrusion detection system,” in *International Conference on System Science, Engineering Design and Manufacturing Informatization*, vol. 2, pp. 110–115, 2010.
- [163] T. Choi, H. Kim, J. Kim, C. S. Yoon, and G. M. Lee, “Quantum key distribution networks for trusted 5g and beyond: An itu-t standardization perspective,” in *ITU Kaleidoscope: Connecting Physical and Virtual Worlds (ITU K)*, pp. 1–9, 12 2021.

- [164] A. Acín, “Device-independent quantum key distribution,” in *2015 Conference on Lasers and Electro-Optics (CLEO)*, pp. 1–2, 2015.
- [165] J. Russell, “Application of quantum key distribution,” in *MILCOM 2008 - IEEE Military Communications Conference*, pp. 1–6, 11 2008.
- [166] S. Rass and P. Schartner, “Game-theoretic security analysis of quantum networks,” in *2009 Third International Conference on Quantum, Nano and Micro Technologies*, pp. 20–25, 2009.
- [167] R. T. Possignolo and C. B. Margi, “A quantum-classical hybrid architecture for security algorithms acceleration,” in *2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*, pp. 1032–1037, 2012.
- [168] A. Hassan, W. Batrafi, and K. Elleithy, “Approaching secure protocol from quantum perspective,” in *IEEE Long Island Systems, Applications and Technology Conference (LISAT)*, pp. 1–6, 2016.
- [169] A. Sharma and D. Sharma, “Big data protection via neural and quantum cryptography,” in *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, pp. 3701–3704, 3 2016.
- [170] H. Chen, J. Shi, F. Zhou, J. Cheng, and R. Shi, “Quantum blind dual-signature scheme based on cluster states,” in *2017 IEEE 17th International Conference on Nanotechnology (IEEE-NANO)*, pp. 30–35, 7 2017.
- [171] P. K. Tysowski, X. Ling, N. Lütkenhaus, and M. Mosca, “The engineering of a scalable multi-site communications system utilizing quantum key distribution (QKD),” *Quantum Science and Technology*, 2017.
- [172] A. Aguado, ro, V. Lopez, J. Martinez-Mateo, M. Peev, D. Lopez, and V. Martin, “Virtual network function deployment and service automation to provide end-to-end quantum encryption,” *Journal of Optical Communications and Networking*, vol. 10, no. 4, pp. 421–430, 2018.
- [173] Z. Cai, J. Qu, P. Liu, and J. Yu, “A blockchain smart contract based on light-weighted quantum blind signature,” *IEEE Access*, vol. 7, pp. 138657–138668, 2019.
- [174] K. Dong, Y. Zhao, X. Yu, J. Zhang, H. Yu, and Y. Zhang, “Auxiliary topology based global quantum key distribution for secure multicast service,” in *2019 24th OptoElectronics and Communications Conference (OECC) and International Conference on Photonics in Switching and Computing (PSC)*, pp. 1–3, 7 2019.
- [175] R. Diovu and J. Agee, “Enhancing the security of a cloud-based smart grid AMI network by leveraging on the features of quantum key distribution,” *Transactions on Emerging Telecommunications Technologies*, vol. 30, no. 6, 2019.

- [176] A. E. Qoussini, Y. I. Daradkeh, S. M. Al Tabib, S. Gnatyuk, T. Okhrimenko, and V. Kinzeryavyy, “Improved model of quantum deterministic protocol implementation in channel with noise,” in *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, vol. 1, pp. 572–578, 9 2019.
- [177] Y. Li, P. Zhang, and R. Huang, “Lightweight quantum encryption for secure transmission of power data in smart grid,” *IEEE Access*, vol. 7, pp. 36285–36293, 2019.
- [178] H. Wang, Y. Zhao, X. Yu, Z. Ma, J. Wang, A. Nag, L. Yi, and J. Zhang, “Protection schemes for key service in optical networks secured by quantum key distribution (qkd),” *Journal of Optical Communications and Networking*, vol. 11, no. 3, pp. 67–78, 2019.
- [179] Z. Ji, H. Zhang, and H. Wang, “Quantum private comparison protocols with a number of multi-particle entangled states,” *IEEE Access*, vol. 7, pp. 44613–44621, 2019.
- [180] S. Xia, H. Ye, X. Xia, B. Xue, K. Yang, and Y. Lu, “Research and design of key management system in power distribution network based on quantum private communication,” in *2nd International Conference on Information Systems and Computer Aided Education (ICISCAE)*, pp. 558–562, 2019.
- [181] S. Zhou, “A quantum image encryption method based on dnacnot,” *IEEE Access*, vol. 8, pp. 178336–178344, 2020.
- [182] A. Aguado, ro, V. López, J. P. Brito, A. Pastor, D. R. López, and V. Martin, “Enabling quantum key distribution networks via software-defined networking,” in *2020 International Conference on Optical Network Design and Modeling (ONDM)*, pp. 1–5, 5 2020.
- [183] T. Ngoben, B. Kabaso, and A. Mukherjee, “Towards a secure quantum communication implementation for osi layer 2 devices,” in *2020 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD)*, pp. 1–6, 8 2020.
- [184] A. A. Saki, M. Alam, and S. Ghosh, “Impact of noise on the resilience and the security of quantum computing,” in *2021 22nd International Symposium on Quality Electronic Design (ISQED)*, pp. 186–191, 2021.
- [185] W. Gao and L. Yang, “Quantum election protocol based on quantum public key cryptosystem,” *Security and Communication Networks*, vol. 2021, pp. 1–15, 2021.
- [186] A. H. Ismail, H. M. Waseem, M. Ishtiaq, S. S. Jamal, and M. Khan, “Quantum spin half algebra and generalized megrelshvili protocol for confidentiality of digital images,” *International Journal of Theoretical Physics*, vol. 60, no. 5, pp. 1720–1741, 2021.

- [187] A. A. Saki, A. Suresh, R. O. Topaloglu, and S. Ghosh, "Split compilation for security of quantum circuits," in *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pp. 1–7, 2021.
- [188] R.-H. Shi and Y.-F. Li, "A feasible quantum sealed-bid auction scheme without an auctioneer," *IEEE Transactions on Quantum Engineering*, vol. 3, pp. 1–12, 2022.
- [189] B. Abd-El-Atty, "A robust medical image steganography approach based on particle swarm optimization algorithm and quantum walks," *Neural Comput. Appl.*, vol. 35, p. 773–785, 9 2022.
- [190] R.-H. Shi, "Anonymous quantum sealed-bid auction," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, pp. 414–418, 2 2022.
- [191] Z. Yang, T. Salman, R. Jain, and R. D. Pietro, "Decentralization using quantum blockchain: A theoretical analysis," *IEEE Transactions on Quantum Engineering*, vol. 3, pp. 1–16, 2022.
- [192] V. K. Ralegankar, J. Bagul, B. Thakkar, R. Gupta, S. Tanwar, G. Sharma, and I. E. Davidson, "Quantum cryptography-as-a-service for secure uav communication: Applications, challenges, and case study," *IEEE Access*, vol. 10, pp. 1475–1492, 2022.
- [193] A. A. Saki, R. O. Topaloglu, and S. Ghosh, "Shuttle-exploiting attacks and their defenses in trapped-ion quantum computers," *IEEE Access*, vol. 10, pp. 2686–2699, 2022.
- [194] K. Witek, M. Caccia, W. Kucewicz, M. Baszczyk, P. Dorosz, and L. Mik, "A method for implementing a sha256 hardware accelerator inside an quantum true random number generator (qtrng)," in *30th International Conference on Mixed Design of Integrated Circuits and System (MIXDES)*, pp. 251–256, 2023.
- [195] M. Al-Hawawreh and M. S. Hossain, "A privacy-aware framework for detecting cyber attacks on internet of medical things systems using data fusion and quantum deep learning," *Information Fusion*, vol. 99, p. 101889, 2023.
- [196] L. D., N. Nagpal, Ch, S. rasekaran, and J. H. D., "A quantum-based approach for offensive security against cyber attacks in electrical infrastructure," *Applied Soft Computing*, vol. 136, p. 110071, 2023.
- [197] M. Wang, J. Li, K. Xue, R. Li, N. Yu, Y. Li, Y. Liu, Q. Sun, and J. Lu, "A segment-based multipath distribution method in partially-trusted relay quantum networks," *IEEE Communications Magazine*, pp. 1–7, 2023.
- [198] S. Paramasivam, J. Jenitha, S. Sanjana, and M. Haghparast, "Compact quantum circuit design of puffin and print lightweight ciphers for quantum key recovery attack," *IEEE Access*, vol. 11, pp. 66767–66776, 2023.

- [199] S. Gupta, A. Gupta, I. Y. Pandya, A. Bhatt, and K. Mehta, “End to end secure e-voting using blockchain & quantum key distribution,” *Materials Today: Proceedings*, p. S2214785321051130, 2021-08.
- [200] R.-h. Shi and H. Yu, “Privacy-preserving range query quantum scheme with single photons in edge-based internet of things,” *IEEE Transactions on Network and Service Management*, pp. 1–1, 2023.
- [201] Q. Zhu, Y. Zhao, X. Yu, and J. Zhang, “Quantum cloud with communication, computing, caching, and cipher (4c) resource coordination,” in *2023 International Conference on Optical Network Design and Modeling (ONDM)*, pp. 1–3, 2023.
- [202] Y.-G. Yang, Z.-C. Liu, J. Li, X.-B. Chen, H.-J. Zuo, Y.-H. Zhou, and W.-M. Shi, “Quantum private query with perfect user privacy against a joint-measurement attack,” *Physics Letters A*, vol. 380, no. 48, pp. 4033–4038, 2016.
- [203] R.-h. Shi and M. Zhang, “Privacy-preserving Quantum Sealed-bid Auction Based on Grover’s Search Algorithm,” *Sci. Rep.*, vol. 9, pp. 1–10, May 2019.
- [204] R. Runse, P. Toliver, S. McNown, T. Chapuran, M. Goodman, J. Jackel, R. Hughes, J. Nordholt, C. Peterson, K. Tyagi, P. Hiskett, and K. McCabe, “Quantum cryptography in optical networks and supporting metrology,” in *Technical Digest: Symposium on Optical Fiber Measurements, 2004.*, pp. 159–162, 2004.
- [205] M. Armanuzzaman, K. M. R. Alam, M. M. Hassan, and Y. Morimoto, “A secure and efficient data transmission technique using quantum key distribution,” in *2017 4th International Conference on Networking, Systems and Security (NSysS)*, pp. 1–5, 12 2017.
- [206] A. Aguado, ro, E. Hugues-Salas, P. A. Haigh, J. Marhuenda, A. B. Price, P. Sibson, J. E. Kennard, C. Erven, J. G. Rarity, M. G. Thompson, A. Lord, R. Nejabati, and D. Simeonidou, “Secure nfv orchestration over an sdn-controlled optical network with time-shared quantum key distribution resources,” *Journal of Lightwave Technology*, vol. 35, no. 8, pp. 1357–1362, 2017.
- [207] P. V. Trinh, T. V. Pham, N. T. Dang, H. V. Nguyen, S. X. Ng, and A. T. Pham, “Design and security analysis of quantum key distribution protocol over free-space optics using dual-threshold direct-detection receiver,” *IEEE Access*, vol. 6, pp. 4159–4175, 2018.
- [208] D. Li, P. Huang, Y. Zhou, Y. Li, and G. Zeng, “Memory-saving implementation of high-speed privacy amplification algorithm for continuous-variable quantum key distribution,” *IEEE Photonics Journal*, vol. 10, no. 5, pp. 1–12, 2018.

- [209] A. A. A. EL-Latif, B. Abd-El-Atty, and S. E. Venegas-Andraca, “Controlled alternate quantum walk-based pseudo-random number generator and its application to quantum color image encryption,” *Physica A: Statistical Mechanics and its Applications*, vol. 547, p. 123869, 2020.
- [210] S. Xia, Z. Wang, Z. Hou, H. Ye, B. Xue, S. Wang, X. Zhang, and K. Yang, “Design of quantum key fusion model for power multi-terminal,” in *2020 IEEE 3rd International Conference on Information Systems and Computer Aided Education (ICISCAE)*, pp. 196–199, 2020.
- [211] L. Xue, L. Cheng, Y. Li, and Y. Mao, “Quantum machine learning for electricity theft detection: an initial investigation,” in *2021 IEEE International Conferences on Internet of Things (iThings) and IEEE Green Computing & Communications (GreenCom) and IEEE Cyber, Physical & Social Computing (CPSCoM) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics)*, pp. 204–208, 2021.
- [212] W.-K. Lee, K. Jang, G. Song, H. Kim, S. O. Hwang, and H. Seo, “Efficient implementation of lightweight hash functions on gpu and quantum computers for iot applications,” *IEEE Access*, vol. 10, pp. 59661–59674, 2022.
- [213] B. Abd-El-Atty and A. A. Abd EL-Latif, “Applicable image cryptosystem using bit-level permutation, particle swarm optimisation, and quantum walks,” *Neural Comput. Appl.*, vol. 35, p. 18325–18341, 6 2023.
- [214] H. Wen, A. Xu, and H. Qi, “Application of quantum key distribution in intelligent security operation and maintenance of power communication networks,” *Results in Physics*, vol. 54, p. 107041, 2023.
- [215] S. Deshpande, C. Xu, T. Trochatos, H. Wang, F. Erata, S. Han, Y. Ding, and J. Szefer, “Design of quantum computer antivirus,” in *2023 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 260–270, 2023.
- [216] B.-Z. Yan, Q. Li, H.-K. Mao, H.-W. Xu, and A. A. A. El-Latif, “Large-scale and high-speed fpga-based privacy amplification for quantum key distribution,” *Journal of Lightwave Technology*, vol. 41, no. 1, pp. 169–175, 2023.
- [217] H. Kirkland and S. J. Koppal, “Schrödinger’s camera: First steps towards a quantum-based privacy preserving camera,” in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, (Los Alamitos, CA, USA), pp. 18–27, IEEE Computer Society, jun 2023.
- [218] R. V and P. G, “Securely transmit data over long distances using quantum key distribution based on e91 protocol,” in *2023 International Conference on Networking and Communications (ICNWC)*, pp. 1–7, 4 2023.

- [219] K. Phalak, A. A. Saki, M. Alam, R. O. Topaloglu, and S. Ghosh, “Quantum puf for security and trust in quantum computing,” *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 11, no. 2, pp. 333–342, 2021.
- [220] A. Suresh, A. A. Saki, M. Alam, R. Onur Topaloglu, and S. Ghosh, “Short paper: A quantum circuit obfuscation methodology for security and privacy,” in *Proceedings of the 10th International Workshop on Hardware and Architectural Support for Security and Privacy*, HASP ’21, (New York, NY, USA), Association for Computing Machinery, 2022.
- [221] A. A. Abd EL-Latif, B. Abd-El-Atty, E. M. Abou-Nassar, and S. E. Venegas-Andraca, “Controlled alternate quantum walks based privacy preserving healthcare images in internet of things,” *Optics & Laser Technology*, vol. 124, p. 105942, 2020.
- [222] C. H. Bennett, “Quantum cryptography using any two nonorthogonal states,” *Phys. Rev. Lett.*, vol. 68, pp. 3121–3124, May 1992.
- [223] J. Park, S. Cho, T. Lim, and M. Tehranipoor, “Qec: A quantum entropy chip and its applications,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 6, pp. 1471–1484, 2020.
- [224] M. Piattini, G. Peterssen, and R. Pérez-Castillo, “Quantum computing: A new software engineering golden age,” *ACM SIGSOFT Software Engineering Notes*, vol. 45, no. 3, pp. 12–14, 2020.
- [225] M. T. Baldassarre, V. S. Barletta, D. Caivano, and M. Scalera, “Privacy oriented software development,” in *Quality of Information and Communications Technology* (M. Piattini, P. Rupino da Cunha, I. García Rodríguez de Guzmán, and R. Pérez-Castillo, eds.), (Cham), pp. 18–32, Springer International Publishing, 2019.
- [226] V. S. Barletta, D. Caivano, M. De Vincentiis, A. Ragone, M. Scalera, and M. Á. S. Martín, “V-soc4as: A vehicle-soc for improving automotive security,” *Algorithms*, vol. 16, no. 2, 2023.
- [227] G. Dupont, J. den Hartog, S. Etalle, and A. Lekidis, “A survey of network intrusion detection systems for controller area network,” in *2019 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, pp. 1–6, IEEE, 2019.
- [228] V. S. Barletta, D. Caivano, A. Nannavecchia, and M. Scalera, “Intrusion detection for in-vehicle communication networks: An unsupervised kohonen som approach,” *Future Internet*, vol. 12, no. 7, 2020.
- [229] H. Lee, K. Choi, K. Chung, J. Kim, and K. Yim, “Fuzzing can packets into automobiles,” in *2015 IEEE 29th International Conference on Advanced Information Networking and Applications*, pp. 817–821, IEEE, 2015.

- [230] K. Iehira, H. Inoue, and K. Ishida, “Spoofing attack using bus-off attacks against a specific ecu of the can bus,” in *2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, pp. 1–4, IEEE, 2018.
- [231] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, *et al.*, “Experimental security analysis of a modern automobile,” in *2010 IEEE symposium on security and privacy*, pp. 447–462, IEEE, 2010.
- [232] C. Miller and C. Valasek, “A survey of remote automotive attack surfaces,” *black hat USA*, vol. 2014, p. 94, 2014.
- [233] Z. Bi, G. Xu, G. Xu, M. Tian, R. Jiang, and S. Zhang, “Intrusion detection method for in-vehicle can bus based on message and time transfer matrix,” *Security and Communication Networks*, vol. 2022, 2022.
- [234] M. Angelelli, S. Arima, C. Catalano, and E. Ciavolino, “Cyber-risk perception and prioritization for decision-making and threat intelligence,” *arXiv preprint arXiv:2302.08348*, 2023.
- [235] M. A. Serrano, L. E. Sánchez, A. Santos-Olmo, D. García-Rosado, C. Blanco, V. S. Barletta, D. Caivano, and E. Fernández-Medina, “Minimizing incident response time in real-world scenarios using quantum computing,” *Software Quality Journal*, pp. 1–30, 2023.
- [236] S. Mukkamala and A. H. Sung, “Detecting denial of service attacks using support vector machines,” in *The 12th IEEE International Conference on Fuzzy Systems, 2003. FUZZ’03.*, vol. 2, pp. 1231–1236, IEEE, 2003.
- [237] V. Tanksale, “Intrusion detection for controller area network using support vector machines,” in *2019 IEEE 16th International Conference on Mobile Ad Hoc and Sensor Systems Workshops (MASSW)*, pp. 121–126, IEEE, 2019.
- [238] E. Seo, H. M. Song, and H. K. Kim, “Gids: Gan based intrusion detection system for in-vehicle network,” in *2018 16th Annual Conference on Privacy, Security and Trust (PST)*, pp. 1–6, Aug 2018.
- [239] H. M. Song, J. Woo, and H. K. Kim, “In-vehicle network intrusion detection using deep convolutional neural network,” *Vehicular Communications*, vol. 21, p. 100198, 2020.
- [240] F. Sommer, J. Dürrewang, and R. Kriesten, “Survey and classification of automotive security attacks,” *Information*, vol. 10, no. 4, p. 148, 2019.
- [241] Microsoft, “The STRIDE Threat Model.” [https://learn.microsoft.com/en-us/previous-versions/commerce-server/ee823878\(v=cs.20\)](https://learn.microsoft.com/en-us/previous-versions/commerce-server/ee823878(v=cs.20)). Accessed on 2023-05-17.
- [242] MITRE, “Common Weakness Enumeration.” <https://cwe.mitre.org/>. Accessed on 2023-05-17.

- [243] C. Miller and C. Valasek, “Adventures in automotive networks and control units,” *Def Con*, vol. 21, no. 260-264, pp. 15–31, 2013.
- [244] M. T. Baldassarre, V. Santa Barletta, D. Caivano, and A. Piccinno, “Integrating security and privacy in hcd-scrum,” in *CHIItaly 2021: 14th Biannual Conference of the Italian SIGCHI Chapter*, pp. 1–5, 2021.
- [245] M. T. Baldassarre, V. S. Barletta, D. Caivano, A. Piccinno, and M. Scalera, “Privacy Knowledge Base for Supporting Decision-Making in Software Development,” in *Sense, Feel, Design*, vol. , (Cham, Switzerland), pp. 147–157, Springer, March 2022.
- [246] M. Saltarella, G. Desolda, R. Lanzilotti, and V. S. Barletta, “Translating privacy design principles into human-centered software lifecycle: A literature review,” *International Journal of Human–Computer Interaction*, pp. 1–19, 2023.
- [247] T. Di Mascio, F. Caruso, L. Tarantino, and G. Valente, “Monica vision: An approach, a model and the interactive tools for cyber-physical systems designers,” in *CHIItaly 2021: 14th Biannual Conference of the Italian SIGCHI Chapter*, CHIItaly ’21, (New York, NY, USA), Association for Computing Machinery, 2021.
- [248] D. Caivano, M. Fernández-Ropero, R. Pérez-Castillo, M. Piattini, and M. Scalera, “Artifact-based vs. human-perceived understandability and modifiability of refactored business processes: an experiment,” *Journal of Systems and Software*, vol. 144, pp. 143–164, 2018.
- [249] S. Paiho, P. Tuominen, J. Rökman, M. Ylikerälä, J. Pajula, and H. Siikavirta, “Opportunities of collected city data for smart cities,” *IET Smart Cities*, vol. 4, no. 4, pp. 275–291, 2022.
- [250] V. S. Barletta, P. Buono, D. Caivano, G. Dimauro, and A. Pontrelli, “Deriving smart city security from the analysis of their technological levels: a case study,” in *2021 IEEE International Conference on Omni-Layer Intelligent Systems (COINS)*, pp. 1–6, 2021.
- [251] R. Sánchez-Corcuera, A. Nuñez-Marcos, J. Sesma-Solance, A. Bilbao-Jayo, R. Mulero, U. Zulaika, G. Azkune, and A. Almeida, “Smart cities survey: Technologies, application domains and challenges for the cities of the future,” *International Journal of Distributed Sensor Networks*, vol. 15, no. 6, p. 1550147719853984, 2019.
- [252] V. S. Barletta, P. Buono, D. Caivano, G. Dimauro, and A. Pontrelli, “An overview on the security technological levels in the italian smart cities,” *Proceedings <http://ceur-ws.org> ISSN*, vol. 1613, p. 0073, 2020.
- [253] C. Catalano, P. Afrune, M. Angelelli, G. Maglio, F. Striani, and F. Tommasi, “Security testing reuse enhancing active cyber defence in public administration,” in *ITASEC*, pp. 120–132, 2021.

- [254] L. Zhuhadar, E. Thrasher, S. Marklin, and P. O. de Pablos, “The next wave of innovation—review of smart cities intelligent operation systems,” *Computers in Human Behavior*, vol. 66, pp. 273–281, 2023.
- [255] A. Bhowmick, E. Francellino, L. Glehn, R. Lored, P. Nesbitt, and S. W. Yu, “IBM intelligent operations center for smarter cities administration guide,” *IBM Corporation, International Technical Support Organization*, 2012.
- [256] A. Hwoij, A. Khamaiseh, and M. Ababneh, “SIEM architecture for the internet of things and smart city,” in *International Conference on Data Science, E-learning and Information Systems 2021*, pp. 147–152, ACM, 2021.
- [257] M. Babar, M. U. Tariq, and M. A. Jan, “Secure and resilient demand side management engine using machine learning for iot-enabled smart grid,” *Sustainable Cities and Society*, vol. 62, p. 102370, 2020.
- [258] D. Chen, P. Wawrzynski, and Z. Lv, “Cyber security in smart cities: A review of deep learning-based applications and case studies,” *Sustainable Cities and Society*, vol. 66, p. 102655, 2021.
- [259] F. Kurniawan, A. P. Wibawa, S. M. Susiki, and M. Hariadi, “MAKASSAR SMART CITY OPERATION CENTER PRIORITY OPTIMIZATION USING FUZZY MULTI-CRITERIA DECISION- MAKING,” in *International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*, 2017.
- [260] Jose D. Martin-Guerrero and Lucas Lamata, “Quantum machine learning,” in *ESANN 2020 Proceedings*, pp. 257–266, i6doc.com, 2020.
- [261] ISO, “ISO/SAE 21434:2021 - road vehicles — cybersecurity engineering,” 2021.
- [262] A. Sequeira, L. P. Santos, and L. S. Barbosa, “Policy gradients using variational quantum circuits,” *Quantum Mach. Intell.*, vol. 5, pp. 1–15, June 2023.
- [263] M. T. Baldassarre, V. S. Barletta, D. Caivano, and M. Scalera, “Privacy Oriented Software Development,” in *Quality of Information and Communications Technology*, (Cham, Switzerland), pp. 18–32, Springer, August 2019.
- [264] M. T. Baldassarre, V. S. Barletta, D. Caivano, and M. Scalera, “Integrating security and privacy in software development,” *Software Qual. J.*, vol. 28, pp. 987–1018, September 2020.
- [265] S. Wiesner, “Conjugate coding,” *SIGACT News*, vol. 15, p. 78–88, 01 1983.

- [266] A. K. Ekert, “Quantum cryptography based on bell’s theorem,” *Phys. Rev. Lett.*, vol. 67, pp. 661–663, 08 1991.
- [267] ‘OWASP’, “OWASP Top 10:2021,” May 2023. [Online; accessed 5. Mar. 2024].
- [268] H. Rauch, M. Suda, and S. Pascazio, “Decoherence, dephasing and depolarization,” *Physica B: Condensed Matter*, vol. 267-268, pp. 277–284, 1999.
- [269] J.-H. Hoepman, “Privacy design strategies,” 2012.
- [270] J.-H. Hoepman, “Privacy design strategies (the little blue book),” 2022.
- [271] S. o. I. UC Berkeley, “Privacy Patterns.”
- [272] C. Thomborson, “Privacy patterns,” in *2016 14th (PST)*, pp. 656–663, 2016.
- [273] N. Acharya and S. M. Saeed, “A lightweight approach to detect malicious/unexpected changes in the error rates of nisq computers,” in *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pp. 1–9, 2020.
- [274] G. J. H. González and C. A. Paradela, “Quantum Agile Development Framework,” in *Quality of Information and Communications Technology*, pp. 284–291, August 2020.
- [275] ‘CIS’, “CIS Benchmarks™,” March 2024.
- [276] A. A. Khan, M. A. Akbar, V. Lahtinen, M. Paavola, M. Niazi, M. N. Alatawi, and S. D. Alotaibi, “Agile meets quantum: a novel genetic algorithm model for predicting the success of quantum software development project,” *Autom. Softw. Eng.*, vol. 31, pp. 34–39, April 2024.
- [277] A. Satapathy, S. S. Sahoo, H. Raj, Y. Gupta, and S. Maity, “Website on smart project management system by application of scrum technology,” in *Intelligent and Cloud Computing* (D. Mishra, R. Buyya, P. Mohapatra, and S. Patnaik, eds.), (Singapore), pp. 675–683, Springer Singapore, 2021.
- [278] Z. Wang, “Modelling and simulation of scrum team strategies: A multi-agent approach,” in *Software Engineering Perspectives in Intelligent Systems* (R. Silhavy, P. Silhavy, and Z. Prokopova, eds.), (Cham), pp. 32–63, Springer International Publishing, 2020.
- [279] E. P. Wonohardjo, R. F. Sunaryo, and Y. Sudiyono, “A systematic review of scrum in software development,” 2019.
- [280] R. Sobral and F. Moreira, “Scrum: From rugby to teamwork in higher education,” in *ICERI2018 Proceedings*, 11th annual International Conference of Education, Research and Innovation, pp. 6407–6410, IATED, 12-14 November, 2018 2018.

- [281] G. R. Stopa and C. L. Rachid, “Scrum: metodologia ágil como ferramenta de gerenciamento de projetos,” 2019.
- [282] C. Feio, N. Santos, N. Escravana, and B. Pacheco, “An empirical study of devsecops focused on continuous security testing,” in *2024 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pp. 610–617, 2024.
- [283] A. Kulakarni, “Integrating security into the devops pipeline,” *International Journal For Science Technology And Engineering*, vol. 12, no. 6, pp. 1861–1866, 2024.
- [284] O. O. Abiona, O. J. Oladapo, O. T. Modupe, O. C. Oyeniran, A. O. Adewusi, and A. M. Komolafe, “The emergence and importance of devsecops: Integrating and reviewing security practices within the devops pipeline,” *World Journal of Advanced Engineering Technology and Sciences*, 2024.
- [285] M. K. Kushwaha, P. David, and G. Suseela, “Automation and devsecops: Streamlining security measures in financial system,” in *2024 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, pp. 1–6, 2024.
- [286] M. S. Sachdeva, “Mlops: Revolutionizing ai development and deployment,” *International Journal For Multidisciplinary Research*, 2024.
- [287] A. Chakraborty, S. Das, and K. Gary, “Machine learning operations: A mapping study,” 2024.
- [288] N. E. Vijayan, “Building scalable mlops: Optimizing machine learning deployment and operations,” *Indian Scientific Journal Of Research In Engineering And Management*, vol. 08, no. 10, pp. 1–9, 2024.
- [289] L. Faubel, T. Woudsma, L. Methnani, A. G. Ghezeljhomeidan, F. Buelow, K. Schmid, W. D. van Driel, B. Kloepper, A. Theodorou, M. Nosratinia, and M. Bang, “A mlops architecture for xai in industrial applications,” in *2024 IEEE 29th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 1–4, 2024.
- [290] P. Gupta and A. Bagchi, *MLOps: Machine Learning Operations*, pp. 489–518. Cham: Springer Nature Switzerland, 2024.
- [291] . S. J. Schwaber Ken, “The Scrum Guide,” November 2017.
- [292] J. Sutherland, *Scrum: The Art of Doing Twice the Work in Half the Time*. London, England, UK: Random House Business Books, 2015.
- [293] K. Schwaber, *Agile Project Management With Scrum*. USA: Microsoft Press, 2004.

- [294] M. Sami, A. Ahmed, and S. Jufout, *Intrusion Detection in Controller Area Network Bus Using Deep Neural Network*. PhD thesis, California State University, Long Beach, 2020. AAI27833202.
- [295] T. Hastie, S. Rosset, J. Zhu, and H. Zou, “Multi-class adaboost,” *Statistics and its Interface*, vol. 2, no. 3, pp. 349–360, 2009.
- [296] L. Breiman, “Random forests,” *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.
- [297] A. Mott, J. Job, J.-R. Vlimant, D. Lidar, and M. Spiropulu, “Solving a higgs optimization problem with quantum annealing for machine learning,” *Nature*, vol. 550, no. 7676, pp. 375–379, 2017.
- [298] J. Ashraf, M. Keshk, N. Moustafa, M. Abdel-Basset, H. Khurshid, A. D. Bakhshi, and R. R. Mostafa, “Totbot-ids: A novel statistical learning-enabled botnet detection framework for protecting networks of smart cities,” *Sustainable Cities and Society*, vol. 72, p. 103041, 2021.
- [299] T. Booi, I. Chiscop, E. Meeuwissen, N. Moustafa, and F. den Hartog, “ToN_IoT: The Role of Heterogeneity and the Need for Standardization of Features and Attack Types in IoT Network Intrusion Data Sets,” *IEEE Internet of Things Journal*, vol. PP, pp. 1–1, 05 2021.
- [300] N. Moustafa, “New generations of internet of things datasets for cybersecurity applications based machine learning: Ton_iot datasets,” in *eResearch Australasia Conference, Brisbane, Australia*, 2019.
- [301] N. Moustafa, “A systemic iot-fog-cloud architecture for big-data analytics and cyber security systems: A review of fog computing,” 2019.
- [302] N. Moustafa, M. Ahmed, and S. Ahmed, “Data analytics-enabled intrusion detection: Evaluations of ton_iot linux datasets,” 2020.
- [303] N. Moustafa, M. Keshk, E. Debie, and H. Janicke, “Federated ton_iot windows datasets for evaluating ai-based security applications,” 2020.
- [304] N. Moustafa, “A new distributed architecture for evaluating ai-based security systems at the edge: Network ton_iot datasets,” *Sustainable Cities and Society*, vol. 72, p. 102994, 2021.
- [305] A. Alsaedi, N. Moustafa, Z. Tari, A. Mahmood, and A. Anwar, “Ton_iot telemetry dataset: A new generation dataset of iot and iiot for data-driven intrusion detection systems,” *IEEE Access*, vol. 8, pp. 165130–165150, 2020.
- [306] V. S. Barletta, F. Cassano, A. Pagano, and A. Piccinno, “New perspectives for cyber security in software development: when end-user development meets artificial intelligence,” in *2022 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*, pp. 531–534, 2022.

- [307] E. C. P. Neto, S. Dadkhah, R. Ferreira, A. Zohourian, R. Lu, and A. A. Ghorbani, “Ciciot2023: A real-time dataset and benchmark for large-scale attacks in iot environment,” *Sensors*, vol. 23, no. 13, 2023.
- [308] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [309] P. Rebentrost, M. Mohseni, and S. Lloyd, “Quantum support vector machine for big data classification,” *Physical Review Letters*, vol. 113, no. 13, p. 130503, 2014.
- [310] Qiskit contributors, “Qiskit: An open-source framework for quantum computing,” 2023.
- [311] H. Neven, V. S. Denchev, G. Rose, and W. G. Macready, “Training a binary classifier with the quantum adiabatic algorithm,” 2008.
- [312] H. Neven, V. S. Denchev, G. Rose, and W. G. Macready, “Training a large scale classifier with the quantum adiabatic algorithm,” 2009.
- [313] S. Shalev-Shwartz, Y. Singer, and N. Srebro, “Pegasos: Primal estimated sub-gradient solver for svm,” in *Proceedings of the 24th International Conference on Machine Learning, ICML '07*, (New York, NY, USA), p. 807–814, Association for Computing Machinery, 2007.
- [314] “Superflow.” <https://www.ibm.com/docs/en/qsip/7.4?topic=monitoring-superflows>, 2023. Accessed: 2023-09-28.
- [315] G. Wassermann and Z. Su, “Static detection of cross-site scripting vulnerabilities,” in *Proceedings of the 30th international conference on Software engineering*, pp. 171–180, 2008.
- [316] “DVWA damn vulnerable web application.” <https://github.com/digininja/DVWA>, 2017. Accessed: 2023-09-28.
- [317] W. G. Halfond, J. Viegas, A. Orso, *et al.*, “A classification of sql-injection attacks and countermeasures,” in *Proceedings of the IEEE international symposium on secure software engineering*, vol. 1, pp. 13–15, IEEE, 2006.
- [318] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, “Surveying port scans and their detection methodologies,” *The Computer Journal*, vol. 54, no. 10, pp. 1565–1581, 2011.
- [319] G. F. Lyon, *Nmap network scanning: The official Nmap project guide to network discovery and security scanning*. Insecure, 2009.

- [320] K. Dwivedi, M. Haghparast, and T. Mikkonen, “Quantum software engineering and quantum software development lifecycle: a survey,” *Cluster Comput.*, vol. 27, pp. 7127–7145, Sept. 2024.
- [321] J. J. Borking, P. Verhaar, B. M. A. van Eck, P. Siepel, G. W. van Blarckom, R. Coolen, M. Den Uyl, J. Holleman, P. Bison, R. van der Veer, J. Giezen, A. S. Patrick, C. Holmes, J. C. A. van der Lubbe, R. Lachman, S. Kenny, R. Song, K. Cartrysse, J. Huizenga, A. Youssouf, L. Korba, J. M. Kabasele, M. van Breukelen, A. P. Meyer, A. K. Ahluwalia, J. A. Hamers, J. G. E. Olk, A. Ricchi, G. Manzo, and X. Zhou, *Handbook of Privacy and Privacy-Enhancing Technologies The case of Intelligent Software Agents*. ”: ”, 11 2003.
- [322] M. Scheerer, J. Klamroth, S. Garhofer, F. Knäble, and O. Denninger, “Experiences in quantum software engineering,” in *2023 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pp. 552–559, 2023.
- [323] “Post-Quantum Cryptography.” <https://csrc.nist.gov/projects/post-quantum-cryptography>, 2016. Accessed: 2024-05-10.
- [324] V. S. Barletta, D. Caivano, M. De Vincentiis, A. Pal, and F. Volpe, “Automotive knowledge base for supporting vehicle-soc analysts,” in *2023 IEEE International Conference on Metrology for eXtended Reality, Artificial Intelligence, and Neural Engineering (MetroXRaine)*, pp. 1–6, IEEE, 2023.
- [325] K. Petersen, S. Vakkalanka, and L. Kuzniarz, “Guidelines for conducting systematic mapping studies in software engineering: An update,” *Information and Software Technology*, vol. 64, pp. 1–18, 2015.
- [326] Y. Freund and R. E. Schapire, “A decision-theoretic generalization of on-line learning and an application to boosting,” *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997.

Appendices

Appendix A

Quantum and Quantum-Resilient Protocols

A.1 Algorithms and Security Aspects

As quantum algorithms continue to advance, they introduce both unprecedented capabilities and critical security and privacy challenges. While some algorithms pose threats to traditional cryptographic systems, others expose vulnerabilities in quantum and hybrid software architectures.

A.1.1 Shor's Algorithm

Capable of efficiently factoring large integers, Shor's algorithm [34, 82] threatens the security of RSA and other public-key cryptosystems that rely on the difficulty of factorization. Shor's algorithm solves the factorization problem using quantum mechanics principles, specifically through the use of quantum bits (qubits) and quantum entanglement. We present a simplified overview of how it works below:

- *Quantum Superposition*: Begins by placing qubits into a superposition of states, allowing the algorithm to explore multiple possibilities simultaneously, unlike classical algorithms that process one possibility at a time.
- *Period Finding*: The core involves finding the period of a specific function related to the number to be factored. This is done using quantum Fourier transform, which enables the extraction of periodicity efficiently.
- *Measurement*: The qubits are then measured. This measurement collapses the superposition into a definite state, revealing information about the period of the function.
- *Classical Post-Processing*: Once the period is determined, classical algorithms are used to derive the prime factors from this information. This step is significantly faster than classical factorization methods.

A.1.2 Grover's Algorithm

Provides a quadratic speedup for unstructured search problems, impacting symmetric key cryptosystems like AES by reducing their effective key length [35]. This algorithm fundamentally changes how we approach searching through unsorted databases or solving certain types of problems. Presented below is a simplified overview of how Grover's algorithm works.

- *Unstructured Search Problem:* In classical computing, searching for a specific item in an unsorted database of N items requires, on average, $N/2$ queries to find the target item, and in the worst case, N queries. Grover's algorithm, however, allows a quantum computer to find the target item in about $\sqrt{N}$ queries, providing a quadratic speedup.
- *Quantum Superposition:* Grover's algorithm leverages the principle of superposition, and initializes a set of qubits into a superposition state that represents all possible entries in the database.
- *Oracle Function:* The core of Grover's algorithm is the Oracle function, which is a black-box function that can identify the correct answer. When the algorithm queries the oracle, it marks the correct solution by flipping the sign of its amplitude. This operation does not reveal the solution directly but allows the algorithm to distinguish the correct answer from incorrect ones.
- *Amplitude Amplification:* After marking the correct answer, Grover's algorithm amplifies the probability amplitude of the marked state. This is done through a series of operations that involve:
 - *Inversion about the mean:* This step increases the amplitude of the correct solution while decreasing the amplitudes of the incorrect ones.
 - *Repeating the process:* The marking and amplification steps are repeated approximately $\frac{\pi}{4}\sqrt{N}$ times. With each iteration, the probability of measuring the correct solution increases.
- *Measurement:* Once the iterations are complete, the algorithm measures the qubits. Due to the amplification process, the measurement will yield the correct answer with high probability.

Impact on Cryptosystems: The implications of Grover's algorithm extend to symmetric key cryptosystems, such as the Advanced Encryption Standard (AES). Since Grover's algorithm can effectively reduce the key length needed for security, it poses a significant threat to the strength of symmetric encryption. For instance, if a symmetric key length is k , Grover's algorithm would allow an attacker to find the key using $2^{k/2}$ operations instead of 2^k . This effectively halves the security level of the encryption, meaning that a 128-bit key would offer security comparable to a 64-bit key against a quantum adversary.

A.1.3 Quantum Annealing

This approach can transform cryptographic problems into QUBO formats that quantum annealers can solve more efficiently, posing risks to algorithms like SHA-256 and AES [83]. The process can be summarized as follows:

- *Representation as Boolean Expressions:* Cryptographic algorithms frequently encompass a series of intricate Boolean operations. The initial phase in the transformation process entails articulating these operations as Boolean expressions. For instance, operations within algorithms such as AES incorporate substitutions and permutations that can be delineated in terms of Boolean algebra.
- *Linearization of Boolean Expressions:* The Boolean expressions are subsequently linearized. This process involves the conversion of Boolean operations into algebraic forms that can be articulated as linear equations. This phase is pivotal as it transposes the non-linear characteristics of cryptographic operations into a format that can be processed by quantum annealers.
- *QUBO Formulation:* Ultimately, these equations are converted into a QUBO problem. It is designed to minimize a quadratic function of binary variables, which precisely corresponds to the type of problem that a quantum annealer can resolve with efficiency. This phase frequently necessitates the introduction of penalty terms to guarantee that the resolution of the QUBO problem aligns with a solution to the original cryptographic problem.

A.1.4 Quantum Key Distribution (QKD)

Quantum Key Distribution (QKD), a seminal technology in data transmission security, leverages quantum mechanics to establish secure communication channels that are theoretically immune to eavesdropping, offering a promising solution for future-proof encryption [60]. QKD systems utilize the principles of quantum entanglement and superposition to enable two parties to share encryption keys securely, ensuring that any attempt at interception can be detected immediately [84].

Quantum Key Distribution was used as a principal research component on numerous occasions like a cloud-based Smart Grid Advanced Metering Infrastructure system model utilizing QKD cryptography to enhance cloud-based SG AMI security [175], a scalable QKD-based system enabling secure multi-site communication compatible with various QKD technologies [171], and a real-time JAVA implementation of the software needed to complete a QKD protocol along with a modified version of Cascade algorithm [152].

The essence of QKD lies in its ability to securely distribute cryptographic keys between two parties, commonly referred to as Alice and Bob, over an insecure channel. Eavesdropping attempts by a third party (Eve) can be detected due

to the unique properties of quantum mechanics, particularly the no-cloning theorem and the measurement disturbance principle.

QKD protocols typically rely on transmitting quantum states, such as photons, which encode information in properties like polarization or phase. The security of these protocols arises from the fact that any attempt to measure or copy these quantum states disturbs them, thereby alerting Alice and Bob to potential eavesdropping.

The mathematical foundation of QKD is rooted in quantum information theory and linear algebra. Several protocols have been developed, each with unique approaches to secure key distribution.

The BB84 Protocol, introduced by Bennett and Brassard in 1984, is a simple yet widely used QKD protocol [85]:

- **Quantum States:** Information is encoded in qubits using two complementary bases, such as the rectilinear ($|0\rangle$ and $|1\rangle$) and diagonal ($|+\rangle$ and $|-\rangle$) bases.
- **Key Distribution:**
 - Alice randomly chooses a basis for each bit and prepares qubits accordingly.
 - Bob also randomly chooses a basis to measure each received qubit.
 - Due to the principles of quantum mechanics, Bob’s measurement outcomes are accurate only when his chosen basis matches Alice’s preparation basis.
- **Error Detection:**
 - After transmission, Alice and Bob publicly compare a subset of their basis choices. If the bases match, those bits are kept; otherwise, they are discarded.
 - Any discrepancies indicate potential eavesdropping, prompting Alice and Bob to discard the key.
- **Privacy Amplification:** To mitigate information leakage to Eve, Alice, and Bob perform privacy amplification using hash functions to generate a shorter, secure key.

The B92 protocol, proposed by Bennett in 1992, simplifies BB84 by using only two non-orthogonal quantum states for encoding information:

- **Quantum States:** The states used are $|0\rangle$ and $|+\rangle$.
- **Key Distribution:**
 - Alice prepares qubits using one of the two states ($|0\rangle$ or $|+\rangle$) and sends them to Bob.

- Bob uses a single measurement basis to detect the states. If Bob detects a definitive state (e.g., $|0\rangle$ or $|+\rangle$), he records the bit; if his measurement results in ambiguity, the bit is discarded.
- **Eavesdropping Detection:** As in BB84, Alice and Bob compare a subset of the transmitted bits to detect errors caused by potential eavesdropping.

The simplicity of B92 reduces resource requirements while maintaining the core principles of QKD security. However, it relies heavily on accurate detection of the quantum states to ensure robustness [86].

The E91 protocol, developed by Ekert in 1991, leverages quantum entanglement to establish a secure key:

- **Quantum Entanglement:** Alice and Bob share entangled pairs of qubits, generated by a trusted source or through quantum communication channels.
- **Measurement Correlations:**
 - Each party independently performs measurements on their respective qubits using randomly chosen bases (e.g., rectilinear or diagonal).
 - Due to entanglement, the measurement outcomes exhibit strong correlations that can be used to generate a shared secret key.
- **Eavesdropping Detection:** To detect eavesdropping, Alice and Bob test the violation of Bell’s inequalities. If the correlations deviate from expected quantum predictions, the presence of an eavesdropper is inferred.

E91 is significant because it ties the security of QKD to fundamental tests of quantum mechanics, providing both practical key distribution and a tool to explore the nature of entanglement [87].

Security Guarantees

The security of QKD protocols like BB84, B92, and E91 is based on quantum mechanics rather than computational assumptions. Even a quantum eavesdropper cannot intercept or replicate the transmitted qubits without introducing detectable errors. The no-cloning theorem, which states that it is impossible to create an identical copy of an unknown quantum state, ensures the integrity of the transmitted key [89].

Real-World Implementations

QKD has transitioned from theoretical frameworks to practical implementations using optical fiber and satellite-based systems. For instance:

- The Micius satellite successfully demonstrated QKD over long distances, paving the way for global quantum communication [88].
- Companies like ID Quantique and Toshiba have developed commercial QKD systems for secure enterprise communication [89].

Challenges and Future Directions

Despite its promise, QKD faces practical challenges:

- **Cost and Scalability:** Implementing QKD requires specialized hardware, such as single-photon sources and detectors.
- **Distance Limitations:** Signal loss in optical fibers limits the effective range of QKD, although solutions like quantum repeaters are being explored.
- **Integration with Classical Networks:** Seamlessly integrating QKD into existing communication infrastructures is an ongoing area of research.

Future advancements in quantum technologies and protocols are expected to address these challenges, making QKD a cornerstone of secure communication in the quantum era.

A.2 Quantum-Resistant Cryptography

As quantum computing progresses, traditional cryptographic systems face an existential threat, particularly those relying on factorization and discrete logarithm problems. To safeguard digital security in the quantum era, quantum-resistant cryptographic approaches are being actively researched and standardized. These methods leverage mathematical problems believed to be intractable for quantum algorithms, ensuring long-term security. Lattice-based, hash-based, multivariate, and code-based cryptography offer viable alternatives, each with distinct strengths and challenges. As global initiatives like NIST's post-quantum cryptography standardization continue, developing efficient, scalable, and quantum-secure cryptographic solutions remains critical for the future of secure communication and digital infrastructure.

- **Lattice-Based Cryptography:** Offers strong resistance to quantum attacks by relying on the hardness of lattice problems, which are believed to be secure against quantum algorithms [100]. It supports essential cryptographic primitives such as encryption, digital signatures, and key exchange, making it vital for secure communication in a quantum context [101, 102]. Notable schemes like NTRU, Ring-Lizard, and Kyber have demonstrated significant potential in safeguarding digital infrastructures and enhancing blockchain privacy through linkable ring signature schemes [103, 102, 104]. Ongoing research and NIST's standardization efforts highlight its importance for future secure digital communication. Its versatility and efficiency make it suitable for both small embedded systems and large-scale cloud computing, positioning it as a leader in cryptographic research amidst the rise of quantum computing [102].
- **Hash-Based Cryptography:** Utilizes secure hash functions to create secure digital signatures, providing a quantum-resistant alternative to traditional methods [105]. It addresses vulnerabilities in traditional

systems, such as those based on integer factorization and discrete logarithms, which are susceptible to quantum algorithms like Shor's algorithm [103]. Notably, the SPHINCS+ scheme is a stateless hash-based signature recognized for its quantum resistance and is a candidate in the NIST post-quantum cryptography competition [106]. These cryptographic methods ensure security against quantum chosen message attacks and are applicable in various contexts, including digital signatures and designated verifier signature schemes, enhancing privacy in sensitive applications like electronic voting [107, 108]. Despite its advantages, hash-based cryptography faces challenges in implementation complexity and computational efficiency, however, it plays a crucial role in developing secure systems for the quantum era [109, 110].

- **Multivariate and Code-Based Cryptography:** Multivariate and code-based cryptography are two promising approaches in the realm of quantum-resistant cryptography, designed to withstand the computational power of quantum computers. These approaches exploit complex mathematical structures to ensure security against quantum threats [105]. Multivariate cryptography is based upon the complexity of solving systems of multivariate polynomial equations considered NP-hard. This approach is particularly appealing for environments with limited resources, although it may not offer the same level of security as lattice-based or code-based cryptography [101]. However, some multivariate cryptosystems, such as those based on extended clipped Hopfield neural networks, have been shown to be vulnerable to quantum attacks, indicating that not all multivariate approaches are equally secure against quantum threats [111]. Code-based cryptography, on the other hand, as in McEliece and Niederreiter cryptosystems, exploits the difficulty of decoding certain error-correcting codes for its security. This method has been extensively studied and is considered a robust candidate for post-quantum cryptography due to its resistance to both classical and quantum attacks [101, 112]. Despite its strong security guarantees, the practical deployment of code-based cryptography is hindered by the large key sizes required, although recent advancements aim to reduce these sizes without compromising security [112]. Code-based cryptography is also being explored for applications in multi-party computation and digital signatures, offering competitive performance and strong security compared to other post-quantum schemes [113]. Both multivariate and code-based cryptography are part of the broader effort to develop quantum-resistant cryptographic protocols, alongside lattice-based and hash-based methods, to ensure secure communication in a future dominated by quantum computing.