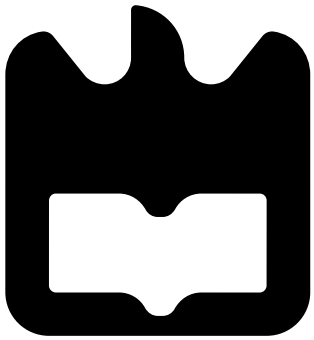
Universidade de Aveiro Departamento de Electrónica, Telecomunicações e Informática,
2025

Zeinab Rahmani

Computação Segura Multiagente Baseada em Tecnologias Quânticas

Secure Multiparty Computation Based on Quantum Technologies



Zeinab Rahmani

Computação Segura Multiagente Baseada em Tecnologias Quânticas

Secure Multiparty Computation Based on Quantum Technologies

Tese apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Doutor em Engenharia Eletrotécnica, realizada sob a orientação científica do Professor Doutor Armando Nolasco Pinto, Professor Catedrático do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, e coorientação do Professor Doutor Luís Soares Barbosa, do Departamento de Informática da Universidade do Minho.

This work is funded by Fundação para a Ciência e a Tecnologia (FCT) through Fundo Social Europeu and through national funds, by the European Regional Development Fund (FEDER), through the Competitiveness and Internationalization Operational Programme (COMPETE 2020) of the Portugal 2020 framework under the International Iberian Nanotechnology Laboratory (INL) Quantum Portugal Initiative PhD Grant with Ref. SFRH/BD/151111/2021. We also acknowledge the support of the Q.Dot (POCI-01-0247-FEDER-039728), QUESTS (UIDB/50008/2020), QuantaGenomics (QuantERA/0001/2021) and IBEX (10.54499/PTDC/CCI-COM/4280/2021) projects.

o júri/the jury

presidente/president	**Professor**
vogais/examining committee	**Professor**
	Professor
orientadores/ supervisors	**Professor Armando Humberto Moreira Nolasco Pinto (Supervisor)** Full Professor, University of Aveiro
	Professor Luís Manuel Dias Coelho Soares Barbosa (Cosupervisor) Full Professor, University of Minho

Acknowledgements

I am grateful to my supervisors for their invaluable guidance and support during my research. I am also grateful to Dr. Johannes Wagner for his insightful advice and mentorship during my Ph.D. internship at Eraneos Analytics Germany.

I express my heartfelt thanks to my husband, Dr. Ehsan Yaghoubi, for his constant encouragement and belief in me. His support has been invaluable in helping me to overcome challenges.

I extend a special thanks to my family and friends, Dr. Diana Laura Borza, Dr. Sara Mantey , and Dr. Romil Patel, whose companionship have been a source of motivation.

I express my appreciation to University of Aveiro, Instituto de Telecomunicações de Aveiro (IT-Aveiro), the International Iberian Nanotechnology Laboratory (INL), Fundação para a Ciência e a Tecnologia (FCT) and Eraneos Analytics Germany for providing me with the resources and welcoming environment necessary to embark on this research.

palavras-chave

Computação Segura Multiagente, criptografia quântica, geração de números aleatórios quânticos, distribuição de chave quântica, distribuição quântica de chaves oblívias, transferência quântica ignorante, computação quântica baseada em medições, redes veiculares, descoberta de medicamentos.

resumo

A Computação Segura Multiparte (SMC) permite que múltiplos indivíduos, cada um com seus dados privados, realizem uma tarefa computacional sem revelar seus dados para os outros. A SMC possui uma ampla gama de casos de uso no mundo real, incluindo *machine learning*, redes veiculares, na área dos serviços financeiros, da defesa, e da saúde. No entanto, as implementações clássicas de SMC enfrentam desafios significativos relacionados à segurança e à eficiência. Os protocolos clássicos de SMC dependem de métodos criptográficos de chave pública, o que introduz custos substanciais computacionais e de comunicação. Além disso, esses protocolos são vulneráveis a ataques de computadores quânticos devido ao algoritmo de Shor. Neste trabalhos procurámos encontrar soluções para o problema da eficiência e segurança, usando tecnologias quânticas das comunicações e da computação. A comunicação quântica, contribui para melhorar a segurança, enquanto a computação quântica, ao aproveitar processadores quânticos, oferece o potencial para melhorar significativamente a eficiência. Na abordagem de comunicação quântica, propomos uma estrutura de SMC Quântica (QSMC) que utiliza três tecnologias avançadas de comunicação quântica—Geração de Números Aleatórios Quânticos (QRNG), Distribuição Quântica de Chaves (QKD) e Distribuição Oblívia de Chaves Quânticas (QOKD)—em conjunto com um Sistema de Gestão de Chaves (KMS) e o protocolo Faster Malicious Arithmetic Secure Computation with Oblivious Transfer (MASCOT). Explorando a estrutura proposta, implementamos dois casos de uso de SMC: Saída da Auto-Estrada em Segurança (SRD) e Previsão de Solubilidade de Fármacos (DSP), aplicados a redes veiculares e à descoberta de fármacos, respectivamente. O SRD facilita a comunicação segura entre veículos, permitindo que mudem de faixa e saiam de rotas preservando informações sensíveis. Conseguimos uma melhoria de 97% na eficiência ao reduzir significativamente os custos de comunicação, enquanto houve um aumento moderado de 42% no custo computacional. O DSP, por outro lado, permite que empresas farmacêuticas treinem colaborativamente uma Rede Neural Convolucional Gráfica (GCN) para prever a solubilidade de moléculas de fármacos, protegendo totalmente seus conjuntos de dados privados. Durante o treino, a função de perda média entre todas as partes foi de 0.0046, enquanto o Erro Quadrático Médio (MSE) no conjunto de testes foi de 1.2, indicando um aprendizado eficaz do modelo. Por fim, explorando a abordagem de computação quântica, propomos dois novos protocolos de SMC para computação de funções lógicas, utilizando a Computação Quântica Baseada em Medidas (MBQC) e qubits únicos. Implementamos os esquemas propostos na plataforma IBM Qiskit e validámos sua viabilidade.

keywords Quantum secure multiparty computation, quantum cryptography, quantum random number generation, quantum key distribution, quantum oblivious key distribution, quantum oblivious transfer, measurement based quantum computing, vehicular networks, drug discovery.

abstract Secure Multiparty Computation (SMC) allows multiple individuals, each having their own private data, to perform a computation task without unveiling their data to others. SMC has a wide range of real-life use cases, including machine learning, vehicular networks, banking, defense, and healthcare. However, classical SMC implementations face significant challenges related to both security and efficiency. Classical SMC protocols rely on public-key cryptographic methods, which introduce substantial computational and communication costs. Additionally, these protocols are vulnerable to quantum computer attacks due to Shor's algorithm. We tackle these challenges by enhancing the security and efficiency of SMC implementations through quantum communication and quantum computing. Quantum communication contributes to enhancing security, while quantum computing, by leveraging quantum processors, offers the potential to significantly improve efficiency. In the quantum communication approach, we propose a Quantum SMC (QSMC) framework that leverages three advanced quantum communication technologies—Quantum Random Number Generation (QRNG), Quantum Key Distribution (QKD), and Quantum Oblivious Key Distribution (QOKD)—in conjunction with a Key Management System (KMS) and the Faster Malicious Arithmetic Secure Computation with Oblivious Transfer (MASCOT) protocol. Exploiting the proposed framework, we implement two essential SMC use cases: Safe Route Departure (SRD) and Drug Solubility Prediction (DSP), applied to vehicular networks and drug discovery, respectively. The SRD facilitates secure vehicular communication, allowing vehicles to safely change lanes and exit routes without compromising sensitive information. We achieved a 97% improvement in efficiency by significantly reducing communication costs, while incurring a moderate 42% increase in computational cost. The DSP, on the other hand, empowers pharmaceutical companies to collaboratively train a Graph Convolutional Network (GCN) to predict the solubility of drug molecules while fully safeguarding their private datasets. During training, the loss function across all parties averaged to 0.0046, while the Mean Squared Error (MSE) on the test set is 1.2, indicating effective model learning. Afterwards, exploiting the quantum computing approach, we propose two novel SMC protocols for Boolean function computation resorting to the Measurement Based Quantum Computing (MBQC) approach and single qubits. We implement the proposed schemes on the IBM Qiskit platform and validate their feasibility.

Contents

List of Figures

Acronyms

1WQC One-Way Quantum Computer.

3DES Triple Data Encryption Standard.

ADC Analog-to-Digital Converter.

AES Advanced Encryption Standard.

AES-NI AES New Instructions.

BGW BenOr-Goldwasser-Wigderson.

BLA Biologics License Application.

BMR Beaver-Micali-Rogaway.

BS Beam Splitter.

CARLA Car Learning to Act.

CBMC-GC Circuit-Based Model Checking - Garbled Circuits.

COPE Correlated Oblivious Product Evaluation.

CTR Counter.

CV Continuous Variable.

DH Diffie-Hellman.

DHKE Diffie-Hellman Key Exchange.

DI Device-Independent.

DoS Denial-of-Service.

DP Differential Privacy.

DSP Drug Solubility Prediction.

DTI Drug–Target Interactions.

DV Discrete Variable.

EB Entanglement-Based.

ECC Elliptic Curve Cryptography.

EMA European Medicines Agency.

EPC Electronic Polarization Controller.

ESOL Estimated Solubility.

ETSI European Telecommunications Standards Institute.

FDA Food and Drug Administration.

FL Federated Learning.

FPGA Field-Programmable Gate Array.

GC Garbled Circuit.

GCN Graph Convolutional Network.

GCNConv GCN Convolution.

GHZ Greenberger-Horne-Zeilinger.

GMW Goldreich-Micali-Wigderson.

HAR Human Activity Recognition.

HE Homomorphic Encryption.

IoV Internet of Vehicles.

KMS Key Management System.

KSID Key Stream ID.

LibSCAPI Library for Secure Computation API.

LO Local Oscillator.

MAC Message Authentication Code.

MASCOT Faster Malicious Arithmetic Secure Computation with Oblivious Transfer.

MBQC Measurement-Based Quantum Computing.

MELLODDY Machine Learning Ledger Orchestration for Drug Discovery.

MitM Man-in-the-Middle.

MP-SPDZ Multi-Protocol SPDZ.

MPCDDI Multiparty Computation-Based Deep Learning Framework for Drug-Drug Interaction.

MPI Message Passing Interface.

MSE Mean Squared Error.

MZM Mach-Zehnder Modulator.

NDA New Drug Application.

NHS National Health Service.

NN Neural Network.

OT Oblivious Transfer.

OTP One-Time-Pad.

OWF One Way Function.

PBS Polarization Beam Splitter.

PKC Public-Key Cryptography.

PRF Pseudo-Random Function.

PRNG Pseudorandom Number Generation.

PSIC Privacy Set Intersection Cardinality.

QBER Quantum Bit Error Rate.

QFL Quantum Federated Learning.

QKD Quantum Key Distribution.

QMP-SPDZ Quantum Multi-Protocol SPDZ.

QOKD Quantum Oblivious Key Distribution.

QoS Quality of Service.

QOT Quantum Oblivious Transfer.

QPU Quantum Processing Unit.

QRNG Quantum Random Number Generator.

QRx QKD Receiver.

QSAR Quantitative Structure–Activity Relationships.

QSMC Quantum Secure Multiparty Computation.

QTx QKD Transmitter.

RIN Relative Intensity Noise.

ROT Random Oblivious Transfer.

RSA Rivest–Shamir–Adleman.

RSU Roadside Unit.

S-OT Simple OT.

SAE Secure Application Entity.

SAFEFL Safe Federated Learning.

SDN Software Defined Networking.

SePCAR Secure and Privacy-Enhancing Protocol for Car Access Provision.

SMC Secure Multiparty Computation.

SMILES Simplified Molecular Input Line Entry System.

SOP State Of Polarization.

SPD Single-Photon Detector.

SPDZ Smart-Pastro-Damgård-Zakarias.

SRD Safe Route Departure.

SS Secret Sharing.

SSL Secure Sockets Layer.

SVP Shortest Vector Problem.

TIA Transimpedance Amplifier.

TQC Teleportation Quantum Computation.

TRL Technology Readiness Level.

UC Universal Composability.

V2I Vehicle-to-Infrastructure.

V2P Vehicle-to-Pedestrian.

V2V Vehicle-to-Vehicle.

V2X Vehicle-to-Everything.

VM Virtual Machine.

VOA Variable Optical Attenuator.

VPriv Vehicle Privacy.

WDM Wavelength-Division Multiplexing.

ZKP Zero-Knowledge Proof.

Chapter 1

Introduction

In this chapter, we provide an introduction to the research topic, discuss the motivation behind conducting this work, and define the research problem. We detail the contributions, enumerate the key outputs, outline the structure of the thesis and provide a conclusion.

In Section 1.1, we provide the motivation behind our research. Section 1.2 defines the problem. Subsequently, Section 1.3 presents the list of goals for this Ph.D. program. Following this, Section 1.4 details the achieved contributions. Section 1.5 enumerates our publications, and in Section 1.6, we present the structure of the thesis chapters.

1.1 Motivation

In today's data-driven age, information serves as a vital resource for scientific development. However, the increasing flow of information also poses significant privacy challenges. One notable example is the UK's National Health Service (NHS) plan to create a centralized database called the *Care.data* [3]. The *Care.data* program aimed to link together various healthcare data sources, including records from general practitioners and hospitals. The goal was to utilize this data to improve healthcare outcomes, identify trends, and allocate resources more effectively. However, the program faced significant criticism and public concern over privacy issues [3, 4]. Many patients were worried about the security of their sensitive medical information and the potential for it to be misused or accessed without their consent. There were also concerns about the possibility of re-identification of individuals from anonymized data, as well as worries about data being sold to third parties. Ultimately, the *Care.data* program was scrapped in 2016 due to these privacy concerns, lack of public trust, and criticism from various stakeholders, including patients, healthcare professionals, and privacy advocates [5].

Secure Multiparty Computation (SMC) offers a revolutionary paradigm for conducting computations over distributed data while preserving the privacy of each party involved [6]. By enabling multiple parties to jointly compute functions over their private inputs without revealing any individual data, SMC provides cross-organizational collaboration, data analysis, and decision-making while safeguarding sensitive information. However, despite its immense potential, SMC still faces numerous challenges and limitations that hinder its widespread adoption and practical implementation [7]. This doctoral thesis seeks to embark on a journey towards advancing the state-of-the-art in SMC, leveraging quantum technologies. By addressing the challenges faced by classical SMC and harnessing the benefits of quantum technology,

this research aims to enable individuals and organizations to collaborate effectively for the betterment of society.

1.2 Problem Definition

SMC is a technology in which a group of n parties $\{P_1, P_2, ..., P_n\}$, each holding a secret input a_i $(1 \leq i \leq n)$, collaboratively calculates a function $f(a_1, a_2, ..., a_n)$, without leaking any information about their secret input to others [7]. In traditional computation models, participants would need to disclose their private data to a trusted third party or each other to perform computations collaboratively. However, in many real-world scenarios, this is not feasible or desirable due to privacy concerns or lack of trust among parties. SMC has a variety of applications in real life. For instance, it can enhance road safety and passengers' convenience in vehicular networks enabling secure collaboration in Vehicle-to-Everything (V2X) communications [8, 9, 10]. Additionally, SMC is widely used in healthcare [11, 12, 13], where it can be integrated with Federated Learning (FL) to facilitate secure collaboration in clinical trials and drug discovery processes [14]. Another application of SMC in healthcare involves genomics data mining, allowing medical professionals to identify genetic variants associated with diseases without sharing raw genomic data [11, 15]. Moreover, SMC can play a role in fraud detection [16], risk assessment, and compliance monitoring within the banking, financial, and regulatory sectors [17].

Despite the numerous real-world applications, the practical implementation of SMC has been hindered due to the limitations of existing algorithms. In general, classical SMC protocols suffer from two main challenges: security and efficiency. One of the main security challenges in classical SMC arises from the potential threat posed by quantum computers, particularly due to Shor's algorithm [18]. Shor's algorithm has the capability to efficiently factor large integers and address the discrete logarithm problem. These problems underlie the security of widely used cryptographic schemes, such as Rivest–Shamir–Adleman (RSA) [19], Diffie-Hellman Key Exchange (DHKE) [20], and Elliptic Curve Cryptography (ECC) [21]. This causes a significant risk to the security of classical SMC protocols. The second challenge arises due to the reliance of classical SMC protocols on Public-Key Cryptography (PKC) [22], which can be computationally intensive, especially for operations such as encryption, decryption, and key generation. These operations typically involve complex mathematical computations, which can lead to performance bottlenecks and scalability issues.

To address security challenges in classical SMC, post-quantum cryptography [23] aims to develop new cryptographic primitives and protocols that remain secure even in the presence of quantum computers. These primitives are typically based on mathematical challenges that are considered difficult even for quantum computers to address. Such problems include lattice-based cryptography [24], code-based cryptography [25], and hash-based cryptography [26]. Although post-quantum public-key protocols are being developed to address these challenges, it remains to be proven that they can offer a solution that complies with the security and efficiency requirements of SMC [23].

Quantum-based SMC emerges as a promising paradigm that leverages the unique properties of quantum mechanics, such as no cloning, superposition, and entanglement along with cryptographic techniques to achieve higher levels of efficiency and security. Generally, two primary quantum approaches are being developed: quantum communication-based approach [27, 28, 10, 29], and quantum computing-based approach [30, 31, 32, 33, 34]. Quantum

communication-based SMC exhibits a higher Technology Readiness Level (TRL), indicating a significant degree of maturity and readiness for practical application. Within this approach, quantum communication technologies, such as Quantum Random Number Generator (QRNG) [35], Quantum Key Distribution (QKD) [36], and Quantum Oblivious Key Distribution (QOKD) [22] have been extensively researched and developed with successful implementations. In this context, quantum communication technologies are integrated into SMC protocols to combine the strengths of quantum and classical cryptographic techniques. For instance, in [27], QKD and QOKD technologies are integrated into classical Faster Malicious Arithmetic Secure Computation with Oblivious Transfer (MASCOT) [8] protocol to provide a lane change service in vehicular networks. In [11], authors computed phylogenetic trees of SARS-CoV-2 genomes by integrating QRNG, QKD, and QOKD into Yao protocol. On the other hand, the quantum computing-based approach deploys quantum computers and processors for its computation, leading to higher efficiency. Within this approach, researchers have investigated different quantum resources such as entangled particles [37, 38, 39] and single qubits [31, 40, 41, 33], to achieve higher performance. For instance, in [30], multiple schemes for the private computation of Boolean functions are proposed utilizing the entanglement of Greenberger-Horne-Zeilinger (GHZ) state through Measurement-Based Quantum Computing (MBQC) [42]. In [31], authors suggested a new approach to compute pairwise AND function by employing single qubit measurements and linear classical computing. Quantum computing-based SMC currently demonstrates a lower TRL, indicating that it is still in the early stages of development. While quantum computing holds immense potential for enhancing the efficiency of SMC protocols, the current implementations face significant challenges caused by quantum decoherence [43]. Decoherence, driven by interactions with the surrounding environment, can introduce noise during computations, jeopardizing the reliability and accuracy of quantum algorithms. To address these challenges, researchers are actively pursuing the development of robust error correction methods [44] to mitigate the effects of decoherence and other error sources within quantum systems. Furthermore, advancements in fault-tolerant quantum computing [45] aim to build quantum computers that can operate reliably even in the presence of errors.

1.3 Goals

This Ph.D. thesis is developed under the scope of the following objectives:

1. **SMC with Quantum Communication Resources**

   Classical SMC relies heavily on PKC, resulting in potential security vulnerabilities and inefficiencies. We aim to propose a Quantum Secure Multiparty Computation (QSMC) framework to enable secure computation among parties. The QSMC framework leverages quantum communication technologies, whose security is inherently guaranteed by the principles of quantum mechanics.

2. **Exploring Practical Applications of SMC**

   SMC has a wide range of real-world applications that, when implemented effectively, could significantly improve human life. This work aims to use the proposed QSMC framework to implement privacy-preserving services, particularly in the domains of vehicular networks and healthcare.

3. **SMC with Quantum Computing Resources**

   While quantum computers are not yet widely accessible, their immense computational power holds the potential to revolutionize SMC protocols, significantly boosting their security and efficiency. Within this objective, we aim to investigate how quantum computing resources can be utilized for enhanced SMC implementations.

1.4 Main Contributions

The main contributions of this Ph.D. program are encapsulated as follows:

1. We develop a QSMC framework, utilizing three advanced quantum communications technologies known as QRNG, QKD, and QOKD in conjunction with a Key Management System (KMS) and the MASCOT SMC protocol. Using the proposed QSMC framework, participants can communicate freely to exchange different sorts of data in a way that their privacy is fully guaranteed. The proposed QSMC framework provides unconditional security even against quantum computer attacks. This work is published in [10].

2. Utilizing the proposed QSMC framework, we implemented two essential use cases of SMC in vehicular networks and health care: Safe Route Departure (SRD) and Drug Solubility Prediction (DSP).

   The SRD significantly enhances road safety by providing proactive measures to prevent accidents and ensure safe navigation for drivers. Within this service, vehicles communicate freely in a way that nothing about their private data (ID, location, velocity, etc.) is leaked to other vehicles. This work is published in [27].

   The DSP empowers pharmaceutical companies to perform computations on confidential datasets while fully safeguarding against data breaches. By integrating SMC techniques with federated machine learning methods, we enhance the security of training processes. This collaborative approach facilitates the development of new drugs, ultimately benefiting individuals in need of medical treatment. This work is published in [46].

3. By resorting to the quantum computing approach, we proposed two quantum-based SMC protocols designed for computing binary Boolean functions. The first protocol adopts the MBQC approach and leverages a three-qubit GHZ state. In contrast, the second protocol utilizes single qubits for Boolean function computation. We designed and implemented the corresponding quantum circuits on the IBM QisKit platform, showcasing the practical feasibility and correctness of these protocols. These works are published in [47, 48].

1.5 List of Publications

The findings of this Ph.D. were published in the following journals and conferences.

Journal papers

1. **Zeinab Rahmani**, Johannes Wagner, Diogo Matos, Ehsan Yaghoubi, Armando N. Pinto, and Luis S. Barbosa. Quantum-Secured Federated Learning for Solubility

Prediction in Drug Molecules. Under review in Quantum Machine Intelligence, 2025.

2. **Zeinab Rahmani**, Armando N. Pinto, and Luis S. Barbosa. Secure two-party computation via measurement-based quantum computing. Quantum Information Processing, 23(6):221-236, 2024. Springer. 10.1007/s11128-024-04433-7.
3. **Zeinab Rahmani**, Luis S. Barbosa, and Armando N. Pinto. Quantum privacy-preserving service for secure lane change in vehicular networks. IET Quantum Communication, 4(3):103–111, 2023. Wiley. 10.1049/qtc2.12059.

Conferences

1. **Zeinab Rahmani**, Armando N. Pinto and Luis S. Barbosa. Private computation of Boolean functions using single qubits. Second Workshop on Quantum Computing and Communication within 15th International conference on parallel processing and applied mathematics, Ostrava, Czech Republic, 1-12, 2024. Springer. 10.1007/978-3-031-85700-3_22.
2. **Zeinab Rahmani**, Luis S. Barbosa, and Armando N. Pinto. Collision warning in vehicular networks based on quantum secure multiparty computation. Workshop de comunicação e computação quântica (WQuantum), Fortaleza, Brazil, 19-24, 2022. Sociedade Brasileira de Computação. 10.5753/wquantum.2022.223569.

1.6 Outline

This thesis is organized as follows.

- Chapter 2: We provide a literature review for the two fundamental approaches to implement SMC: classical and quantum SMC. Our review emphasizes key studies and methodologies within each approach. Furthermore, we discuss foundational concepts upon which the research within this thesis is built.

- Chapter 3: We explain the architecture of the proposed QSMC framework that leverages three advanced quantum communication technologies—QRNG, QKD, and QOKD—in conjunction with a KMS and MASCOT protocol. Additionally, we describe how different components of the QSMC work together to facilitate secure collaborations.

- Chapter 4: First, we define the SRD service and explain how it can secure inter-vehicular communications in a network by exploiting the QSMC framework. When compared to classical implementation, the proposed SRD service showcases a significant efficiency boost with a 97% reduction in necessary communication resources, accompanied by a modest 42% increase in computational resources. Afterward, we elaborate on the DSP use case in which QSMC framework is utilized to develop a quantum-based FL platform that enables multiple pharmaceutical companies to collaboratively train a deep learning model on their private datasets while ensuring the privacy of both the training data and the model parameters. Within this use case, we trained a Graph Convolutional Network (GCN) to predict the solubility of drug molecules using the Estimated Solubility (ESOL) dataset. During training, the loss function across all parties averaged to 0.0046, indicating effective model training, while the Mean Squared Error (MSE) on the test

set is 1.2. Furthermore, we evaluated our proposed DSP use case against a range of attacks. The results show that the performance remains stable, even in the presence of adversarial behavior.

- Chapter 5: Resorting to quantum computing approach, we propose a quantum SMC protocol using the correlations of the GHZ state to compute binary Boolean functions. Our method introduces an additional random Z-phase rotation to the GHZ qubits to increase the protocol's security. The security and efficiency analyses show that we have achieved a higher security level while utilizing the same quantum resources and preserving the existing complexity. We implement the proposed scheme on the IBM QisKit platform and validated its feasibility through consistent and reliable results. Expanding on this work, we introduce a quantum SMC protocol for Boolean functions computation utilizing single qubits. Complexity analyses demonstrate a reduction of 66.7% in required quantum resources, achieved by utilizing single qubits instead of multi-particle entangled states. However, the quantum communication cost has increased by 40% due to the amplified exchange of qubits among participants. We implemented the corresponding circuit on the IBM QisKit platform and validate its feasibility through consistent results.

- Chapter 6: We summarize the main conclusions resulting from this Ph.D. thesis and explore potential avenues for future research.

Chapter 2

Secure Multiparty Computation

Before delving into the main contents of this thesis, we discuss foundational concepts upon which the research within this thesis is built. This review chapter serves to provide readers with an understanding of the principles and frameworks that will be explored in the subsequent chapters.

Section 2.1 defines the concept of SMC. Section 2.2 provides a thorough literature review, discussing seminal works and contemporary research findings in SMC across two domains: classical and quantum SMC. Following this, in Section 2.3, we delve into the practical applications of SMC, highlighting its significance across various domains. Subsequently, in Section 2.4, we expound upon the foundational concepts crucial to understanding SMC in depth. Section 2.5 explains the earliest fundamental building block for SMC known as Yao Garbled Circuit (GC). Section 2.6 concludes the chapter.

2.1 Definition

SMC is a cryptographic technique that allows untrusted parties to compute a function retaining the privacy of their inputs and outputs [6]. For example, consider a scenario where Alice faces the challenge of investigating a potential genetic disease [7]. She possesses a DNA sample and suspects that she may have a genetic condition. Given that Bob holds a database containing DNA patterns associated with various diseases, Alice seeks Bob's assistance in diagnosing her condition. However, Alice and Bob are both concerned about the privacy of their assets. Alice is concerned about sharing her DNA sample directly with Bob. Bob is also concerned about providing access to his dataset. In an SMC scenario, Alice and Bob would engage in a collaborative computation protocol that allows them to determine the diagnosis while ensuring the privacy of Alice's DNA sample and Bob's dataset.

Let us consider a scenario where there exist N entities aiming to collectively compute a function, denoted as f. This function f relies on input parameters $x_1, x_2, \ldots, x_N$, each of which is privately held by one of the N parties involved. This computation can be described by

$$f\left(x_1, x_2, \ldots, x_N\right) = \left(y_1, y_2, \ldots, y_N\right), \tag{2.1}$$

where $y_1, ..., y_n$ represent the respective outputs of the function. The pictorial representation of this computation is shown in Fig. 2.1. Calculating the function f is straightforward in theory, provided that the involved parties are not concerned with keeping their inputs and outputs private. Each party can carry out the computation independently if they exchange

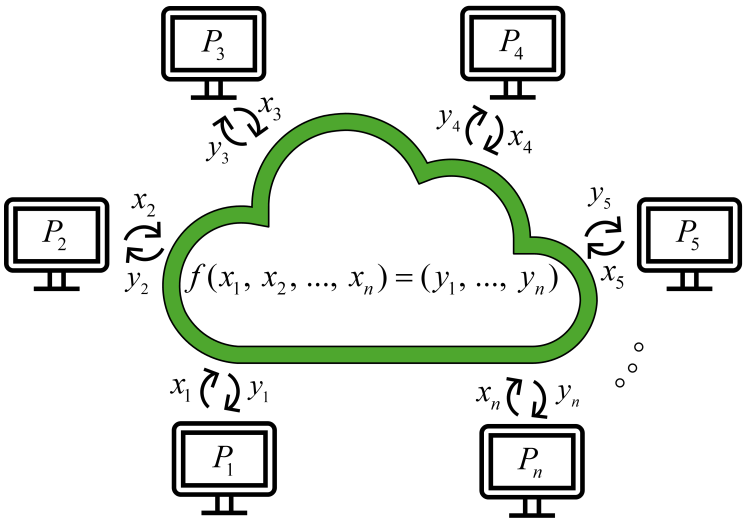


Figure 2.1: Overview of secure multiparty computation.

their inputs with one another and have access to the function f. An alternative method involves assigning the computation to a third-party entity, which collects the required inputs from all participants, carries out the calculation, and then shares the resulting output. This method remains feasible as long as the N parties can mutually agree on a trustworthy external entity. However, if the external entity is compromised or if the N parties cannot reach a consensus on who should perform this role, the process may become impossible to execute. Furthermore, legal requirements often impose restrictions on certain organizations, such as healthcare providers, preventing them from sharing sensitive data with external entities, even if those entities are deemed trustworthy.

The aim of SMC is to empower parties to conduct desired computations without relying on an external trusted entity, while still maintaining the same level of privacy and correctness that would be provided by an ideal trusted external entity.

2.2 Literature Review

In the 1980s, Andrew Yao presented the concept of SMC, in which two untrusted parties compute a GC while retaining the privacy of their inputs [6]. Later, the Yao protocol was extended to involve more than two parties by Goldreich-Micali-Wigderson (GMW) protocol [49]. GMW is based on Secret Sharing (SS) [50] which is used to distribute a secret among multiple parties. In this approach, information is shared such that certain groups of parties (known as qualified sets) can reconstruct it, while smaller groups (known as unqualified sets) are unable to infer anything from the shares they receive. As an example, consider additive SS. A number x is split into n shares $x_1, x_2, \ldots, x_n$ and distributed among n parties $P_1, \ldots, P_n$ such that $\sum_i x_i = x$. GMW also uses Zero-Knowledge Proof (ZKP) [51] to avoid malicious behaviours. For instance, if a dishonest player is detected, the computation can continue with the dishonest player eliminated, or their input can be revealed for further scrutiny. This ensures that the computation can proceed securely even in the presence of malicious adversaries. Numerous protocols for multiparty scenarios, including the Beaver-

Micali-Rogaway (BMR) protocol [52], which enhances Yao's passively secure protocol for active security, and the BenOr-Goldwasser-Wigderson (BGW) protocol [53], which utilizes SS and targets arithmetic circuits, have been developed. Afterward, protocols based on preprocessing models in which expensive computations are delegated to the offline phase were designed to accelerate the computation process [54, 55, 56]. In this model, parties generate correlated randomness that can be utilized during the online phase. This randomness is independent of both the inputs and the circuit. Subsequently, during the online phase, the desired circuit is evaluated. The first protocol in the preprocessing model was Bendlin-Damgard-Orlandi-Zakarias (BDOZ) protocol [54]. Shortly after, the Smart-Pastro-Damgård-Zakarias (SPDZ) protocol was presented [55, 56]. Both of these protocols are based on Homomorphic Encryption (HE) [57]. HE is a cryptographic method enabling computations on encrypted data without requiring prior decryption. They also use Message Authentication Code (MAC) to guarantee that they remain secure in the UC framework [58] even against a malicious majority. MAC is used for verifying the integrity and authenticity of a message. It ensures that the message has not been altered and comes from a legitimate source. There are also protocols in the preprocessing model that use Oblivious Transfer (OT) instead of HE to implement the offline phase. OT enables a sender to transmit one of multiple private messages to a receiver, ensuring the receiver can access only one message without disclosing to the sender which one was selected. OT is a building block for many cryptographic protocols. One of the OT-based protocols is TinyOT [59], a two-party protocol secure against an active adversary. This protocol was later extended to the multiparty case [60, 61]. Subsequently, the MASCOT protocol [8] was introduced within a preprocessing model, enabling secure computation of arithmetic circuits over finite fields, even in the presence of a dishonest majority. MASCOT combines the online phase of the SPDZ with an OT-based preprocessing and is also UC-secure against a malicious majority. Classical OT is known to be constructed only using PKC. As a result, executing a high volume of OT operations can be computationally intensive and inefficient, particularly for SMC applications requiring hundreds or thousands of transfers. To address this issue, MASCOT uses an OT extension protocol [62] that takes a small number of OTs as a base and generates a large number of OTs using symmetric cryptographic techniques, such as Pseudo-Random Function (PRF). The concept of OT extension was first introduced by Beaver in 1996 [63]. Subsequent improvements were made by Ishai et al. [64], who optimized Beaver's original protocol. More recently, a study [62] proposed a straightforward consistency check that enhances the security of OT extension against malicious attacks, thereby ensuring maliciously secure OT extension. This extension is crucial for real-life applications because it significantly increases the efficiency.

While classical SMC approaches have been extensively researched, they suffer from security and efficiency challenges, primarily due to their reliance on PKC. Classical SMC implementations that are based on prime number factorization or discrete logarithms are not secure in the presence of quantum computers due to the Shor's algorithm [18]. Moreover, classical SMC approaches heavily rely on PKC, leading to substantial computational and communication costs [22]. To tackle the limitations of classical SMC, two main approaches are under research: post-quantum and quantum cryptography. Post-quantum cryptography [23] refers to cryptographic algorithms specifically designed to remain secure against the potential threats posed by quantum computers. These algorithms are built upon mathematical problems that are presumed to be computationally challenging for both classical and quantum computers to solve efficiently. Prominent areas within post-quantum cryptography include lattice-based cryptography [24], which leverages the computational difficulty of problems such

as Shortest Vector Problem (SVP) [65], and code-based cryptography [25], which relies on the complexity of decoding random linear codes. Additionally, hash-based cryptography [26] employs hash functions that are thought to be resilient to quantum attacks. While post-quantum cryptography aims to secure data against classic and quantum attacks, it remains to be proven if these protocols can indeed resist quantum computers [22]. On the other hand, quantum-based approaches, exploit the unique feature of quantum mechanics such as no-cloning, entanglement, and superposition to establish advanced cryptographic protocols. These quantum features enable new levels of security and privacy that classical cryptographic methods cannot achieve [28, 11]. Generally, quantum approaches can be classified into two categories: quantum communication and quantum computing-based approaches. Following the successful implementation of QKD, researchers have shifted their focus towards addressing the security and efficiency challenges of classical SMC through the utilization of quantum communication protocols. Within this approach, quantum communication technologies such as QRNG [66], QKD [36], and QOKD [22] are utilized to achieve higher performance. A significant property of these quantum technologies is that an eavesdropper attempting to measure a quantum state introduces detectable perturbations [36]. This quantum communication-based approach offers both enhanced security and practical implementability due to its higher TRL. For instance, in [27], QKD and QOKD technologies were integrated into classical MASCOT [8] protocol to provide a lane change service in vehicular networks. In [11], authors computed phylogenetic trees of SARS-CoV-2 genomes by integrating QRNG, QKD and QOKD with the Yao protocol.

After the advent of quantum computers, researchers have been exploring their possibility to revolutionize different domains, including cryptography and secure computation. Within the quantum computing approach, the unique features of quantum mechanics, such as superposition, entanglement, and quantum measurements are deployed to compute the desired function without revealing any information about parties' private inputs. Various quantum resources such as entangled particles [37, 38, 39, 67] and single qubits [31, 40, 41, 33] are exploited to implement SMC protocols to compute various types of functions, such as Boolean functions [31, 68], polynomials [32], and arithmetic operations [69, 70, 34, 71]. For example, in [72], Raussendorf and Briegel proposed a new approach called Measurement-Based Quantum Computing (MBQC) in which entangled particles are used to perform SMC. In this quantum computing model, computations are accomplished through a sequence of measurements performed on a highly entangled quantum state, referred to as a cluster state or a resource state. This type of quantum computing is different from the traditional circuit model that involves the manipulation of qubits using quantum gates, similar to how classical computers employ logic gates for bit processing. The MBQC provides a number of advantages as it inherits certain characteristics that make it more fault-tolerant compared to the circuit model. For instance, if the quantum state prepared in the initial step is too imprecise, we can simply discard this state before the computation is carried out and re-prepare it to make sure the output of the computation is accurate [73]. There are two MBQC schemes [74]: One-Way Quantum Computer (1WQC), also referred to as the cluster state model [75], and Teleportation Quantum Computation (TQC) [76]. The 1WQC method utilizes one-qubit measurements on a highly entangled state while the teleportation-based model requires joint (entangled) measurements [37]. It has been demonstrated that the 1WQC offers promising security features, as it leverages Blind Quantum Computation techniques [77]. In [42], authors demonstrate that any quantum algorithm can be implemented using qubit measurements on a cluster state. Furthermore, the authors discuss how this approach can be used to implement

various quantum algorithms, including Shor's and Grover's algorithms. In addition to entangled particles, single qubits can also be used to perform SMC. For instance, in [40], authors computed the secure Manhattan distance between two points by performing a phase-shift operation on a sequence of single qubits. Also, in [41], a protocol for Privacy Set Intersection Cardinality (PSIC) is proposed, in which a sequence of n qubits is used to compute the intersections between parties' private sets without revealing any details about the content of their respective sets. While employing single qubits reduces the quantum resource requirements, it does lead to higher communication costs compared to methods that rely on entangled states.

While the quantum computing approach offers promising results, the availability of quantum computers to individual users is currently impractical due to cost, accessibility, and technical limitations. As a potential solution, quantum computing may be deployed in a cloud computing paradigm, where users can access quantum computation resources via the cloud without directly owning or operating quantum hardware.

2.3 Applications of SMC

One of the first practical applications of SMC occurred during the conduct of an electronic double auction within the Danish Sugar Beet Auction in January 2008 [78]. Afterward, there has been a growing interest in applying SMC to real-world applications, such as vehicular networks [79, 80, 9, 27, 81], healthcare [82, 83, 11, 13, 29], machine learning [12, 13, 84, 85], finance [17, 16, 86], blockchain [87, 88, 89], among others. In the following, we outline some of the main applications of SMC.

2.3.1 Vehicular Networks

As cities get smarter and autonomous driving becomes a reality, vehicular communications and Internet of Vehicles (IoV) are becoming increasingly important. However, the widespread exchange of sensitive information within the network brings significant privacy and security concerns. For example, an adversary with access to the location information of vehicles can track the movements of specific individuals over time. By continuously monitoring the location data transmitted by vehicles, the adversary can infer sensitive information such as the daily routines, habits, and frequently visited locations of targeted individuals. This information can then be exploited for malicious purposes, such as physical surveillance, burglary, harassment, or other criminal activities. Traditional approaches often rely on trusted centralized entities to secure these networks, which may not be practical in dynamic vehicular environments. SMC holds immense potential for vehicular networks, enabling vehicles to share critical data for improved traffic flow, accident prevention, and overall network efficiency, all while safeguarding individual privacy. In the following, we explain how SMC can be deployed in vehicular network applications.

- *Safe Route Departure [27]:* SMC can ensure safe route departure by leveraging private information such as speed, lane availability, and proximity of neighboring vehicles while safeguarding sensitive data. By aggregating this sensitive data, SMC can suggest an appropriate departure time for vehicles intending to exit the route, thereby enhancing overall network safety.

- *Collision Warning [10]*: If an accident occurs in a network, SMC can promptly detect and warn vehicles about this critical situation. To achieve this, SMC constantly collects

private data such as the location and speeds of vehicles and informs them about the accident location and their distance from the accident site. This allows vehicles to adjust their route to avoid traffic and to drive more safely, preventing chain accidents.

- *Traffic Analysis [90]*: SMC enables multiple parties, such as transportation authorities, traffic management centers, and insurance companies, to collaboratively analyze traffic data without revealing the identities or movements of individual vehicles. This allows for efficient traffic management, congestion detection, and route planning while protecting the privacy of vehicles.

- *Location-Based Services [91]*: SMC can be utilized for location-based services, such as finding the nearest gas station or the closest vegetarian restaurant, while ensuring privacy. In this system, a vehicle provides its location to the SMC service, which then determines the nearest point of interest and the best route without compromising the privacy of the vehicle's location.

2.3.2 Health Care

The development of electronic wearable devices (smartwatches, fitness trackers, etc.), and telemedicine platforms has led to a huge collection of sensitive health information. While sharing this data holds immense potential for improving medical outcomes, it also raises concerns about data misuse. For example, consider a scenario in which an attacker could gain access to a healthcare provider's database containing patient medical records. With this information, the attacker could target individuals with specific medical conditions, such as diabetes, and use their medical history to sell fake or ineffective medical products. Now consider a more serious scenario where a malicious entity gains access to databases containing genomic data of individuals from a specific race or ethnic group. Using this information, the entity could develop a biological weapon causing widespread illness or even fatalities within the targeted population. Such an attack could be carried out for political purposes, as a means of ethnic cleansing or genocide. SMC emerges as a transformative approach to address these concerns by enabling collaborative data analysis and computation. By leveraging cryptographic techniques, healthcare stakeholders can collaborate on disease surveillance, clinical research, and personalized medicine without compromising patient privacy. In the following, we provide examples of how SMC can be utilized in the healthcare sector.

- *Drug Discovery [82]:* Drug discovery is a resource-intensive process that involves various stages such as target identification (proteins, enzymes, etc.), lead discovery, preclinical and clinical testing, and more. To reduce costs and accelerate the process, pharmaceutical companies are increasingly interested in conducting collaborative computations on their datasets. However, these datasets are highly confidential and considered valuable assets. SMC enables pharmaceutical companies to perform desired computations while preserving the privacy of their datasets.

- *Genomic Data Analysis [29]*: Genomic data contains highly sensitive information that must be protected to maintain patients' privacy. SMC techniques enable the secure analysis of genomic data from multiple sources, allowing researchers to investigate genetic variations, detect disease markers, and design personalized treatment plans while maintaining patient privacy.

- *Clinical Data Analysis [92]*: SMC allows multiple healthcare institutions to collaborate on analyzing clinical data without sharing patients' information. For example, an orthopedic clinic, a cardiology center, and a neurology hospital can securely combine their datasets using SMC to analyze common risk factors for certain medical conditions, such as cardiovascular diseases.

- *Disease Surveillance and Epidemiology [11]*: SMC enables secure aggregation of healthcare data for disease surveillance and epidemiological studies. When an epidemic emerges, public health officials can oversee disease outbreaks, trace the transmission of infectious diseases, and deploy prompt interventions to safeguard public health.

2.3.3 Finance

In the realm of finance, where data security and privacy are important, the utilization of SMC is essential to safeguard sensitive information and maintain trust among stakeholders. The advent of digital transactions, online banking, and complex financial instruments has significantly simplified financial processes, making them more efficient and accessible than ever before. However, this rapid digitization has also heightened concerns about data breaches, fraud, and privacy violations. The following are some examples of how SMC can be deployed for financial tasks such as fraud detection and risk assessment.

- *Fraud Detection [16]*: Banks and financial institutions can utilize SMC to detect fraudulent activities from multiple sources without sharing sensitive customer information. For instance, they can identify patterns such as sudden large transactions, transactions from multiple locations within a short time frame, or purchases inconsistent with a customer's typical spending habits.

- *Data Sharing [93]*: SMC facilitates secure data sharing among banks, financial institutions, and other stakeholders. It enables them to collectively analyze market trends, customer behavior, and other relevant data without compromising sensitive information.

- *Benchmarking and Analysis [94]*: Financial institutions can use SMC to compare their performance and financial metrics with other competitors without revealing proprietary information.

2.3.4 Machine Learning

As the volume and sensitivity of data increase, concerns about privacy and data ownership also increase. Traditional approaches often involve centralized data repositories or the sharing of raw data among multiple parties, raising significant privacy risks. SMC allows multiple parties to collaboratively train machine learning models on their corresponding private datasets while ensuring that individual data remains inaccessible to others. Here are some examples of how SMC is utilized for machine learning.

- *Federated Learning [16]*: SMC can support the deployment of FL protocols, enabling multiple participants to jointly train a machine learning model without revealing their raw data. In this approach, each participant computes model updates locally using their private datasets, and only the encrypted updates are sent to a central server for aggregation. This decentralized approach assures that data privacy is preserved throughout the training process.

- *Feature Aggregation*: In this setup, multiple parties possess unique sets of features and aim to conduct computations on the combined feature sets without exchanging data. For instance, various healthcare providers might each hold partial records of a patient's medical history but seek to utilize the complete history to improve predictive accuracy while ensuring patient privacy remains intact.

- *Dataset Augmentation [95]*: In this case, multiple parties each possess a small number of samples but aim to combine all examples to enhance the statistical strength of a model. For example, several hospitals may each have datasets containing a limited number of patient X-rays or MRI scans. These institutions seek to jointly analyze all available images to increase the precision and reliability of diagnostic models or medical research efforts.

2.4 Concepts and Framework

In this section, we explain the main concepts that are commonly used in SMC protocols. We start by explaining the types of adversaries and the various attacks that can occur during the computation process. We also discuss the threshold setting for party corruption, the distinction between information-theoretic and computational security, and how the complexity of SMC protocols is evaluated in terms of communication, computation, and resource requirements. We also describe the different types of circuits used in SMC and provide an overview of various SMC libraries.

2.4.1 Types of Adversaries

Understanding the potential threats and adversaries that cryptographic systems may encounter is fundamental for designing effective security measures. Cryptographic protocols should be robust enough to withstand various types of adversaries with differing levels of resources, capabilities, and motivations. These adversaries can range from passive eavesdroppers seeking to obtain sensitive information without trying to modify the protocol, to sophisticated attackers with advanced computational resources aiming to break the protocol and even generating false outputs. Common categories include semi-honest (passive) and malicious (active) adversaries. Semi-honest models offer simplicity and efficiency but provide weaker security guarantees, while malicious models offer stronger security guarantees but are more complex and resource-intensive. The two common adversarial types can be summarized as follows:

Semi-Honest (Passive Adversary)

In this model, adversaries aim to gain information about the computation without directly interfering with the protocol execution. These adversaries typically eavesdrop on the communication between parties and try to gain sensitive data from the exchanged messages. While passive adversaries do not actively manipulate the protocol, they pose a significant threat to the confidentiality and integrity of the computation by exploiting vulnerabilities in the communication channels. This approach achieves a balance between security and efficiency, making it well-suited for numerous real-world applications.

Malicious (Active Adversary)

In this model, the adversary actively manipulates the protocol execution to achieve their goals. Unlike passive adversaries, active adversaries may inject malicious messages into the communication, tamper with the protocol parameters, or disrupt the computation process to alter the results or extract sensitive information. For example, an active adversary may modify the computation inputs or manipulate the intermediate computations to bias the outcome in their favor. Active adversaries pose more serious threats to the security and correctness of the computation, requiring robust defense mechanisms to detect and mitigate their malicious actions.

Adversaries can also be categorized based on their computational capabilities. For instance, some adversaries may have access to limited computational resources and may only be able to execute basic attacks, while others, such as quantum adversaries, may possess advanced computational capabilities enabled by quantum technology.

2.4.2 Type of Attacks

Similar to any cryptographic protocol, SMC protocols are vulnerable to a range of attacks that can compromise their security. By analyzing these attacks, cryptographers can develop effective security mechanisms to mitigate these threats and ensure the resilience of SMC protocols in real-world scenarios. The following outlines several types of attacks that an SMC protocol may encounter:

- *Malicious Participant Attack*: In this attack, one or more participants deviate from the protocol execution to gain unauthorized information about other participants' inputs or compromise the correctness of the computation.

- *Collusion Attack*: In this attack, multiple participants collaborate to undermine the privacy of others. Unlike a malicious participant attack, where a single participant acts maliciously, a collusion attack involves collusion among multiple participants to achieve a common malicious goal.

- *Sybil Attack*: This attack involves creating multiple fake identities (Sybils) to gain control over the majority of the participants in the protocol, allowing the adversary to manipulate the computation's outcome.

- *Side-Channel Attack*: These attacks exploit unintended channels, such as timing or power consumption, to extract sensitive information about the computation or participants' inputs.

- *Denial-of-Service (DoS) Attack*: This attack aims to disrupt the normal operation of the SMC protocol by overwhelming the system with a high volume of requests or by exploiting vulnerabilities to cause crashes or downtime.

- *Information Leakage Attack*: These attacks attempt to infer sensitive information about participants' inputs or the computation's outcome by analyzing the communication or execution patterns of the protocol.

- *Corruption Attack*: In this attack, an adversary corrupts the inputs or outputs of the computation, leading to incorrect results or compromising the integrity of the computation.

- *Man-in-the-Middle (MitM) Attack*: During a MitM attack, a malicious adversary intercepts and manipulates the communication between parties, potentially modifying inputs or outcomes without being detected.

2.4.3 Threshold Settings

The threshold of SMC protocol represented by th is the maximum number of parties that can be corrupted by malicious adversaries without jeopardizing the integrity of the computation. The threshold of a protocol is typically defined by the cryptographic researcher who developed the secure SMC protocol. Higher thresholds may provide stronger security guarantees but can also result in increased computational and communication costs. Among the most common threshold settings are honest majority and dishonest majority. In an honest majority setting, the threshold is set such that a majority of parties must behave honestly for the protocol to succeed securely. Conversely, in a dishonest majority setting, the threshold allows for a majority of parties to be corrupted by adversaries while still preserving the security of the computation. The selection of threshold settings in SMC protocols depends on factors such as the level of trust among participants, the nature of the computation, and the desired security guarantees. By carefully selecting the appropriate threshold setting, SMC protocols can effectively balance security and efficiency to meet the requirements of diverse applications. The following outlines the main threshold settings for SMC protocols:

- *Honest Majority*: In this model, the adversary is presumed to corrupt a maximum of $th < n/2$ parties. This allows the protocol to tolerate a certain number of malicious or compromised participants without compromising security.

- *Two-thirds Honest Majority*: In this model, it is considered that the adversary can corrupt no more than $th < n/3$ parties. In this case, the group of honest parties always constitutes at least two-thirds of the total number of parties.

- *Dishonest Majority*: In this scenario, there is no predefined restriction on th, allowing it to attain its highest possible value, $th = n - 1$. This means the adversary can corrupt every party except one. Despite this, a protocol operating under such conditions would still ensure the privacy of the uncorrupted parties. This scenario represents the most stringent security setting. Essentially, each participant can be assured that their inputs remain confidential, even if all other parties conspire against them.

2.4.4 Security Guarantees

Information-theoretic security and computational security are two distinct models for evaluating the security of cryptographic protocols. In the domain of information-theoretic security, the emphasis is on establishing provable security guarantees that remain valid irrespective of an adversary's computational capabilities. Computational security depends on the assumption that specific computational problems are difficult to solve efficiently. The two common types of security guarantees can be classified as follows:

Information-Theoretic Security

Information-theoretic security provides assurances based on mathematical principles, such as entropy and Shannon's theory of communication. In essence, information-theoretic security aims to ensure that an adversary gains no information about the encrypted data beyond what is already available through the ciphertext. This approach offers strong security guarantees in scenarios where computational assumptions may not hold or where the adversary's computational capabilities are not well-defined. However, achieving information-theoretic security often comes with trade-offs in terms of efficiency and practicality, as cryptographic protocols designed to achieve such security may be more complex and resource-intensive. Information-theoretic security refers to the highest level of security achievable in cryptography, where the security of a cryptographic system is based on principles from information theory rather than computational assumptions. In information-theoretic security, an adversary is assumed to have unlimited computational power but is still unable to gain any details about the plaintext from the ciphertext. This implies that even if the adversary possesses unlimited computational power, they cannot compromise the system's security. One example of an information-theoretically secure scheme is the One-Time-Pad (OTP), where the key is as long as the plaintext and completely random.

Computational Security

Computational security relies on computational assumptions about the hardness of certain mathematical problems. For example, the security of RSA encryption is based on the assumption that factoring large integers is computationally difficult. In computational security, the security of a system may be compromised if the underlying computational assumptions are proven false or if there are breakthroughs in algorithmic or computational techniques that render the problem easy to solve.

2.4.5 Efficiency and Complexity

The efficiency of SMC protocols is crucial for practical applications. Several factors contribute to the computational and communication costs of SMC, including the complexity of the function being computed, the number of parties involved, and the type of adversary considered. When referring to the protocol's efficiency, the following terms need to be considered:

- *Computation complexity*: refers to the amount of computation required to execute a protocol. Fully Homomorphic Encryption (HE) protocols tend to be computationally complex. Measuring computational complexity involves analyzing the algorithm's behavior in terms of its run time, memory usage, etc. both theoretically and empirically.

- *Communication complexity*: Denotes the amount of communication exchanged between participants during the execution of the protocol, usually measured in terms of the number of bits or messages transmitted. GCs have large communication complexity. Here are some common approaches to measuring communication complexity:

  - *Message Count:* One straightforward measure of communication complexity is the overall number of messages exchanged between parties during the operation of the protocol. Counting the number of messages provides a basic assessment of communication overhead.

  - *Message Size:* Besides the number of messages, the size of each message exchanged between parties is also important. Measuring the size of messages helps quantify the amount of information transmitted in each communication round.
  - *Bandwidth:* Bandwidth refers to the rate at which data can be transmitted over a communication channel. Measuring communication bandwidth involves assessing the rate at which messages are exchanged between parties, which can impact the overall efficiency of the protocol.
  - *Bit Complexity:* Bit complexity measures the total number of bits transmitted between parties during the operation of the protocol. It considers the number of messages exchanged and the size of each message.

- *Round complexity*: refers to the number of communication rounds needed for the protocol to complete. To measure the round complexity of a protocol, we start by analyzing the protocol design to understand the sequence of message exchanges between parties.

- *Required Resources*: refers to the required resources to execute a protocol. Some examples of quantum resources are single qubits, cobite, and entangled particles.

Balancing these complexities is essential to ensure that protocols fulfill their intended objectives and remain practical.

2.4.6 Types of Circuit

In the context of SMC, a circuit refers to the logical representation of a computational task or the function being computed. The two primary types of circuits employed in SMC are classical and quantum circuits. Within classical circuits, two of the most common types are Boolean and arithmetic circuits. Additionally, with the advent of quantum computers, quantum circuits presented a promising approach for performing computations.

Each type of circuit has its unique strengths and limitations. Boolean circuits excel in handling logical operations and Boolean expressions, making them suitable for tasks involving conditional branching and logical comparisons. Arithmetic circuits are more adept at handling arithmetic operations and numeric computations, making them ideal for tasks involving mathematical operations and financial calculations. Quantum circuits, while still in their infancy, offer the potential for exponentially faster computations in certain quantum algorithms, such as Grover's algorithm for searching and Shor's algorithm for integer factorization. When determining the type of circuit to employ, careful consideration of the intended computation becomes necessary. For instance, when comparing binary values or evaluating logical conditions, opting for a Boolean circuit is typically more efficient. Conversely, when the computation involves arithmetic operations or numeric calculations, utilizing an arithmetic circuit is often the preferred choice. However, in scenarios involving quantum phenomena such as entangled qubits, the deployment of quantum circuits becomes essential for accurately modeling and processing quantum data.

Boolean Circuits

Boolean circuits receive Boolean values (true and false) as inputs and consist of logical gates such as AND, OR, and NOT gates. Each gate in the circuit performs a specific operation on its input values and produces an output, which serves as the input to subsequent gates

in the circuit. Boolean circuits are suitable for tasks where the computation revolves around logical operations and Boolean expressions. They are fundamental in protocols like Yao GC.

To generate Boolean circuits, various frameworks such as Fairplay [96] and Circuit-Based Model Checking - Garbled Circuits (CBMC-GC) [97] are deployed. The Fairplay platform was the first solution to address the challenge of circuit generation. It consists of a compiler that allows users to write programs in a straightforward, high-level language. This compiler then translates these programs into a Boolean circuit representation. CBMC-GC, on the other hand, generates Boolean circuits in the GC format. In the GC model, gates are connected through wires, which carry the output of one gate to the input of another. The format of the GC typically includes the following components:

- *Input Wires:* Input wires correspond to the inputs of the circuit, with each wire representing a single bit of data provided by the parties involved in the computation. For each input wire, there are typically two garbled values associated with it, one for each possible input value 0 or 1.

- *Intermediate Wires:* Intermediate wires carry the output of one gate to the input of another. These wires propagate the computed values through the circuit as the computation progresses.

- *Output Wires:* Output wires represent the final output of the circuit. The values on these wires correspond to the result of the computation, which is typically revealed only to the parties authorized to receive the output.

- *Garbled Tables:* Garbled tables contain encrypted representations of the truth table entries for each gate in the circuit. These tables are constructed such that each party can use their input to determine the correct output value for each gate without learning the other party's input.

Arithmetic Circuits

Arithmetic circuits receive numerical data as inputs and perform on arithmetic gates including addition, subtraction, multiplication, and division. Examples of arithmetic functions computed using arithmetic circuits include the addition of secret-shared integers, multiplication, and comparison. Various techniques are employed to compute different types of gates in arithmetic circuits. For example to compute the addition of two inputs, additive SS could be used. To obtain the multiplication of two input values, Beavers' multiplication triples are employed. Beavers' trick allows parties to securely compute the product of their shares using precomputed triples of secret-shared values. Arithmetic circuits can be viewed as a broader perspective of Boolean circuits, where Boolean circuits correspond to arithmetic circuits over the field $\mathbb{F}_2$. Arithmetic circuits are fundamental in protocols like Shamir's SS for secure computation of arithmetic functions.

Arithmetic circuits can be generated using libraries such as MP-SPDZ [98]. The process typically begins with the formulation of the computational task or function to be computed securely. Afterward, the library employs specialized techniques and algorithms to construct the circuit structure, including gate arrangement, connectivity, and optimization to minimize resource utilization and computational complexity. This construction process may involve circuit decomposition, gate-level optimization, and protocol-specific optimizations to ensure efficiency and scalability. Once the circuit is generated, MP-SPDZ provides functionalities for

simulating, executing, and verifying the computation securely across multiple parties while preserving the privacy and integrity of their inputs.

Quantum Circuit

Quantum circuits are a specialized type of circuit designed to execute quantum algorithms and computations. Unlike classical circuits, quantum circuits incorporate quantum gates such as Hadamard, CNOT, and Phase gates, allowing for operations on quantum bits or qubits. Quantum circuits are utilized in quantum computing applications, offering potential advantages in certain types of computations, such as factorization and search algorithms.

Quantum circuits are typically generated using quantum frameworks such as IBM QisKit [99] and Google Cirq [100]. In IBM QisKit, quantum circuits are generated using a combination of programming and graphical tools provided by the QisKit framework. Here is how quantum circuits are generated within QisKit:

- *Quantum Circuit Objects:* In QisKit, quantum circuits are represented as objects in Python code. Users can create a new quantum circuit object by instantiating the QuantumCircuit class provided by QisKit. This object serves as a container for quantum gates and operations.

- *Gate Operations:* QisKit provides a wide range of quantum gates and operations that can be applied to quantum circuits. These gates include fundamental single-qubit gates such as the Pauli-X gate (X), Pauli-Y gate (Y), and Pauli-Z gate (Z), as well as multi-qubit gates such as the CNOT gate (CX) and controlled-phase gate (CP). Users can add these gates to their quantum circuits to perform various quantum operations.

- *Quantum Registers and Classical Registers:* Quantum circuits in QisKit consist of quantum registers and optional classical registers. Quantum registers represent the qubits in the circuit, while classical registers can be used to store measurement outcomes or classical information derived from quantum computations.

- *Circuit Visualization:* QisKit provides tools for visualizing quantum circuits, allowing users to inspect and analyze the structure and operations of their circuits. The draw() method can be used to generate a visual representation of a quantum circuit, which can be displayed in Jupyter notebooks or saved as an image file.

QisKit provides a rich set of programming interfaces and functions for creating and manipulating quantum circuits. This allows users to build complex quantum algorithms and protocols by combining basic quantum operations.

2.4.7 Output Guarantees

In the context of SMC, different approaches can be employed to control the distribution of computation outputs to specific parties, offering various types of output guarantees:

- *Public Output*: In this setting, all parties receive the identical output of the computation.

- *Private Output*: In this setting, more than one output could be generated such that each party P_i obtains a different output O_i. We can tailor the protocol to generate the private outputs as follows: each party can introduce an additional input consisting of

a secret key known only to them. By adapting the computation function, all outputs can be made accessible to every party, except that output O_i is encrypted using the key supplied by party P_i. Consequently, solely this party possesses the means to decrypt and access its respective output.

These output guarantees offer flexibility in customizing how computation results are shared, striking a balance between transparency and privacy in different SMC scenarios.

2.4.8 SMC Libraries

Various frameworks exist for implementing SMC protocols, such as MP-SPDZ [98], Library for Secure Computation API (LibSCAPI) [101], SCALE-MAMBA [102], ABY [103], and FRESCO [104], among others. These libraries offer a wide range of functionalities, including cryptographic primitives, protocol implementations, and optimization techniques tailored for SMC applications. For instance, LibSCAPI [101] implements variety of SMC protocols including Yao and GMW [49]. It also provides support for a wide range of cryptographic primitives such as hash functions, MACs, pseudorandom and permutations functions, random oracle, etc. It also offers support for different types of circuits, including Boolean circuits and Arithmetic circuits. LibSCAPI offers support for various communication mechanisms, including socket-based communication, Message Passing Interface (MPI), and Secure Sockets Layer (SSL) protocols. These mechanisms ensure secure and efficient communication between parties, facilitating the execution of SMC protocols over distributed networks. LibSCAPI incorporates various performance optimization techniques to improve the efficiency of SMC protocols, including parallelization, batching, and optimized cryptographic implementations. These techniques help reduce computation overhead and communication latency, enabling faster and more scalable execution of secure computations. LibSCAPI is primarily implemented in C++, leveraging its low-level control over system resources. It also provides bindings for other programming languages such as Python and Java. LibSCAPI is developed by Bar Ilan University Cryptography Research Group.

Another framework for implementing SMC is MP-SPDZ [98] which supports a variety of SMC protocols, allowing developers to choose the most suitable protocol for their specific application requirements. Some of the implemented protocols include Yao GC [6], SPDZ [55, 56], BMR [52], MASCOT [8], and more. These protocols are implemented in different security models namely honest and dishonest majority, semi-honest and malicious adversaries, making it suitable for a variety of SMC scenarios. In addition, MP-SPDZ also provides support for various SMC primitives such as SS, HE, GCs and OT extension, along with a wide range of cryptographic primitives such as encryption schemes, commitment schemes, ZKP, and PRF. In MP-SPDZ, different types of circuits such as Boolean circuits, arithmetic circuits, GCs, and mixed circuits are being implemented. This platform mainly uses Python and C++ programming. MP-SPDZ is known for its efficiency and scalability, making it suitable for a wide range of practical applications.

2.5 Yao Garbled Circuit

The origins of SMC can be traced back to the pioneering work of Andrew Yao in the 1980s [6]. In his groundbreaking paper "Protocols for Secure Computations," Yao introduced a cryptographic protocol to address the Millionaires' Problem, a fundamental challenge in

cryptography where two parties aim to determine who is wealthier without disclosing their actual financial details to one another. The Yao protocol operates under the assumption that the function f can be represented using a Boolean circuit.

Let us perform the Yao protocol for a circuit with one gate. Suppose Alice and Bob want to evaluate the output of a single AND gate circuit using one input bit from Alice, $a \in \{0,1\}$, and one input bit from Bob, $b \in \{0,1\}$. First, Alice, referred to as garbler, generates six random keys: k_A^0 associated with her logical input value 0, k_A^1 associated with her logical input value 1, k_B^0 associated with Bob's input logical value 0, k_B^1 associated with Bob's input logical value 1, G_0^0 associated with the logical output value 0 of gate G_0, and G_0^1 associated with the logical output value 1 of gate G_0. Alice also generates three random permutation bits, p_A^0, p_B^0, and $p_{G_0}^0$, and defines $p_A^1 = p_A^0 \oplus 1$, $p_B^1 = p_B^0 \oplus 1$, and $p_{G_0}^1 = p_{G_0}^0 \oplus 1$. Alice computes the four string blocks which correspond to four possible inputs $(a,b) \in \{(0,0);(0,1);(1,0);(1,1)\}$ as

$$\begin{aligned}
\tilde{G}^{00} &= \mathrm{Enc}(k_A^0||k_B^0) \oplus ({G_0}^0||p_{G_0}^0) \\
\tilde{G}^{01} &= \mathrm{Enc}(k_A^0||k_B^1) \oplus ({G_0}^0||p_{G_0}^1) \\
\tilde{G}^{10} &= \mathrm{Enc}(k_A^1||k_B^0) \oplus ({G_0}^1||p_{G_0}^0) \\
\tilde{G}^{11} &= \mathrm{Enc}(k_A^1||k_B^1) \oplus ({G_0}^1||p_{G_0}^1),
\end{aligned} \tag{2.2}$$

where Enc could be SHA-1 or other encryption functions. The orders of the string blocks in $\tilde{G}^{ab}$ are computed using this formula: $2p_A^a + p_B^b$. As an example, if $p_A^0 = 0$, $p_B^0 = 1$ (therefore, $p_A^1 = 1$, $p_B^1 = 0$), the strings $\tilde{G}^{00}$, $\tilde{G}^{01}$, $\tilde{G}^{10}$, and $\tilde{G}^{11}$ will be at positions 1, 0, 3, and 2, respectively. In this case the GC is $\tilde{G} = \tilde{G}_0^{01}||\tilde{G}_0^{00}||\tilde{G}_0^{11}||\tilde{G}_0^{10}$. Equation 2.2 can be summarised as

$$\tilde{G}^{ab} = \mathrm{Enc}(k_A^a||k_B^b) \oplus ({G_0}^{g_0}||p_{G_0}^{g_0}), \tag{2.3}$$

where $g_0 = \mathrm{AND}(a,b)$. Note that instead of using the concatenation of strings we could XOR both keys. Since Bob is the evaluator, he is going to receive the circuit, i.e. $\tilde{G}$, from Alice and evaluate it. In order to do so, he needs two input keys with their corresponding permutation bits to decrypt the gate. As explained above, the permutation bits tell Bob which string block $\tilde{G}^{ab}$ from $\tilde{G}$ he has to decrypt. For instance, if he receives keys with permutation bits 11, he will decrypt the last string block of $\tilde{G}$. So, Alice sends her input key (e.g. $k_A^1||p_A^1 = k_A^1||1$) to Bob. Alice has to send to Bob, one of the corresponding keys: $k_B^0||p_B^0$ or $k_B^1||p_B^1$), based on Bob's input bit. If Bob's input is 0, he should receive $k_B^0||p_B^0$. If Bob's input is 1, he should receive $k_B^1||p_B^1$. But we don't want Alice to learn about Bob's input. Also, we do not want Bob to know both keys. To this end, we use Oblivious Transfer (OT) [105]. OT is a cryptographic protocol where a sender transmits one of multiple pieces of information to a receiver, while the sender remains unaware of which specific piece was chosen by the receiver. This ensures that the sender remains unaware of the receiver's choice, while the receiver only gains access to the specific piece they requested. With this information, Bob is able to evaluate the circuit, i.e. obtaining an output string $G_0^{g_0}$. To obtain $G_0^{g_0}$, he has to hash the key values and XOR it with the correct string block $\tilde{G}^{ab}$ from $\tilde{G}$

$$\mathrm{Enc}(k_A^a||k_B^b) \oplus {\tilde{G}_0}^{ab} = G_0^{g_0}||p_{G_0}^{g_0}. \tag{2.4}$$

For a single gate circuit the permutation bit $p_{G_0}^{g_0}$ is discarded, but if the output of the gate was to be used as input in another gate, the permutation bit, $p_{G_0}^{g_0}$, would be used to decrypt

the next gate. When the evaluator, Bob in this case, reaches the last gate, in a n gate circuit, that we are going to assume to correspond to gate G_{n-1}, he sends the last key, $G_{n-1}^{g_{n-1}}$, to Alice, and Alice tells him the final result, g_{n-1}. Indeed, the output of a circuit can be the result of more than one gate, and in this case, the evaluator will send all final keys to the garbler. Notice that as the permutation bits reveal no information, they can be transmitted openly.

Let us consider a specific example where we chose Enc = SHA-1. Assume a key length of 4 bits and the following keys generated by Alice: $k_A^0 = 0100$, $k_A^1 = 1000$, $k_B^0 = 0010$, $k_B^1 = 0011$, $G_0^0 = 1100$ and $G_0^1 = 0100$. Also, Alice sets randomly the permutation bits to be $p_A^0 = 0$, $p_B^0 = 1$, $p_{G_0}^0 = 1$ and, thus, $p_A^1 = 1$, $p_B^1 = 0$ and $p_{G_0}^1 = 0$. Note that the hash function SHA-1 generates a 160-bit output. So, for the sake of the example let us consider a truncation of its output to the last generated 5 bits. Alice then generates the four possible garbled strings: $\tilde{G}_0^{00} = \text{SHA-1}(k_A^0||k_B^0) \oplus (G_0^0||1) = 01111 \oplus 11001 = 10110$, $\tilde{G}_0^{01} = \text{SHA-1}(k_A^0||k_B^1) \oplus (G_0^0||1) = 00110 \oplus 11001 = 11111$, $\tilde{G}_0^{10} = \text{SHA-1}(k_A^1||k_B^0) \oplus (G_0^0||1) = 00001 \oplus 11001 = 11000$, $\tilde{G}_0^{11} = \text{SHA-1}(k_A^1||k_B^1) \oplus (G_0^1||0) = 10110 \oplus 01000 = 11110$. Then, Alice concatenates the four elements according with the order given by the corresponding permutation bits. So, she gets $\tilde{G} = \tilde{G}_0^{01}||\tilde{G}_0^{00}||\tilde{G}_0^{11}||\tilde{G}_0^{10} = 11111101101111011000$.

Bob receives $\tilde{G} = 11111101101111011000$ and Alice's input, say $k_A^1||1 = 10001$. Through OT, Bob also receives his input, say $k_B^0||1 = 00101$. The permutation bits (the fifth element in each string) tell Bob the correct string block for him to decode. In this case, Bob has to use the string in the binary position $11_2 = 3_{10}$ i.e. 11000, notice that the string's position goes from 0 to 3. In this case, Bob uses the last 5 bits of $\tilde{G}$ and proceeds as

$$\tilde{G}|_3 \oplus \text{SHA-1}(k_A^1||k_B^0) = 11000 \oplus \text{SHA-1}(10000010) = 11000 \oplus 00001 = 11001. \tag{2.5}$$

Therefore, Bob concludes that the output key is 1100 and its permutation bit is 1. In case there are no more gates to decrypt, Bob sends Alice the output key, 1100, and she gives back the result which, in this example, corresponds to the logical value false, i.e. 0.

2.6 Final Remark

In this review chapter, we explained the definition of SMC. We provided a literature review, discussing state-of-the-art for classical and quantum-based SMC. We delved into the practical applications of SMC in various domain such as vehicular networks, health care, finance, etc. While SMC holds promises for real-life applications, its widespread adoption is hindered by significant challenges. Firstly, security vulnerabilities arise with the advent of quantum computers, making classical SMC implementations susceptible to attacks. Secondly, efficiency remains a critical concern, particularly for large-scale applications where existing SMC protocols can be computationally expensive and inefficient. Although post-quantum cryptography aims to develop new cryptographic primitives resilient to quantum attacks, it remains to be proven that they can offer a solution that complies with the security and efficiency requirements of SMC. Therefore, quantum-communication and quantum computing-based approaches are being deployed to reach SMC. The quantum communication approach has been successfully implemented in laboratory settings and exhibits a high TRL, but challenges such as key generation rates must still be overcome for broader adoption. In contrast, quantum computing-based approaches are still in the early stages of development,

with lower TRL, and their success depends on advancements in quantum hardware, including quantum processors. Ongoing research continues to work on overcoming these challenges.

Chapter 3

Quantum SMC Framework

In this chapter, we propose a QSMC framework by leveraging quantum communication technologies. This framework offers promising avenues for implementing real-life applications of SMC across diverse domains.

In Section 3.1, We present the components of the proposed QSMC framework, which integrates three advanced quantum communication technologies—QRNG, QKD, and QOKD—in conjunction with a KMS and the MASCOT SMC protocol. In Section 3.2, we explain the implementation details to build the QSMC frameworks. Section 3.3, evaluates the proposed framework by providing security and efficiency analyses. Finally, Section 3.4 concludes the chapter.

3.1 QSMC framework

In this section, we propose a QSMC framework that enables multiple parties to securely collaborate on a desired computation while ensuring the privacy of each party's private inputs. The proposed framework comprises six components that work together to facilitate quantum-based SMC, as illustrated in Fig. 3.1. The first component generates random numbers using QRNG protocol. These numbers are truly random and can be generated through homodyne detection of vacuum fluctuation. The second component generates symmetric keys through the QKD protocol. These keys are later employed for encryption and decryption purposes and to extend the number of OTs using the AES protocol. The third component generates asymmetric bit sequences known as oblivious keys using QOKD protocol. These keys are required for the generation of Quantum Oblivious Transfer (QOT) in which two parties are able to transfer messages in an oblivious way. Within the QSMC framework, both a classical channel and a quantum channel are established between each pair of parties, facilitating secure quantum key generation. The fourth component is a KMS, responsible for securely storing and managing the generated keys. The role of the KMS is crucial because the QRNG, QKD, and QOKD protocols continuously produce cryptographic keys that need structured management throughout their life cycle. The KMS validates each key, distributes it to authorized users, and securely retires outdated keys, thus reducing the risks associated with compromised or expired keys. The fifth component is an SMC protocol called MASCOT, which is responsible for enabling secure computations among parties. Finally, the sixth component generates the corresponding arithmetic circuit for a desired use case. The use case can vary widely, from vehicular networks to collaborative data analysis in healthcare. With the arithmetic circuit

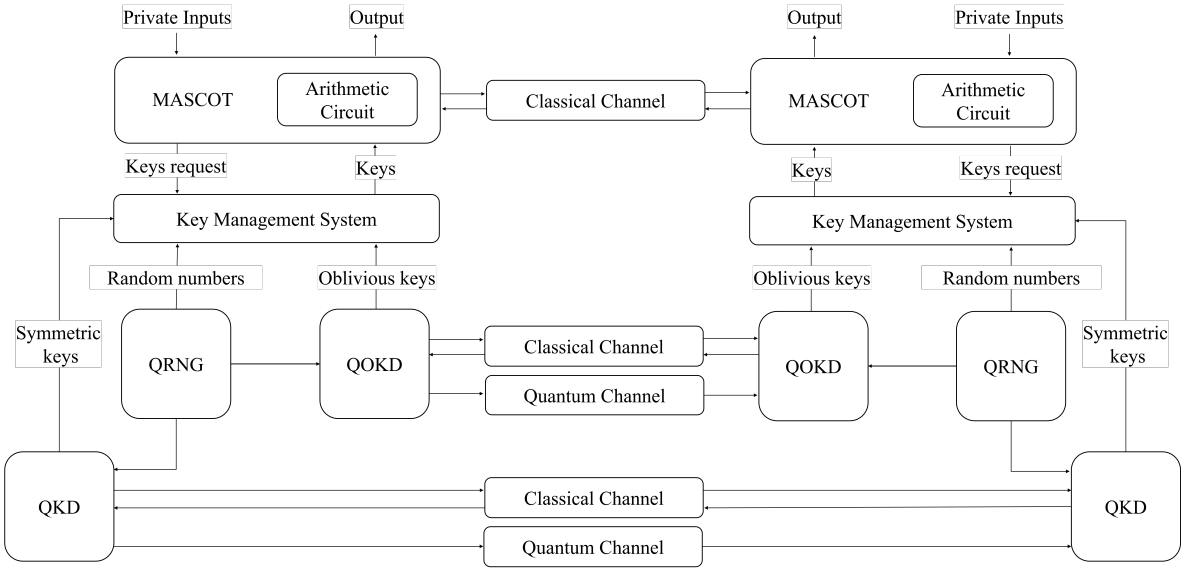


Figure 3.1: Overview of the proposed QSMC framework.

generated and the necessary cryptographic keys established, we can now feed the private inputs from each party into the framework. Afterward, the computation is performed in a way that ensures no party gains any knowledge about the other parties' inputs, except for the final output of the computation. In the remaining sections, we explain each component in more depth.

3.1.1 Quantum Random Number Generation

Random numbers are currently an essential resource in security-critical cryptographic applications. So far, classical entropy sources and Pseudorandom Number Generation (PRNG)s have been able to suppress this demand, but such methods yield inherently periodic sequences that become predictable to an adversary with access to enough computational power [106]. To overcome these limitations, hardware-based solutions have been developed to derive randomness from unpredictable physical processes, such as atmospheric or electrical noise [107]. However, these devices typically suffer from slow performance. Additionally, practical imperfections and the potential for external tampering by malicious adversaries can degrade the quality of their output, making them less reliable [108]. QRNG address these limitations by exploring the probabilistic nature of quantum measurements as their randomness source. QRNG provide information-theoretic security by leveraging the inherent randomness of quantum mechanics. True random numbers can be generated using various methods such as radioactive decay [109]. However, the majority leverage quantum optics properties such as photon arrival times [110] and vacuum noise fluctuation [1].

As shown in Fig. 3.2, a typical QRNG can generally be divided into two distinct stages, each with different functions: the physical layer and the post-processing layer. The physical layer includes the Physical Entropy Source, which generates raw quantum data, and digitization, which converts this data into a binary format, resulting in a potentially biased raw output. The post-processing layer consists of entropy estimation, which assesses the randomness of the raw data, and randomness extraction, where any bias is removed to produce a truly

random output. Additionally, random seeds are used to assist in the randomness extraction process.

In [1], a device-dependent continuous variable QRNG protocol based on homodyne measurements of vacuum fluctuations was implemented. According to quantum mechanics, even empty space, or a vacuum, is not truly empty. It is filled with temporary, random energy fluctuations due to the Heisenberg Uncertainty Principle, which results in the spontaneous creation and annihilation of virtual particle pairs. These fluctuations are inherently unpredictable, making them an excellent source of randomness. To generate random numbers, a highly stable laser, known as the Local Oscillator (LO), produces a coherent light beam with a known wavelength and phase. The LO is described by the annihilation operator $\hat{a}_{\mathrm{LO}}$ as

$$\hat{a}_{\mathrm{LO}} = \hat{\alpha}(t)e^{i(\omega_{\mathrm{LO}}t+\theta(t)}, \tag{3.1}$$

in which $\hat{\alpha}(t)$ is the photon flux, ω_{LO} represents the frequency, and $\theta(t)$ is the starting phase of the laser. The $\hat{a}_{\mathrm{LO}}$ satisfies

$$\hat{a}_{\mathrm{LO}} \left|\alpha\right\rangle = \hat{\alpha}(t) \left|\alpha\right\rangle, \tag{3.2}$$

where $|\alpha\rangle$ is the coherent state. This coherent light is directed towards the first input port of an imbalanced Beam Splitter (BS), while the second input port remains blocked, ensuring that only the vacuum state is present at that port. Two Variable Optical Attenuator (VOA) precisely adjusts the power level of the output signals. These VOAs are modeled as lossless BSs, meaning that they split the signal without introducing any additional loss beyond the intended attenuation. The outputs are subsequently measured using two detectors, and the signal is derived by calculating the difference between the two output measurements as

$$\hat{i}_H = q\left[\hat{d}^{\dagger}_{\mathrm{att}_1}(t)\hat{d}_{\mathrm{att}_1}(t) - \hat{d}^{\dagger}_{\mathrm{att}_2}(t)\hat{d}_{\mathrm{att}_2}(t)\right], \tag{3.3}$$

in which q is the charge of the electron, and $\hat{d}^{\dagger}_{\mathrm{att}_1,\mathrm{att}_2}$ correspond to the photon flux of each output beam. This signal is then amplified by a Transimpedance Amplifier (TIA), leading to the output voltage

$$\hat{V}_H(t) = G_{\mathrm{TIA}}[\hat{i}_e(t) + \hat{i}_H(t)] \otimes h(t), \tag{3.4}$$

where G_{TIA} is the transimpedance amplifier gain, i_e is the amplifier electronic noise, and $h(t)$ is the detector's impulse response function. The impulse response function $h(t)$ accounts for the effects of both the photodetectors' bandwidth and the TIA's time response. However, because the TIA typically has a lower bandwidth, its contribution is more significant. For simplicity, an ideal impulse response $h_p(t) = \delta(t)$ is used for both photodiodes. With these assumptions, the response in the frequency domain can be modeled as a Butterworth filter

$$|H(\omega)|^2 = \frac{1}{1 + \left(\frac{\omega}{2\pi\Delta f}\right)^{2n}}, \tag{3.5}$$

where Δf is the detector's bandwidth and n represent the filter-order. The resulting signal is digitized by an Analog-to-Digital Converter (ADC), producing a stream of digital values that represent the random fluctuations. The quantum signal is unavoidably mixed with classical noise components. This noise often originates during the digitization process, where electronic noise inherent to the physical devices overlaps with the acquired signal. In the following, we explain how classical noise, including phase noise, thermal and Relative Intensity Noise (RIN), affects the quantum signal.

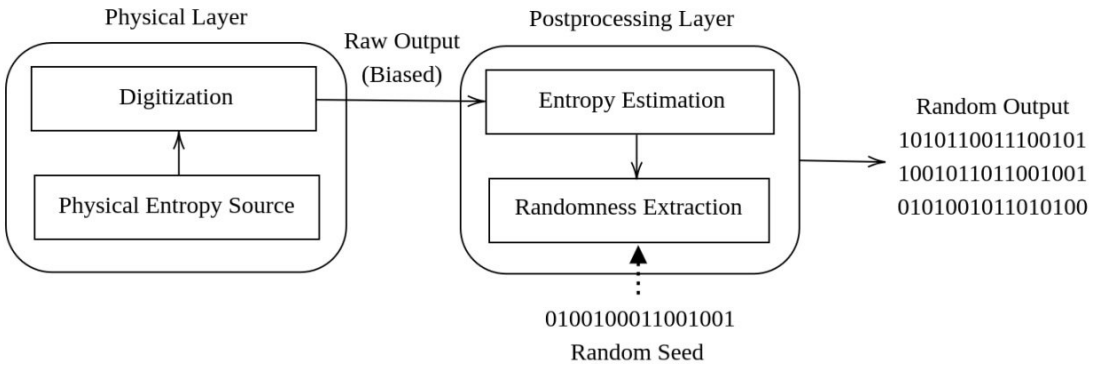


Figure 3.2: The two key stages of a QRNG.

As represented in Eq. (3.1), the starting phase of the laser changes over time, which leads to phase noise. Phase noise are random and small changes in the phase of a signal that occur over time. Additional noise is avoided provided there is no phase shift, $\Delta\phi$, between the beams exiting the beam splitter. In this analysis, the optical path lengths of both output arms are considered identical, ensuring $\Delta\phi = 0$, and thus its impact can be neglected. Under the assumption of a null average electronic current, $\langle i_e(t)\rangle$, and defining $\hat{\alpha}(t) = \sqrt{F + \Delta f_{\text{lo}}(t)}$, that F indicates the average photon flux, and $\Delta f_{\text{lo}}(t)$ shows the intensity variations, output is derived as

$$\langle V_H(t)\rangle = qG_{\text{TIA}}\gamma F \otimes h(t), \tag{3.6}$$

where

$$\gamma = \left(\frac{1}{2} + \Delta\right)\eta_1\eta_{\text{VOA}_1} - \left(\frac{1}{2} - \Delta\right)\eta_2\eta_{\text{VOA}_2}, \tag{3.7}$$

where η_1 and η_2 are VOA's attenuation factors; and η_{VOA_1} and η_{VOA_2} express the transmissivities of each VOA. Now we can compute the variance of the voltage which gives information about how much randomness comes from quantum noise (which is essential for QRNG) versus classical noise (which is unwanted and needs to be minimized). The variance of the voltage, $\sigma_H^2(t)$, can be obtained by evaluating the autocovariance function, $K(\tau)$, at $\tau = 0$. The autocovariance function can be derived using Eqs. (3.4) and (3.6) as

$$K(\tau) = G_{\text{TIA}}^2\left[\frac{2k_BT}{R} + q^2\beta F + q^2\text{RIN}\gamma^2F^2\right]\int_{-\infty}^{+\infty} d\tau' h(t-\tau')h(t-\tau'+\tau), \tag{3.8}$$

where

$$\beta = \left(\frac{1}{2} + \Delta\right)\eta_1\eta_{\text{VOA}_1} + \left(\frac{1}{2} - \Delta\right)\eta_2\eta_{\text{VOA}_2}. \tag{3.9}$$

To this end, it is assumed

$$\langle i_e(t_1)i_e(t_2)\rangle = \frac{2k_BT}{R}\delta(t_1 - t_2) \tag{3.10}$$

and

$$\langle \Delta f_{\text{lo}}(t_1)\Delta f_{\text{lo}}(t_2)\rangle = \text{RIN}\,\text{F}^2\delta(\text{t}_1 - \text{t}_2), \tag{3.11}$$

in which k_B represents the Boltzmann constant, T is the temperature, R is the load resistance of the TIA, and RIN denotes the mean relative intensity noise across the detector's bandwidth.

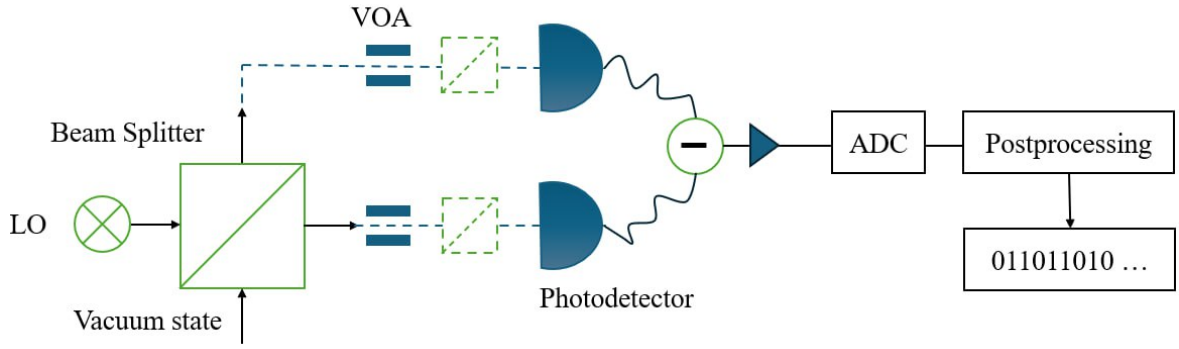


Figure 3.3: Laboratory setup for quantum random number generation [1].

This cross-correlation can be reduced to a simpler form using the Wiener-Khinchin theorem. Considering $t = \tau = 0$, the analytical expression for a Butterworth's filter of order $n = 3$ is

$$\sigma_H^2(t=0) = \frac{2\pi}{3} G_{\text{TIA}}^2 \Delta f \left[\frac{2k_B T}{R} + q^2 \beta F + q^2 \text{RIN} \gamma^2 F^2 \right] + \frac{\delta_q^2}{12}. \tag{3.12}$$

Here, the quantization error of an ADC with a bin width of δ_q is considered, resulting in a quantization error variance of $\frac{\delta_q^2}{12}$. Finally, Eqs. (3.6) and (3.12) can be rewritten in terms of the wavelength-dependent responsivity, $R(\lambda)$, and the optical power of LO, P_{LO}. Assuming that $\eta_3 = \eta_4$, we obtain

$$\langle V_H(t) \rangle = G_{\text{TIA}} \gamma', \tag{3.13}$$

$$\sigma_H^2(t=0) = \frac{2\pi}{3} G_{\text{TIA}}^2 \Delta f \left[\frac{2k_B T}{R} + q\beta' + q\text{RIN}\gamma'^2 \right] + \frac{\delta_q^2}{12}, \tag{3.14}$$

where

$$\beta' = R(\lambda) P_{\text{LO}} \left[\left(\frac{1}{2} + \Delta \right) \eta_{\text{VOA}_1} + \left(\frac{1}{2} - \Delta \right) \eta_{\text{VOA}_2} \right], \tag{3.15}$$

and

$$\gamma' = R(\lambda) P_{\text{LO}} \left[\left(\frac{1}{2} + \Delta \right) \eta_{\text{VOA}_1} - \left(\frac{1}{2} - \Delta \right) \eta_{\text{VOA}_2} \right]. \tag{3.16}$$

In order to remove the effect of classical noise and improve the quality of the generated random numbers, entropy estimation could be deployed as part of the post-processing stage. Entropy estimation techniques allow us to evaluate the true randomness of the generated bits by quantifying the amount of uncertainty and randomness in the output. By analyzing the entropy, it becomes possible to identify and filter out any non-random patterns introduced by classical noise, such as thermal or relative intensity noise.

The steps involved in the implementation of random number generation are depicted in Fig. 3.3, while Protocol 1 outlines the detailed procedure for QRNG.

Protocol 1 Quantum Protocol for Random Number Generation [1]

Output: strings of random numbers.

1. Prepare the coherent state representing the LO [111]:

$$|\alpha_{\mathrm{LO}}\rangle = e^{-\frac{|\alpha|^2}{2}} \sum_n \frac{\alpha^n}{\sqrt{n!}} |n\rangle = e^{(\alpha\hat{a}^\dagger - \alpha^*\hat{a})} |0\rangle , \tag{3.17}$$

   where α is a complex number and $|n\rangle$ is a Fock state with n photons, $\hat{a}^\dagger$ ($\hat{a}$) is the photon creation (annihilation) operator.

2. The coherent state is coupled to a vacuum state at an imbalanced BS, where $|\alpha_{\mathrm{LO}}\rangle$ inters through input port in_1, and $\hat{v}_s(t)$ enters through input port in_2. The input state of BS is

$$\Psi_{\mathrm{in}} = |\alpha_{\mathrm{LO}}\rangle_{\mathrm{in}_1} \otimes \hat{v}_s(t)_{\mathrm{in}_2} = e^{(\alpha\hat{a}^\dagger - \alpha^*\hat{a})} |0\rangle_{\mathrm{in}_1} |0\rangle_{\mathrm{in}_2}. \tag{3.18}$$

3. The output signals $\hat{a}_{\mathrm{out}_1}$ and $\hat{a}_{\mathrm{out}_2}$ can be described as

$$\hat{a}_{\mathrm{out}_1}(t) = i\sqrt{\frac{1}{2} + \Delta}\hat{a}_{\mathrm{LO}}(t) + \sqrt{\frac{1}{2} - \Delta}\hat{v}_s(t), \tag{3.19}$$

$$\hat{a}_{\mathrm{out}_2}(t) = \sqrt{\frac{1}{2} - \Delta}\hat{a}_{\mathrm{LO}}(t) + i\sqrt{\frac{1}{2} + \Delta}\hat{v}_s(t), \tag{3.20}$$

   where $\Delta = (|\mathcal{R}|^2 - 1/2)$ is the BS imbalance and $\mathcal{R}$ is reflection coefficient.

4. After passing through VOAs with attenuation factors η_1 and η_2 respectively, the attenuated signals are

$$\begin{aligned}\hat{d}_{\mathrm{att}_1}(t) = {} & i\sqrt{\left(\frac{1}{2} + \Delta\right)\eta_1\eta_{\mathrm{VOA}_1}}\hat{a}_{\mathrm{LO}}(t) + \sqrt{\left(\frac{1}{2} - \Delta\right)\eta_1\eta_{\mathrm{VOA}_1}}\hat{v}_s(t) \\ & + i\sqrt{\eta_1(1-\eta_{\mathrm{VOA}_1})}\hat{v}_{\mathrm{VOA}_1}(t) + i\sqrt{1-\eta_1}\hat{v}_1(t),\end{aligned} \tag{3.21}$$

$$\begin{aligned}\hat{d}_{\mathrm{att}_2}(t) = {} & i\sqrt{\left(\frac{1}{2} - \Delta\right)\eta_2\eta_{\mathrm{VOA}_2}}\hat{a}_{\mathrm{LO}}(t) + \sqrt{\left(\frac{1}{2} + \Delta\right)\eta_4\eta_{\mathrm{VOA}_2}}\hat{v}_s(t) \\ & + i\sqrt{\eta_2(1-\eta_{\mathrm{VOA}_2})}\hat{v}_{\mathrm{VOA}_2}(t) + i\sqrt{1-\eta_2}\hat{v}_2(t),\end{aligned} \tag{3.22}$$

   where $\hat{v}_{1,2}$ and $\hat{v}_{\mathrm{VOA}_1,\mathrm{VOA}_2}$ are the vacuum states of detectors and VOAs, respectively. $\eta_{1,2}$ represents the quantum efficiency of each detectors, and $\eta_{\mathrm{VOA}_1,\mathrm{VOA}_2}$ expresses the transmissivities of each VOA.

5. The resultant current and output voltage are

$$\hat{i}_H = q\left[\hat{d}^\dagger_{\mathrm{att}_1}(t)\hat{d}_{\mathrm{att}_1}(t) - \hat{d}^\dagger_{\mathrm{att}_2}(t)\hat{d}_{\mathrm{att}_2}(t)\right], \tag{3.23}$$

   and

$$\hat{V}_H(t) = G_{\mathrm{TIA}}[\hat{i}_e(t) + \hat{i}_H(t)] \otimes h(t). \tag{3.24}$$

6. Using an ADC, and by performing the post-processing stage, v_H is transformed into a stream of discrete digital values, which represent random numbers, such as 001010101...

3.1.2 Quantum Key Distribution

QKD is a cryptographic approach that securely exchanges symmetric keys between two parties, typically known by Alice and Bob names. The security of QKD arises from basic characteristics of quantum mechanics namely the no-cloning theorem, that mentions it would be impossible to make a copy from an unknown quantum state, and the fact that any measurement on a quantum system disturbs it, which means, if an eavesdropper tries to intercept the quantum key, their presence causes detectable disturbances, letting Alice and Bob to identify when the communication is compromised. Once the keys are securely exchanged, they can be employed to encrypt and decrypt messages through classical cryptographic algorithms, such as OTP or AES. QKD protocols can be implemented using three fundamental approaches: Discrete Variable (DV) [36], Continuous Variable (CV), and Entanglement-Based (EB) QKD. DV-QKD utilizes single photons or their quantum states to encode information. In DV-QKD, data is typically encoded in discrete quantum states of photons, such as polarization [2, 112, 113], phase [114], frequency [115], or time-bin [116]. Polarization encoding, for example, employs different polarization states to represent binary information, while phase encoding involves embedding data in the phase differences between interfering photon modes. Frequency encoding and time-bin encoding use distinct frequencies and time slots, respectively, to encode information. The main benefit of DV-QKD is its ability to provide high security, as the no-cloning theorem guarantees that any effort to intercept or eavesdrop will disturb the quantum state, thereby making any unauthorized access easily detectable. However, DV-QKD systems generally require Single-Photon Detector (SPD), which are expensive and slow, limiting their practicality in some scenarios [117]. Despite these challenges, DV-QKD has demonstrated its effectiveness over distances of up to 421 km [118]. CV-QKD, by contrast, encodes information in the continuous variables of quantum states, such as the amplitude and phase of coherent light [119, 120]. CV-QKD uses multi-photon states and relies on techniques like homodyne detection [121] to measure these continuous variables. This approach tends to be more cost-effective and faster compared to DV-QKD because it does not require SPDs. However, CV-QKD systems are susceptible to classical noise, which can impact their performance. Despite this, CV-QKD has been successfully implemented across distance ranges of around 100 km [122]. EB-QKD utilizes quantum entanglement to securely distribute cryptographic keys. In this approach, entangled photon pairs are generated, with a single photon is transmitted to Alice, while its counterpart is sent to Bob. Their quantum states remain correlated, so that measuring one instantly defines the other's state. This high-security method detects eavesdropping through disturbances in the entanglement. Unlike other QKD approaches, EB-QKD eliminates the need for trusted quantum sources, relying instead on the intrinsic properties of entanglement. Protocols like E91 [123] demonstrate its effectiveness, making EB-QKD especially promising for quantum networks and satellite-based applications. The selection of a QKD system typically depends on the specific requirements of the application, namely the balance between detection complexity, cost, and the desired range of secure communication.

The BB84 Protocol

In 1984, Bennett and Brassard introduced the first QKD protocol [36], revolutionizing cryptography with a groundbreaking approach to secure key distribution. This pioneering protocol, known as BB84, leveraged the concept of encoding information in the State Of

Polarization (SOP) of single photons. The polarization of a photon describes the direction in which its electric field vibrates as it travels. For example, if the electric field moves up and down, the photon is vertically polarized. If it moves side to side, the photon is horizontally polarized. Other directions, like diagonal or circular, are also possible. The protocol's security relies on the no-cloning theorem and the uncertainty principle, ensuring that any eavesdropper attempting to intercept the information will certainly be detected by Alice and Bob.

In the BB84 protocol, Alice encodes classical bits (0s and 1s) into photon polarization states using two randomly chosen bases: the rectangular basis ($|H\rangle$ for horizontal polarization and $|V\rangle$ for vertical polarization) and the diagonal basis ($|+\rangle$ for 45° polarization and $|-\rangle$ for 135° polarization). In this encoding scheme, $|H\rangle$ and $|+\rangle$ correspond to the 0 bit, while $|V\rangle$ and $|-\rangle$ correspond to the 1 bit. To transmit a 0, Alice randomly prepares a photon in either the $|H\rangle$ state (rectangular basis) or the $|+\rangle$ state (diagonal basis). Similarly, to transmit a 1, she sets up a photon in either the $|V\rangle$ state (rectangular basis) or the $|-\rangle$ state (diagonal basis). Alice sends this sequence of photons to Bob. For each photon he receives, Bob randomly decides whether to measure its polarization using either the rectilinear or diagonal basis. He records the results of these measurements, resulting in a mix of correct and incorrect basis measurements because he does not know which basis Alice used. If Bob measures a photon on the correct basis, he will obtain the correct bit. If he uses the incorrect basis, the result is random. Alice and Bob communicate over a public communication channel (which is presumed to be vulnerable to interception but not message tampering) to identify which photons were measured on the correct basis. Bob announces his basis choice for each photon. Alice then tells Bob which of his basic choices were correct. Bob discards the bits where he used the wrong basis. The remaining bits, where Bob used the correct basis, form a sifted key that is shared between Alice and Bob. To ensure there has been no eavesdropping, Alice and Bob check a subset of the sifted key bits. They publicly compare a small portion of their sifted keys. If too many discrepancies are found, they dump the entire key and initiate the process again. If the error rate is acceptably low, they move forward to the subsequent stage of the process. Alice and Bob apply privacy amplification techniques to reduce the knowledge an intruder could have potentially accessed, leading to a reduced yet significantly more secure final key. The following example showcases the BB84 protocol.

Alice's bits	0	1	1	0	1	1	0	0	1	0	1
Alice's bases	D	R	D	R	R	R	R	R	D	D	R
Photons Alice sends	↗	↕	↖	↔	↕	↕	↔	↔	↖	↗	↕
Bob's bases	R	D	D	R	R	D	D	R	D	R	D
Bob's bits	1		1		1	0	0	0		1	1
Bob announces his bases	R		D		R	D	D	R		R	D
Alice announces the correct basis			OK		OK			OK			
Shifted keys			1		1			0			
Bob discloses some keys at random					1						
Alice verifies them					OK						
Remaining shared secret keys			1					0			

Discrete Variable QKD

In this section, we give a more in-depth explanation of the DV-QKD system, as it has been adopted within the proposed QSMC framework. Our explanation builds on the work

presented in [2], where a polarization-encoded QKD system was designed and implemented using rectangular ($|H\rangle$ and $|V\rangle$) and diagonal ($|+\rangle$ and $|-\rangle$) bases. This system follows a prepare-and-measure setup, that consists of two sequential stages: first, quantum communication, and second, classical post-processing. The quantum communication phase is where qubits are exchanged between the parties, while the post-processing phase involves analyzing the exchanged data to ensure security and detect any potential eavesdropping attempts. In the quantum communication phase, a physical layer consisting of a transmitter, a communication channel, and a receiver is employed. The structure of the DV-QKD system is shown in Fig. 3.4.

DV-QKD Transmitter

In general, QKD Transmitter (QTx) involves two steps: single photon generation and information encoding. In the first step, Alice generates photons using a laser source. These photons are weak pulses of light usually generated at a low intensity to approximate single-photon states. Afterward, an Electronic Polarization Controller (EPC) is used to polarize the photons in order to encode the information in a quantum signal. Since Bob's detection system operates in Geiger mode—a highly sensitive operational mode designed to detect single photons—the signals produced by Alice must have a finite duration of time. This sensitivity requires precise timing of the signals Alice sends to ensure compatibility with Bob's detector and prevent overlap or interference. To do so, Alice employs an amplitude modulator, such as a Mach-Zehnder Modulator (MZM), to modulate the photon pulses. Before the MZM is used, the emitted light is guided through a polarization controller to align the beam with the MZM's main axis. The optical signal is then encoded by the MZM through amplitude modulation [124].

As illustrated in Fig. 3.4, in addition to the quantum signal, the QTx also employs a separate synchronization signal to ensure proper alignment between Alice and Bob. While the first MZM (MZM1) is responsible for encoding information onto the quantum signal by modulating its polarization, a second MZM (MZM2) modulates the synchronization signal. Both signals are pulsed and share the same repetition rate, ensuring synchronization. However, to prevent cross-talk, the two optical signals are transmitted at different wavelengths: 1547.72 nm for the quantum signal and 1510 nm for the classical synchronization signal. These signals are then merged into a single optical fiber through the use of a Wavelength-Division Multiplexing (WDM) combiner. These signals travel via a single-mode optical fiber to the receiver, where a WDM splitter separates them.

Photon Generation

Pure single-photon sources are challenging to achieve experimentally. Consequently, practical implementations often rely on faint laser pulses, where the emitted photons follow Poisson statistics. This means there is a small, but non-zero, likelihood of more than one photon occurrence in a single pulse. A straightforward method to approximate single-photon Fock states involves generating coherent states characterized by an extremely low average photon number, denoted by μ. By doing so, the number of photons per pulse follows a Poisson distribution. Mathematically, The likelihood of observing n photons in a pulse obeys Poisson statistics and could be formulated as

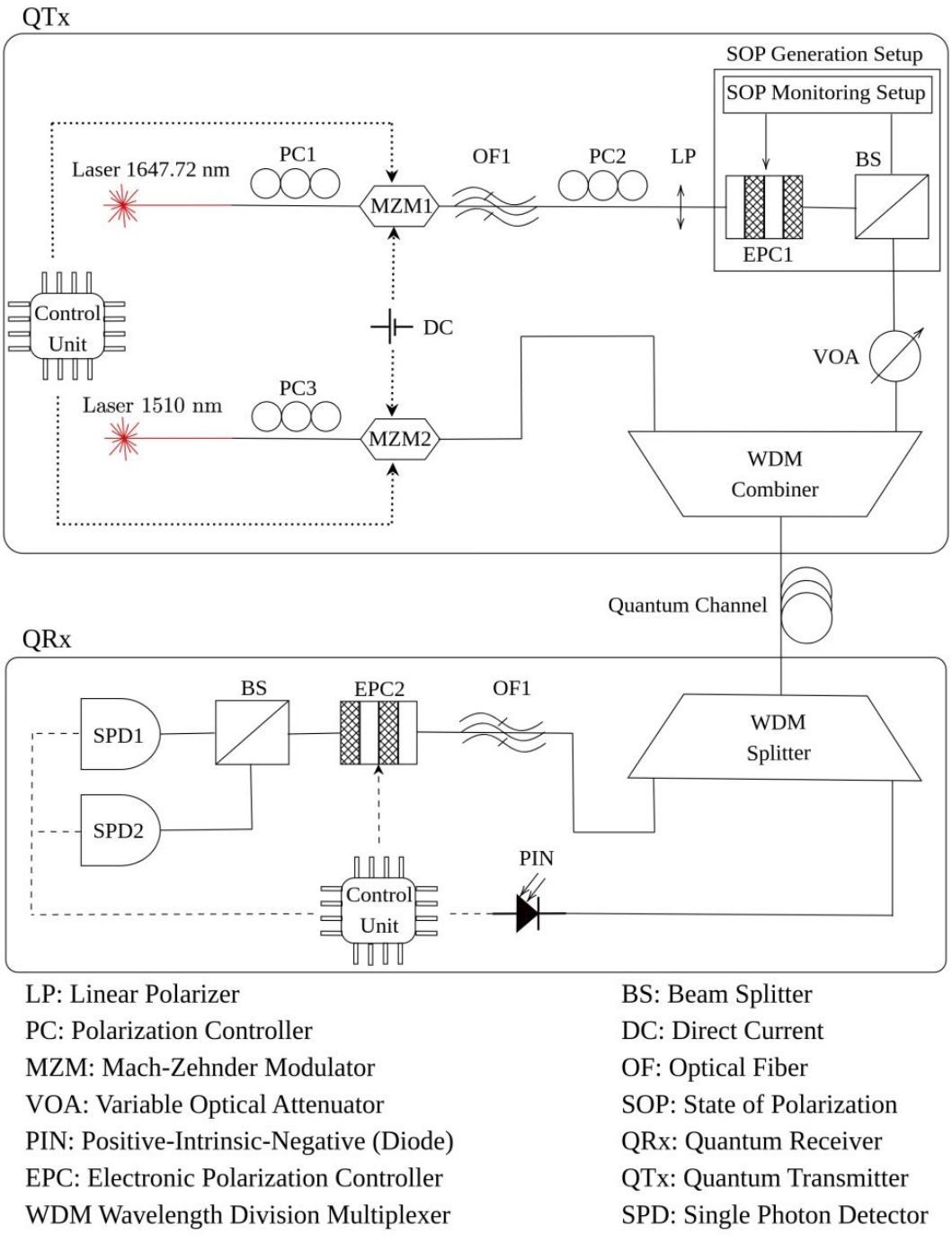


Figure 3.4: Architecture of quantum communication system [2].

$$P(n) = \frac{\mu^n e^{-\mu}}{n!}, \tag{3.25}$$

where $P(n)$ is the likelihood of observing n photons, μ is the average photon number. To ensure a good approximation of single-photon Fock states, the likelihood of a non-empty weak pulse is approximately

$$P(n > 0, \mu) \approx \frac{\mu}{2}. \tag{3.26}$$

However, this approach has a trade-off: when μ is very small, the bit rate decreases significantly due to the large number of empty pulses. In the implementation described, narrow laser pulses are initially produced with a larger number of photons per pulse and are later attenuated to quantum levels. This ensures unconditional secure communication while maintaining a practical bit rate.

Polarization Encoding

Consider the orthogonal basis $\{|H\rangle, |V\rangle\}$, in which the state of a qubit could be expressed as

$$a_0|H\rangle + a_1|V\rangle, \tag{3.27}$$

in which a_0 and a_1 are complex coefficients. Therefore, a general state of a qubit, expressed in terms of its polarization, could then be represented by

$$|\psi\rangle = \cos\frac{\theta}{2}|H\rangle + \sin\frac{\theta}{2}e^{i\phi}|V\rangle, \tag{3.28}$$

where θ is the angle that describes the relative amount of horizontal and vertical polarization, and ϕ is a phase factor that affects the relative phase between the $|H\rangle$ and $|V\rangle$ components. This representation is particularly useful because it can describe any pure qubit state as a point on the Poincaré sphere, as illustrated in Fig. 3.5. This sphere provides a geometrical way to visualize every feasible polarization states of a photon, where both north and south poles correspond to $|H\rangle$ and $|V\rangle$, respectively, and every point on the sphere's surface represents a valid polarization state of the qubit. Four commonly used polarization states—diagonal, anti-diagonal, right-circular, and left-circular polarization—could be represented using the standard orthonormal basis $\{|H\rangle, |V\rangle\}$, which defines a two-dimensional complex Hilbert space $\mathbb{C}^2$ as

$$|\pm 45^\circ\rangle = \frac{|H\rangle \pm |V\rangle}{\sqrt{2}},\ |R\rangle = \frac{|H\rangle + i|V\rangle}{\sqrt{2}},\ |L\rangle = \frac{|H\rangle - i|V\rangle}{\sqrt{2}}. \tag{3.29}$$

These states form a complete set of polarization states for a qubit, allowing a full representation of its quantum state in terms of linear and circular polarization.

Polarization modulation is achieved by an EPC, that consists of 4 separate waveplates. Each waveplate has a fixed fast-axis orientation and a adjustable retardation phase. These waveplates are controlled by voltage signals. The Mueller matrix for a general waveplate, characterized by an orientation angle θ and retardation δ is given as

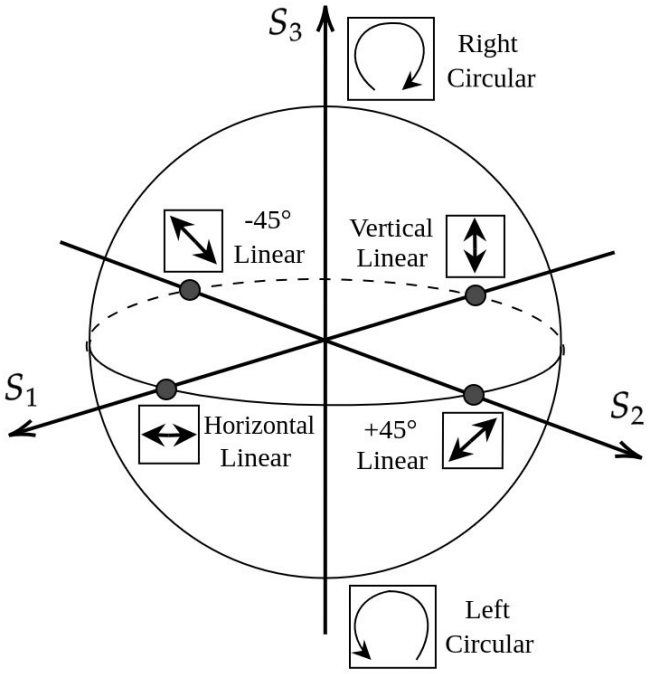


Figure 3.5: Poincaré sphere with the different states of polarization.

$$R(\theta,\delta) = \begin{bmatrix} \cos^2(2\theta) + \cos(\delta)\sin^2(2\theta) & -\cos(2\theta)\sin(2\theta)(\cos(\delta)-1) & -\sin(2\theta)\sin(\delta) \\ -\cos(2\theta)\sin(2\theta)(\cos(\delta)-1) & \cos(\delta)\cos^2(2\theta) + \sin^2(2\theta) & \cos(2\theta)\sin(\delta) \\ \sin(2\theta)\sin(\delta) & -\cos(2\theta)\sin(\delta) & \cos(\delta) \end{bmatrix}. \tag{3.30}$$

The SOP at both the input and output of the EPC can be described using Stokes parameters, which are represented as

$$\hat{s}_{in} = \begin{bmatrix} s_{in1} \\ s_{in2} \\ s_{in3} \end{bmatrix}, \tag{3.31}$$

and

$$\hat{s}_{out} = \begin{bmatrix} s_{out1} \\ s_{out2} \\ s_{out3} \end{bmatrix}, \tag{3.32}$$

where s_{in_i} and s_{out_i} are respectively the components of the input and output Stokes vectors. Each SOP corresponds to a set of 4 voltage values, with each voltage controlling one of the parameters δ_1, δ_2, δ_3, and δ_4, which can generate up to four distinct SOPs.

Communication Channel

The communication channel consists of several components: a WDM combiner at the transmitter output, which merges the quantum and classical reference signals together; an optical

fiber with a standard single-mode that transmits quantom and classical signals; and finally one WDM splitter at the receiver input, which divides the merged signals. Additionally, the channel includes a classical communication layer, which establishes a connection between the transmitter and receiver via Ethernet protocol.

DV-QKD Receiver

At the receiver end, Bob's primary work is to measure the incoming SOPs and decode them into classical bits. A schematic of the QKD Receiver (QRx) is shown in Fig. 3.4. At the receiver's input, the combined optical signals—comprising the quantum signal and the reference clock signal—are separated using a WDM splitter. The reference clock signal is routed directly to a photodetector, converting it from the optical domain to the electrical domain for synchronization purposes. The quantum signal, in contrast, passes through an optical filter designed to minimize noise by removing sideband wavelengths. The filtered quantum signal then enters the receiver's EPC, which selects the appropriate measurement basis. Finally, the quantum signal is directed into the detection module, consisting of a Polarization Beam Splitter (PBS) and two SPDs (SPD1 and SPD2).

Polarization Decoding

The implementation of DV quantum protocols fundamentally relies on the ability to successfully detect single photons. This detection process depends on selecting the appropriate measurement basis and the efficiency of the SPDs in accurately detecting photons. The SPDs are configured to operate in gated mode, triggered externally at a repetition rate of 500 Hz. This external trigger is generated by the Field-Programmable Gate Array (FPGA) board. Due to variations in the fabrication process of each detector, their efficiency settings differ. Assuming an arbitrary SOP at the receiver input, the SOP at the output of the EPC can be expressed as

$$|\hat{s}_B^{\text{in}}\rangle = \frac{\sqrt{2}}{2}|H\rangle e^{i\delta_1} e^{i(\delta_3+\delta_4+\delta_5+\delta_6)} + \frac{\sqrt{2}}{2}|V\rangle e^{i\delta_2} e^{i(\delta_3+\delta_4+\delta_5+\delta_6)}, \tag{3.33}$$

where $\delta_3, \delta_4, \delta_5$, and δ_6 represent the phase retardation introduced by every waveplates, determined by the applied voltages that define the measurement basis. Once the SOP arrives the polarizing BS, it can take one of two possible paths: one directed toward detector SPD1, and the other toward detector SPD2. For simplicity, let's assume that the transmitted light, which corresponds to the horizontal polarization, is routed toward SPD1, while the reflected light, associated with the vertical polarization, travels toward SPD2.

Basis Selecting

The basis selection process involves the EPC at the receiver, which rotates the reference axis to align with the measurement axis of the PBS. To achieve this alignment, an automatic calibration method is implemented to determine the optimal voltage values for each measurement basis. The calibration algorithm begins with the transmitter sending qubits encoded with a known polarization state representing bit 0. While receiving these qubits, the receiver calculates the Quantum Bit Error Rate (QBER) and initiates the process of scanning scanning voltage values in small increments. The scanning process continues until

the QBER begins to increase, at which point the algorithm stops and saves the voltage value corresponding to the minimum QBER. This process is repeated for each waveplate to ensure precise calibration. Once the calibration for horizontal and vertical states is completed, the same procedure is applied to diagonal and circular states. If the estimated QBER exceeds 0.5% at any point, the calibration process is repeated to refine the voltage settings. After all values are determined, they are stored in the EPC, ensuring consistent and accurate basis selection during quantum communication. This method effectively aligns the measurement basis with the transmitted qubits, minimizing detection errors and improving the accuracy of the quantum communication system.

3.1.3 Quantum Oblivious Key Distribution

QOKD is a cryptography protocol that generates and distributes oblivious keys. Oblivious keys are a type of keys where Alice possesses the complete keys, but Bob has access to only half of the keys. Importantly, Alice has no knowledge of which specific parts of the key Bob knows. She is only aware that Bob has learned exactly half of the key's content. One of the main uses of oblivious keys is the generation of QOT. QOT is a cryptographic protocol where a sender transfers one of many possible pieces of information to a receiver, but the sender remains unaware of which piece of information was received. Also, the receiver does not learn about the other message that they did not choose. Consider Alice and Bob as two communicating parties. A 1-out-of-2 QOT protocol takes m_0 and m_1 as Alice's inputs and $b \in \{0, 1\}$ as Bob's input. Then it outputs m_b to Bob and nothing to Alice. Through this protocol, Bob only obtains information about one message (m_b) and he learns nothing about the other message, and Alice learns nothing about Bob's choice (b).

Oblivious Transfer from Oblivious Keys

In [22], a QOT protocol based on QOKD was proposed, consisting of two distinct phases: the oblivious key phase and the oblivious transfer phase. As all heavy computations are shifted to the oblivious key phase, and this phase is implemented before and independently of the second phase, the complexity cost of OT generation through this protocol is significantly reduced. The oblivious key phase outputs k to Alice and $\tilde{k}$ to Bob as their oblivious keys. Note that only half of the bits of $\tilde{k}$ are identical to k; the other half is random. Along with $\tilde{k}$, Bob is provided with a bit stream x. This stream enables Bob to identify which bits in his key are perfectly correlated with Alice's and which bits are uncorrelated. To initiate the oblivious key phase, Alice and Bob agree on the two following non-orthogonal basis: computational basis ($|0\rangle$ and $|1\rangle$) and Hadamard basis ($|\pm\rangle = (|0\rangle \pm |1\rangle)/\sqrt{2}$). Additionally they define the states $|(p_i, b_i)\rangle$ for $p_i, b_i \in \{0, 1\}$ as

$$\begin{aligned} |(0,0)\rangle &= |0\rangle, \quad |(0,1)\rangle = |+\rangle, \\ |(1,0)\rangle &= |1\rangle, \quad |(1,1)\rangle = |-\rangle. \end{aligned} \tag{3.34}$$

Alice randomly selects p and b, and prepares quantum states $|\psi_i\rangle = |(p_i, b_i)\rangle$ for each index $i \leq n+m$. She sends the concatenated state $|\psi\rangle = |\psi_1\psi_2 \dots \psi_{n+m}\rangle$ to Bob. Bob randomly chooses $\tilde{b}$ and measures $|\psi_i\rangle$ in the computational basis when $\tilde{b}_i = 0$, and in the Hadamard basis otherwise. He then constructs the string $\tilde{p} = \tilde{p}_1\tilde{p}_2 \dots \tilde{p}_{n+m}$, where $\tilde{p}_i = 0$ if the measurement outcome of $|\psi_i\rangle$ is 0 or +, and $\tilde{p}_i = 1$ if it is 1 or −. Bob commits $(\tilde{p}_i, \tilde{b}_i)$ to Alice for each i. Commitment ensures that Bob follows the protocol honestly. Alice randomly selects a set

of indices S of size m and shares S with Bob. For each $j \in S$, Bob reveals the commitments corresponding to $(\tilde{p}_j, \tilde{b}_j)$. Alice verifies that $p_j = \tilde{p}_j$ whenever $b_j = \tilde{b}_j$ for all $j \in S$. If the verification process fails, Alice terminates the protocol. If successful, she transmits $b^* = b|_{\overline{S}}$ to Bob and sets $k = p|_{\overline{S}}$. Bob then computes $x = b^* \oplus \tilde{b}|_{\overline{S}}$ and $\tilde{k} = \tilde{p}|_{\overline{S}}$, where $\oplus$ denotes the bitwise XOR.

Protocol 2 QOT protocol based on secure commitments [22].

Input: Alice provides two strings, m_0 and m_1, while Bob provides a single bit c.
Output: Alice receives no output from the protocol, while Bob receives the message m_c, which corresponds to either m_0 or m_1 based on his input bit c. Specifically, if $c = 0$, Bob obtains m_0, and in case $c = 1$, he obtains m_1.

Quantum oblivious key distribution

1. Alice generates two random binary strings $p, b \in \{0,1\}^{n+m}$. For each index $i \leq n+m$ she prepares the quantum state $|\psi_i\rangle = |(p_i, b_i)\rangle$ and sends the composite state $|\psi\rangle = |\psi_1\psi_2 \ldots \psi_{n+m}\rangle$ to Bob.

2. Bob draws a random string $\tilde{b} \in \{0,1\}^{n+m}$. For each i, he measures $|\psi_i\rangle$ in the computational basis if $\tilde{b}_i = 0$; else he uses the Hadamard basis. Afterward, he constructs the string $\tilde{p} = \tilde{p}_1\tilde{p}_2 \ldots \tilde{p}_{n+m}$, in which $\tilde{p}_i = 0$ in case the measurement result is 0 or $+$, and $\tilde{p}_i = 1$ in case the result is 1 or $-$.

3. Bob commits to each pair $(\tilde{p}_i, \tilde{b}_i)$ and sends the commitments to Alice.

4. Alice draws a random subset of indices $S \subset \{1, \ldots, n+m\}$ and communicates it to Bob.

5. For every $j \in S$, Bob reveals the corresponding commitments for $(\tilde{p}_j, \tilde{b}_j)$.

6. Alice verifies that $p_j = \tilde{p}_j$ whenever $b_j = \tilde{b}_j$ for all $j \in S$. If any mismatch occurs, she aborts the protocol. Else, she sends $b^* = b|_{\overline{S}}$ to Bob and defines $k = p|_{\overline{S}}$.

7. Bob computes $x = b^* \oplus \tilde{b}|_{\overline{S}}$ and sets $\tilde{k} = \tilde{p}|_{\overline{S}}$.

Oblivious transfer

8. Bob constructs two sets of indices: $I_0 = \{i \mid x_i = 0\}$ and $I_1 = \{i \mid x_i = 1\}$. He then sends the pair $(I_c, I_{c\oplus 1})$ to Alice.

9. Alice computes two values, (l_0, l_1), where $l_i = m_i \bigoplus_{j \in I_{c\oplus i}} k_j$, and forwards them to Bob.

10. Bob reconstructs m_c by computing $m_c = l_c \bigoplus_{j \in I_0} \tilde{k}_j$.

Note that Bob's commitment is a crucial step as it ensures that Bob's measurements are genuine and were made before he knew Alice's chosen bases. Without commitment, Bob could potentially measure the qubits in any basis after Alice reveals her basis, which would allow him to cheat by learning the entire key. Having oblivious keys, parties can implement OT within the next phase. To initiate this phase, Bob specifies the sets $I_0 = \{i \,|\, x_i = 0\}$ and $I_1 = \{i \,|\, x_i = 1\}$ and transmits the pair $(I_c, I_c \oplus 1)$ to Alice. The set I_0 corresponds to the

positions in the oblivious keys where Bob has full knowledge of the bits. In other words, Bob is aware of the exact values of the bits at these specific indices. On the other hand, I_1 represents the set of indices for the oblivious keys that remain hidden from Bob, signifying that he has no knowledge of these particular bits. Alice then computes (l_0, l_1), where $l_i = m_i \bigoplus_{j \in I_{c \oplus i}} k_j$, and sends the result to Bob. The notation $\bigoplus_{j \in I_{c \oplus i}} k_j$ refers to indices of the oblivious key k that corresponds to $I_{c \oplus i}$ and $\bigoplus$ denotes the bitwise XOR. Bob computes $m_c = l_c \bigoplus_{j \in I_0} \tilde{k}_j$. The details of the OT implementation are explained in Protocol 2.

3.1.4 Key Management System

In the previous sections, we explained how different cryptographic keys such as random numbers, symmetric, and oblivious keys are generated through quantum protocols. However, once the keys are generated, they are not used immediately. Instead, the keys are stored in a KMS [125]. The KMS in the QSMC manages the synchronization, storage, and distribution of keys. The KMS also facilitates key relay across the network by enabling multiple KMSs to cooperatively establish matching keys between two arbitrary nodes, even when no direct quantum connection exists. While this process requires trust in intermediate nodes, it is managed by the network's controller, which enforces strict relay route policies to ensure that key relays occur through trusted and secure intermediate nodes, minimizing the risk of potential attacks or compromise. Notably, there is a distinction between relaying symmetric keys and oblivious keys. For symmetric keys, the adversary is typically external, making it easier to trust intermediate nodes with partial or complete key content. In contrast, for oblivious keys, where the adversary could be either an external party (Eve) or one of the participants (Alice or Bob), governance becomes more complex. The incorporation of symmetric keys generated from QKD into KMS is realized through European Telecommunications Standards Institute (ETSI) GS QKD 004 [126]. However, for incorporation of random numbers and oblivious keys in the KMS ad hoc solution was developed by [28]. Figure 3.6 illustrates a KMS connecting Site A and Site B. Each site features a Secure Application Entity (SAE) (in our case SAE is the QSMC) that obtains keys from a KMS.

The KMS starts by establishing a session for key exchange between nodes. This session is identified by a Key Stream ID (KSID), a unique identifier for the keystream created for a specific session. Afterward, `OPEN_CONNECT` function is called which reserves an association for the future keys and sets Quality of Service (QoS) parameters like key rate and buffer size. Both the source and destination applications specify these requirements through a QoS structure that defines:

- Key_chunk_size: The size of key segments in bytes.
- Min_bps/Max_bps: Minimum and maximum bit rate for key delivery.
- Jitter: Maximum allowable deviation in key delivery rate.
- Priority and Timeout: Used to prioritize and abort sessions if connections exceed time limits. This function blocks until a connection is established and KSID is confirmed. If a KSID is predefined and both parties have registered it, the KMS immediately proceeds to key generation and distribution.

Finally, when the session is complete or no longer needed, the `CLOSE` function is invoked. The `CLOSE` function ensures that no residual resources are left allocated, preventing memory leaks and ensuring that future sessions can be initialized without conflict.

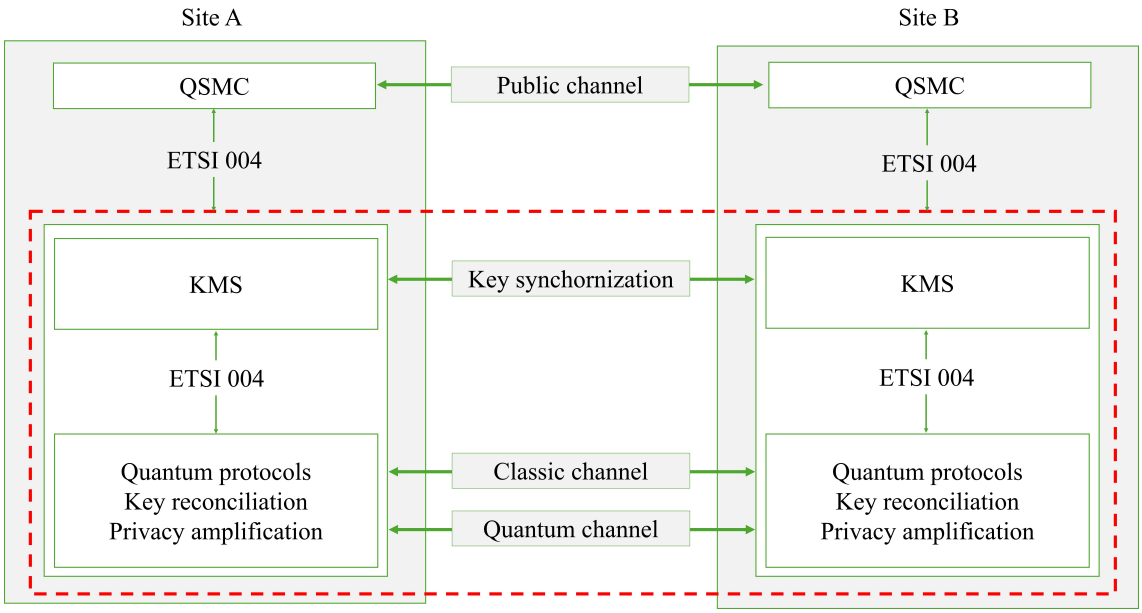


Figure 3.6: Architecture of a KMS.

3.1.5 MASCOT Protocol

Classical SMC can be achieved through protocols based on either arithmetic or Boolean circuits. Unlike Boolean circuits, arithmetic circuits are tailored for computations involving integers. These circuits efficiently handle addition and multiplication operations, making them particularly suitable for applications where numerical computations are essential. However, all existing practical protocols for SMC involving arithmetic circuits either depend on an honest majority or rely on the costly PKC techniques [7]. For example, SPDZ protocol [56] employs HE to perform secure multiplications which requires expensive ZKP or cut-and-choose techniques to guarantee security against adversaries. In contrast, MASCOT protocol [8] utilizes OT for secure computation on arithmetic circuits. One key advantage of OT is that it can be achieved in an offline phase, significantly accelerating the actual computation during the online phase. Alongside OT, MASCOT utilizes an OT extension protocol [62] to further enhance efficiency. This extension is necessary in SMC applications that require a large number of OTs. Additionally, MASCOT employs SS techniques. The SS-based SMC protocols for arithmetic circuits offer a significant advantage: for secure addition no communication among parties is required. MASCOT ensures resilience against a dishonest majority where any number of parties can potentially behave maliciously by providing incorrect inputs, intercepting communication, or altering data. As shown in Fig. 3.7, MASCOT operates in two distinct phases: offline and online. The online phase is designed to minimize computational overhead by conducting intensive computations upfront, thereby enhancing overall efficiency. While the online phase is where the actual computation on the secret-shared inputs takes place. In the following, we explain each phase in more detail.

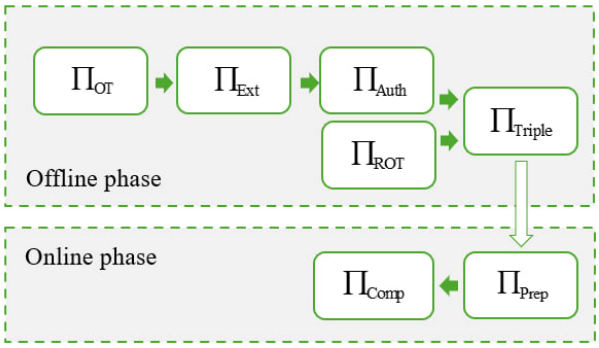


Figure 3.7: Different functionalities in MASCOT protocol during offline and online phases.

Offline Phase

The main goal of MASCOT offline phase is to generate multiplication triples ($[\![a]\!]$, $[\![b]\!]$, $[\![c]\!]$) that are required for multiplication operations. The foundation for triple generation lies in the protocol introduced by Gilboa [127], which performs multiplication of two k-bit field elements using k oblivious transfers. By executing OT between every pair of parties, multiplication triples can be generated. However, malicious parties may disrupt the process by supplying inconsistent inputs across different OT executions, leading to incorrect outcomes. To achieve an actively secure protocol, MASCOT improves Gilboa's protocol by employing simple consistency checks and privacy amplification techniques. The offline phase involves five protocols known as Π_{OT}, Π_{Ext}, Π_{Auth}, Π_{ROT}, and Π_{Triple}. These protocols are explained in detail in the following sections.

Oblivious Transfer - Π_{OT}

The ideal functionality for a 1-out-of-2 OT involving k-bit strings is outlined as follows:

$$\Pi_{\text{OT}}^{1,k} : ((m_0, m_1), b) \rightarrow (\perp, m_b),$$

where $m_0, m_1 \in \{0,1\}^k$, and $b \in \{0,1\}$ is the receiver's choice bit. The notation $\Pi^{l,k}$ is used to denote l instances of oblivious transfers performed on k-bit strings. The OT functionality used in MASCOT is based on [128], a classical protocol derived from the Diffie-Hellman Key Exchange (DHKE) in which players exchange secret keys in a secure way over a public communication channel. The protocol is known as Simple OT (S-OT), and is implemented as follows: Given a group $\mathbb{G}$ and a generator g, the sender (Alice) picks a random value a, calculates $A = g^a$ and sends it to the receiver (Bob). The generator g is a member of $\mathbb{G}$ such that $\forall i \in \mathbb{G}, \exists r : g^r = i$. Symmetrically, Bob samples a random value b and calculates B according to his arbitrary choice c. If $c = 0$, the receiver obtains $B = g^b$, and if $c = 1$, he obtains $B = Ag^b$ and sends it to Alice. Both players obtain $g^{ab} = A^b = B^a$ and derive their corresponding secret keys. Alice computes $k_0 = H(B^a)$ and $k_1 = H((B/A)^a)$ (H for hash) to obtain her keys. Symmetrically, Bob computes $k_R = H(A^b)$ according to his bit choice c. Having these keys, we can start to implement the oblivious transfer functionality, as explained in Protocol 3. Next, Alice encrypts her input messages m_0 and m_1, and obtains $e_0 = E_{k_0}(m_0)$ and $e_1 = E_{k_1}(m_1)$ (E for encryption), which then are sent to Bob. Bob computes $m_c = D_{k_R}(e_b)$ (D for decryption) according to his bit choice by decrypting e_b and

obtains the output message. Note that Bob can only decrypt one of the messages as he only has access to one of the keys.

Protocol 3 Π_{OT} - Implementation of S-OT protocol [8].

Inputs: Strings m_0 and m_1 for Alice, and the bit c for Bob.
Outputs: None for Alice and m_c for Bob.

Key exchange

Alice calls a DHKE service, which sends $k_0 = H(B^a)$ and $k_1 = H((B/A)^a)$ to Alice, and $k_R = H(A^b)$ to Bob.

Oblivious transfer

1. Alice computes $e_0 = E_{k_0}(m_0)$ and $e_1 = E_{k_1}(m_1)$ by encrypting her input messages and sending them to Bob.
2. Bob decrypts one of the messages using his key and obtains the output message $m_c = D_{k_R}(e_b)$.

In the next section, the generated OTs will be extended through Π_{Ext} using symmetric cryptography.

Oblivious Transfer Extension - Π_{Ext}

Π_{Ext} serves two primary purposes: first, it operates as an OT extension protocol, utilizing a minimal number of base OTs and significantly expanding them through symmetric cryptography [64]. Second, it allows two parties to compute the product of $x\Delta$ that is called Correlated Oblivious Product Evaluation (COPE), which is necessary for subsequent authentication steps. In the following, we explain each functionality in more detail.

Oblivious Transfer Extension

To extend the number of OT, Alice begins by generating several pairs of random values, known as seeds. These seeds are then utilized in a base OT protocol where Alice and Bob engage in a secure exchange. During this exchange, Bob selects which value from each seed pair he wants to receive without revealing his specific choices to Alice. As a result, Bob ends up with a set of seeds that correspond to his selections. In the subsequent extension phase, the protocol employs a Pseudo-Random Function (PRF). The PRF is a cryptographic tool that processes each seed along with a fixed additional value to generate pseudorandom outputs. Alice uses the PRF to compute outputs from the seeds she provided, while Bob uses the PRF with the seeds he received to produce results based on his own choices. This method enables both parties to securely generate a large number of results from the original set of base OT seeds. The symmetric nature of the process comes from the fact that both Alice and Bob use the same PRF and the same seeds. The extension process is explained in Protocol 4.

Protocol 4 Π_{Ext} - OT extension [8].

This protocol utilizes a PRF $F : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \mathbb{F}$ and outputs $x \cdot \Delta$. Here, one party possesses $x \in \mathbb{F}$ while the other party has $\Delta \in \mathbb{F}$. The Δ remains constant throughout the protocol execution and each iteration generates shares for different values of x.

Initialize

1. Alice generates k pairs of seeds, denoted as $\{(\mathbf{k}_i^0, \mathbf{k}_i^1)\}_{i=1}^k$, where each seed is a binary string of length λ from the set $\{0,1\}^\lambda$.
2. Both parties execute $\Pi_{\text{OT}}^{k,\lambda}$ with inputs $\{\mathbf{k}_i^0, \mathbf{k}_i^1\}_{i\in[k]}$ from Alice and $\mathbf{\Delta}_B = (\Delta_0, \ldots, \Delta_{k-1}) \in \{0,1\}^k$ from Bob. Each bit Δ_i in $\mathbf{\Delta}_B$ indicates which element of the i-th pair Bob wishes to receive.
3. Bob receives $\mathbf{k}_i^{\Delta_i}$ for $i \in [k]$. This means the Bob gets the Δ_i-th element of the i-th pair (k_i^0, k_i^1). In other words, if $\Delta_i = 0$, Bob gets k_i^0, and if $\Delta_i = 1$, the he gets k_i^1.

Extend

4. For each $i = 1, \ldots, k$, repeat steps 5-8.
5. Alice computes $t_i^0 = F(\mathbf{k}_i^0, j) \in \mathbb{F}$ and $t_i^1 = F(\mathbf{k}_i^1, j) \in \mathbb{F}$, such that she knows (t_i^0, t_i^1).
6. Bob computes $t_i^{\Delta_i} = F(\mathbf{k}_i^{\Delta_i}, j) \in \mathbb{F}$ where Δ_i determines whether the he receives t_i^0 or t_i^1 based on his choice.

COPE

7. Alice computes $u_i = t_i^0 - t_i^1 + x$ and sends the result to Bob.
8. Bob computes $q_i = \Delta_i u_i + t_i^{\Delta_i} = t_i^0 + \Delta_i x$.
9. Parties define $\mathbf{q} = (q_1, \ldots, q_k)$ and $\mathbf{t} = (t_1^0, \ldots, t_k^0)$. Note that $\mathbf{q} = \mathbf{t} + x\mathbf{\Delta}_B \in \mathbb{F}^k$.
10. Bob outputs $q = \langle \mathbf{g}, \mathbf{q} \rangle$.
11. Alice outputs $t = -\langle \mathbf{g}, \mathbf{t} \rangle$.
12. Parties obtain $t + q = x\Delta \in \mathbb{F}$.

Correlated Oblivious Product Evaluation

To perform a linear operation on private inputs, MASCOT uses secret sharing in which a secret value is shared among multiple parties in a way that each party holds a share of the secret, and the shares can be added together to reconstruct the original secret. Parties can perform operations on these shares, such as adding them together or multiplying by a constant. However, at this stage, a malicious party may try to provide incorrect values for the shares. To prevent tampering, each share is paired with a Message Authentication Code (MAC). A MAC is a cryptographic tool that ensures the share hasn't been altered. It acts like a digital signature for each share. By verifying the MACs, parties can ensure that the shares are correct and the final result is accurate. Considering a secret sharing scheme, a

secret value $x \in \mathbb{F}$ is defined as

$$[\![x]\!] = \left(x^{(1)}, ..., x^{(n)}, m^{(1)}, ..., m^{(n)}, \Delta^{(1)}, ..., \Delta^{(n)}\right), \tag{3.35}$$

in which each party P_i owns $x^{(i)}$ as a random share of x, a random MAC share $m^{(i)}$ and a constant MAC key share $\Delta^{(i)}$ such that

$$x = \sum_i x^{(i)}, \;\; m = \sum_i m^{(i)}, \;\; \Delta = \sum_i \Delta^{(i)}, \tag{3.36}$$

and the MAC relation is

$$m = x\Delta. \tag{3.37}$$

The Π_{Ext} enables two parties to compute additive share of $x\Delta$, which is necessary for subsequent MAC authentication.

Oblivious Product Evaluation

To achieve an additive sharing of the product $a \cdot b$, the two parties execute k instances of oblivious transfer on k-bit strings. In each OT instance, the sender provides $t_i \in \mathbb{F}$ and its correlated value $t_i + a$ randomly, where $a \in \mathbb{F}$ is the input of the sender. On the other hand, the receiver provides the binary representation of $(b_1, \ldots, b_k) \in \{0,1\}^k$, and obtains either t_i or $t_i + a$ based on the value of b_i leading to

$$q_i = t_i + b_i a\,, \tag{3.38}$$

where q_i represents the receiver's output in the i-th OT. Each party computes the inner product of $(q_i)_i$ and $(-t_i)_i$ to obtain q and t using the gadget vector g. Afterward, the additive share of input products is obtained by

$$q + t = ab. \tag{3.39}$$

Correlated OPE

To compute the additive share of $x\Delta$, where one party's input remains constant across multiple protocol runs, the k OTs only need to be executed once. This functionality is implemented in Protocol 4, which comprises three phases: Initialize, Extend, and COPE. During the Initialize phase, the sender, Alice, provides pairs of seeds $(\mathbf{k}_i^0, \mathbf{k}_i^1)$, and the receiver chooses one of them based on their choice bit Δ_i. The receiver, Bob, learns one of the seeds without the sender knowing which one. During the subsequent OT extension phase, in each call, the parties use a PRF to expand the original seeds, generating k fresh random OT bits, maintaining the same receiver's choice Δ_B. The PRF F takes two inputs: a seed, which is a binary string of length λ securely exchanged during the initial base OT phase, and an index, which is a public counter ensuring that each application of the PRF generates a unique output. Note that λ represent the computational security parameter and $k = \log|\mathbb{F}|$ defines the bit length of the field elements. During the COPE phase, using x, Alice establishes a correlation between the two sets of PRF outputs. Afterward, the correlation is masked and transmitted to Bob, who modifies the PRF output based on this information. Consequently, both parties possess k correlated OTs. The correlated OTs are subsequently merged into a unified field element by

computing the dot product of their results with the gadget vector g, resulting in an additive sharing of $x\Delta$. The communication overhead for each round of the Π_{Ext}, following the k initial base OTs, amounts to k field elements, equating to k^2 bits in total. The computational cost involves performing $3k$ PRF and $8k$ field operations. Now consider a corrupted P_1^*. The issue is that P_1^* can freely choose which value to reveal at the time of opening, without being committed to a specific value which undermines protocol security. To avoid this, it is necessary to introduce authentication of an additional random value x_0 and verification of a random linear combination of all MACs in the **Input** phase. This solution involves two modifications: first, P_1 introduces a random dummy input x_0 which is authenticated alongside the actual inputs. Afterward, P_1 reveals a random linear combination of $x_0, \ldots, x_l$, and all parties verify the MAC of this combination. The inclusion of x_0 masks the true inputs, preventing P_1 from changing the revealed value later and ensuring commitment to their inputs during the **Input** phase. The implementation steps are shown in Protocol 5.

Authentication and opening additive shares - Π_{Auth}

As previously explained, Π_{Ext} is utilized for generating authenticated shares. However, Π_{Ext} alone does not guarantee active security as malicious adversaries can provide inconsistent inputs during Π_{Ext} execution. To achieve active security, MASCOT employs MAC to ensure that data originates from a trusted source and has not been altered by an adversary. Essentially, Π_{Auth} ensures that secret shared values are correct and have not been altered during Π_{Ext} execution. Π_{Auth} includes five phases:

- **Input:** receives $x_1, \ldots, x_l$ along with their corresponding identifiers $\text{id}_1, \ldots, \text{id}_l$ from one party.
- **LinComb:** computes linear functions on the input values.
- **Open:** reconstructs a value that has been secretly shared among multiple parties and then distributing this reconstructed value back to the involved parties.
- **Check:** involves verifying the correctness of a value produced by parties maintains the integrity of the computation.
- **Abort:** stops the protocol execution and notifies all parties of the failure.

We begin by exploring a straightforward method for a single party to generate authenticated shares of their private inputs using the Π_{Ext}, and discuss its limitations in achieving active security. Subsequently, we demonstrate the attainment of an actively secure protocol by incorporating the authentication of an additional random value and verifying a random linear combination of all MAC during the input phase. Let us consider a situation with two parties P_1 and P_2. Assume that P_1 is honest and intends to authenticate $x \in \mathbb{F}$. P_1 and P_2 execute a sample of Π_{Ext} with x being the input of P_1, while MAC key share $\Delta^{(2)}$ is the input of P_2. Afterward, P_1 gets t and P_2 acquires q leading to $q + t = x\Delta^{(2)}$. Next, P_1 specifies the MAC share $m^{(1)} = x\Delta^{(1)} + t$, and P_2 determines the MAC share $m^{(2)} = q$. Subsequently, we obtain $m^{(1)} + m^{(2)} = x\Delta$. To divide x into shares among the parties, P_1 produces random additive shares $x^{(1)}, x^{(2)}$ and transmits $x^{(2)}$ to P_2.

Protocol 5 Π_{Auth} - Authentication [8].

This protocol distributes and verifies the authenticity of inputs in the field $\mathbb{F}$, enabling linear operations and reconstruction of the original inputs from their shares.

Initialize

1. Each party P_i generates a share of a MAC key denoted as $\Delta^{(i)} \in \mathbb{F}$.
2. Each pair of parties (P_i, P_j) (where $i \neq j$) invokes $\Pi_{\text{Ext}} \cdot$ **Initialize** with party P_j providing the input $\Delta^{(j)}$.

Input

After receiving (Input, $\text{id}_1, \ldots, \text{id}_l, x_1, \ldots, x_l, P_j$) from P_j and (Input, $\text{id}_1, \ldots, \text{id}_l, P_j$) from others:

1. P_j draws $x_0 \in \mathbb{F}$ randomly.
2. For each $h = 0, \ldots, l$, P_j produces share $x_h = \sum_i x_h^{(i)}$ and sends each share $x_h^{(i)}$ to P_i.
3. P_i and P_j $(i \neq j)$ call $\Pi_{\text{Ext}} \cdot$ **Extend** and $\Pi_{\text{Ext}} \cdot$ **COPE**, where P_j inputs $(x_0, \ldots, x_l) \in \mathbb{F}^{l+1}$.
4. P_i is given $q_h^{(i,j)}$ and P_j is given $t_h^{(j,i)}$ with the condition that $q_h^{(i,j)} + t_h^{(j,i)} = x_h \Delta^{(i)}$.
5. Each P_i calculates their MAC shares $m_h^{(i)} = q_h^{(i,j)}$ and P_j calculates their MAC shares $m_h^{(j)} = x_h \Delta^{(j)} + \sum_{j \neq i} t_h^{(j,i)}$ to obtain $[\![x_h]\!]$.
6. The parties sample random vector $r \in \mathbb{F}^{l+1}$.
7. P_j calculates the value $y = \sum_{h=0}^{l} r_h x_h$ and sends it to all the other parties.
8. Each party P_i calculates a weighted sum of their MAC shares $m^{(i)} = \sum_{h=0}^{l} r_h m_h^{(i)}$.
9. Given the revealed value y, along with $m^{(i)}$ and $\Delta^{(i)}$, each party P_i performs the MAC checking as follows:
   (a) Calculate $\sigma^{(i)} = m^{(i)} - y\Delta^{(i)}$ and then use the function $\mathcal{F}_{\text{Comm}}$ to commit to this value, receiving a commitment handle τ_i in return.
   (b) Invoke the function $\mathcal{F}_{\text{Comm}}$ with the input (Open, τ_i) in order to reveal the committed value.
   (c) In case $\sigma^{(1)} + \cdots + \sigma^{(n)} \neq 0$, output $\perp$ (failure) and abort. Else, resume with the protocol.
10. Each party retain their shares and corresponding MAC shares, which are linked to the the handles $\text{id}_1, \ldots, \text{id}_l$.

Protocol 5 (continued)

Linear combination

Upon receiving the input (LinComb, $\overline{\text{id}}$, id_1, ..., id_t, c_1, ..., c_t, c), the parties access the respective shares and MAC shares $\{x_j^{(i)}, m(x_j)^{(i)}\}_{j\in[t], i\in[n]}$ connected to identifiers $id_1, \ldots, id_t$, and then each P_i calculates

$$y^{(i)} = \sum_{j=1}^{t} c_j x_j^{(i)} + \begin{cases} c & i = 1 \\ 0 & i \neq 1 \end{cases}, \tag{3.40}$$

$$m(y)^{(i)} = \sum_{j=1}^{t} c_j m(x_j)^{(i)} + c\Delta^{(i)}. \tag{3.41}$$

Afterward, the parties save the new share and its corresponding MAC of $[\![y]\!]$ under the handle $\overline{id}$.

Open

Upon receiving the input (Open, id):

1. Each party P_i retrieves and sends their respective share $x^{(i)}$ to the other parties.
2. The parties then combine all the shares to compute $x = \sum_{i=1}^{n} x^{(i)}$ and the resulting value is output.

Check

Upon receiving the input $(\text{Check}, \text{id}_1, \ldots, \text{id}_t, x_1, \ldots, x_t)$, the below paces are executed.

1. A random vector $r \in \mathbb{F}$ is sampled publicly.
2. Value $y = \sum_{j=1}^{t} r_j x_j$ and $m(y)^{(i)} = \sum_{j=1}^{t} r_j m_{\text{id}_j}^{(i)}$ are computed, where $m_{\text{id}_j}^{(i)}$ is MAC share of P_i that is stored under id_j for every $i \in [n]$ and $j \in [t]$.
3. Go to step 9 to perform MAC checking using y and $m(y)^{(i)}$.

Random Oblivious Transfer - Π_{ROT}

Random Oblivious Transfer (ROT) is a variant of OT where input messages are chosen randomly. A 1-out-of-2 ROT is outlined as

$$\Pi_{\text{ROT}}^{1,k} : (\perp, b) \rightarrow ((m_0, m_1), m_b). \tag{3.42}$$

We use the notation $\Pi_{ROT}^{l,k}$ to denote l sets of ROT on k-bit strings. The ROT protocol is used in MASCOT to facilitate the secure generation of random authenticated shared values, without requiring parties to explicitly select m_0 and m_1, which would otherwise demand additional rounds of communication and coordination. The random shared values are essential

for the preprocessing phase of secure computations.

Triple Generation - Π_{Triple}

So far, we explained that parties utilize Π_{Auth} to calculate linear functions on their private inputs. Now, we extend this capability to nonlinear functions by generating multiplication triples using Π_{Triple}. This functionality generates multiplication triples that are additive secret sharing of ($[\![a]\!]$, $[\![b]\!]$, $[\![c]\!]$), where a, b are random values and $c = ab$. In addition to these triples, the values $a\Delta$, $b\Delta$, and $c\Delta$ are generated to authenticate a, b, and c, respectively. For a triple generation, MASCOT ensures both correctness and privacy of the triples. To check the correctness, a common sacrifice method [129], which verifies a pair of triples is utilized. Privacy is guaranteed using the privacy amplification technique [129], where initially, a number of imperfect triples are generated, and from them, one random triple is derived by computing random linear combinations.

To generate triples, utilizing Gilboa protocol [127], MASCOT produces a correlated vector triple $(\mathbf{a}, b, \mathbf{c})$ where $b \in \mathbb{F}$ and $\mathbf{a}, \mathbf{c} \in \mathbb{F}^\tau$ and τ is constant. Parties choose $\mathbf{r} \in \mathbb{F}^\tau$ as a public random vector and generate the triple (a, b, c) by defining

$$a = \langle \mathbf{a}, \mathbf{r} \rangle, \quad c = \langle \mathbf{c}, \mathbf{r} \rangle. \tag{3.43}$$

The process is repeated to obtain another triple to verify correctness through a sacrifice step. However, the efficiency of this phase can be improved by leveraging the vector triple $(\mathbf{a}, b, \mathbf{c})$ to derive a second correlated triple. To achieve this, parties choose another random public vector $\hat{\mathbf{r}}$ and compute $\hat{a}$ and $\hat{c}$ accordingly.

The triple $(\hat{a}, b, \hat{c})$ are then used to verify the correctness of (a, b, c). After incorporating MACs into both triples, the parties draw a random value $s \in \mathbb{F}$ and open $\rho = sa - \hat{a}$. This results in equation

$$sc - \hat{c} - b\rho = s(c - ab) + (\hat{a}b - \hat{c}). \tag{3.44}$$

If the result is zero, both triples are correct. Otherwise, if the result is non-zero, one or both triples are false. The next step involves verifying privacy during triple generation. To achieve this, various privacy amplification techniques can be employed. For instance, in [129], privacy amplification was performed on a large set of triples employing Shamir SS. However, MASCOT ensures privacy by mitigating leakage from one of the three triple values, accomplished by merging a vector of correlated triples of constant size. The functionality for triple generation is implemented within Protocol 6.

Online Phase

The online phase of MASCOT mirrors that of SPDZ, which has been proven to deliver highly efficient performance [94]. The online phase is where the actual computation occurs and involves two protocols called Π_{Prep} and Π_{Comp} that are explained in the following.

Protocol 6 Π_{Triple} - Triple generation [8].

It generates the triple $([\![a]\!], [\![b]\!], [\![c]\!])$. $\tau \geq 3$ determines the number of triples to be produced for each output triple.

Multiply

1. Each party chooses $\mathbf{a}^{(i)} \in \mathbb{F}^\tau$ and $b^{(i)} \in \mathbb{F}$ randomly.
2. Every pair of parties P_i and P_j perform as follows:
   (a) Both parties engage in the protocol $\Pi_{\text{ROT}}^{\tau k.k}$, where each party P_i provides its input $(a_1^{(i)}, \ldots, a_{\tau k}^{(i)}) = \mathbf{g}^{-1}(\mathbf{a}^{(i)}) \in \mathbb{F}_2^{\tau k}$.
   (b) Party P_j receives $(q_{0,h}^{(j,i)}, q_{1,h}^{(j,i)}) \in \mathbb{F}$ and P_i obtains $s_h^{(i,j)} = q_{a_h^{(i)},h}^{(j)}$ for each index $h = 1, \ldots, \tau k$.
   (c) P_j transmits $d_h^{(j,i)} = q_{0,h}^{(j,i)} - q_{1,h}^{(j,i)} + b^{(j)}$, $h \in [\tau k]$ and P_i prepares $t_h^{(i,j)} = s_h^{(i,j)} + a^{(i)} \cdot d_h^{(j,i)} = q_{0,h}^{(j,i)} + a_h^{(i)} \cdot b^{(j)}$ for $h = 1, \ldots, \tau k$.
   (d) The values $(t_1^{(i,j)}, \ldots, t_{\tau k}^{(j,i)})$ and $(q_1^{(j,i)} \ldots q_{\tau k}^{(j,i)})$ are partitioned into τ separate vectors, each consisting of k components, forming $(\mathbf{t}_1, \ldots, \mathbf{t}_\tau)$ and $(\mathbf{q}_1, \ldots, \mathbf{q}_\tau)$, respectively.
   (e) Party P_i prepares $\mathbf{c}_{i,j}^{(i)} = (\langle \mathbf{g}, \mathbf{t}_1 \rangle, \ldots, \langle \mathbf{g}, \mathbf{t}_\tau \rangle) \in \mathbb{F}^\tau$ and Party P_j prepares $\mathbf{c}_{i,j}^{(i)} = -(\langle \mathbf{g}, \mathbf{q}_1 \rangle, \ldots, \langle \mathbf{g}, \mathbf{q}_\tau \rangle) \in \mathbb{F}^\tau$, leading to $\mathbf{c}_{i,j}^{(i)} + \mathbf{c}_{i,j}^{(j)} = \mathbf{a}^{(i)} \cdot b^{(j)} \in \mathbb{F}^\tau$.
   (f) Each party P_i calculates $\mathbf{c}^{(i)} = \mathbf{a}^{(i)} \cdot b^{(i)} + \sum_{j \neq i} (\mathbf{c}_{i,j}^{(i)} + \mathbf{c}_{j,i}^{(i)})$.

Combine

1. The parties choose random values r and $\hat{r} \in \mathbb{F}^\tau$.
2. Party P_i calculates $a^{(i)} = \langle \mathbf{a}^{(i)}, \mathbf{r} \rangle$, $\hat{a}^{(i)} = \langle \mathbf{a}^{(i)}, \hat{\mathbf{r}} \rangle$, $c^{(i)} = \langle \mathbf{c}^{(i)}, \mathbf{r} \rangle$, and $\hat{c}^{(i)} = \langle \mathbf{c}^{(i)}, \hat{\mathbf{r}} \rangle$

Authenticate

1. Each party P_i executes $\Pi_{\text{Auth}} \cdot \textbf{Input}$ on their shares to generate the authenticated shares of the values $[\![a]\!], [\![b]\!], [\![c]\!], [\![\hat{a}]\!], [\![\hat{b}]\!], [\![\hat{c}]\!]$.

Sacrifice

1. To verify the correctness of the triple $([\![a]\!], [\![b]\!], [\![c]\!])$, the parties utilize the second triple $[\![\hat{a}]\!], [\![\hat{c}]\!]$ as follows:
   (a) Parties choose a random value $s \in \mathbb{F}$.
   (b) Parties compute $s[\![a]\!] - [\![\hat{a}]\!]$ and store it under $[\![\rho]\!]$, by calling $\Pi_{\text{Auth}} \cdot \textbf{LinComb}$.
   (c) Parties reveal ρ using input $[\![\rho]\!]$, by calling $\Pi_{\text{Auth}} \cdot \textbf{Open}$.
   (d) Parties store $s[\![c]\!] - [\![\hat{c}]\!] - [\![b]\!]\rho$ under $[\![\sigma]\!]$, by calling $\Pi_{\text{Auth}} \cdot \textbf{LinComb}$.
   (e) Parties execute $\Pi_{\text{Auth}} \cdot \textbf{Check}([\![\rho]\!], [\![\sigma]\!], \rho, 0)$ and abort if it fails.

Prepossessing Input Tuples - Π_{Prep}

In addition to generating multiplication triples, the preprocessing phase should also create random shared values known by only one party, referred to as input tuples. This enables the designated party to contribute inputs during the online phase. Protocol 7 facilitates this process: the relevant party provides a random value to Π_{Auth}. During the online phase, this party broadcasts the difference between the random value and their actual input, allowing all parties to adjust the shared random value to the correct input.

Protocol 7 Π_{Prep} - Prepossessing input tuples [8].

Parties preprocess input tuple using input (Input, P_j) as follows:

1. P_j selects a random value $r \in \mathbb{F}$, and submit it to Π_{Auth} with (Input, r , P_j).
2. All parties output the authenticated share $[\![r]\!]$ and P_j outputs r.

Computation - Π_{Comp}

It performs the actual computation. First, each party provides private input which is then secret-shared among all parties. To perform a linear operation on private inputs, let $[\![x]\!]$ and $[\![y]\!]$ represent the secret shares of x and y, respectively. To compute $[\![x \pm y]\!]$, each party locally computes $(x^{(1)} \pm y^{(1)})$ and $(x^{(2)} \pm y^{(2)})$ without any additional communication. Afterwards, they reveal their results, reconstructing $[\![x \pm y]\!] = (x^{(1)} \pm y^{(1)}) + (x^{(2)} \pm y^{(2)})$. However, multiplication is more complex and relies on the preprocessed multiplication triples and the Beaver's multiplication trick [130]. Assume that a pre-shared triple $([\![a]\!], [\![b]\!], [\![ab]\!])$ is available, where a and b are independent random values. To compute $[\![xy]\!]$ the parties first reveal $x + a$ and $y + b$. Revealing these values does not compromise the secrecy of x or y because a and b are secret and randomly chosen. Using the revealed values, the parties compute the product as

$$[\![xy]\!] = [\![(x + a - a)(y + b - b)]\!] = (x + a)(y + b) - (x + a)[\![b]\!] - (y + b)[\![a]\!] + [\![ab]\!]. \quad (3.45)$$

3.1.6 Circuit Generation

As explained, SMC enables multiple parties to securely compute a function. In this context, the "function" represents the specific task being executed, such as performing a mathematical operation or training a machine learning model. To compute a function within QSMC framework, the function needs to be converted into a format compatible with MASCOT protocol, typically referred to as an arithmetic circuit. The arithmetic circuit receives inputs and provides outputs. In the context of QSMC, the inputs of the arithmetic circuit are the private information held by each party, and the output is the final result of the computation.

To generate an arithmetic circuit, we use the circuit generation tool proposed by MP-SPDZ library [98]. To this end, the function is programmed in high-level Python-like language. For instance, Listing 3.1 illustrates the Python-like code used to generate the arithmetic circuit for calculating the average of multiple values. After programming the arithmetic circuit, a compiler translates this high-level code into bytecode, as shown in Fig. 3.8. This bytecode is

Protocol 8 Π_{Comp} - Computation [8].

Initialize: Parties execute Π_{Prep} to generate multiplication triples $(\llbracket a \rrbracket, \llbracket b \rrbracket, \llbracket c \rrbracket)$ and obtain references to these triplets. Parties mask values $(r_i, \llbracket r_i \rrbracket)$ as required by the function being computed. If F_{Prep} fails, the parties terminate the protocol and output $\perp$.
Input: To securely share an input x_i, party P_i uses a pregenerated mask $(r_i, \llbracket r_i \rrbracket)$ and performs as follows:

1. Party P_i computes and broadcast $\epsilon = x_i - r_i$ to all parties.
2. The parties calculate $\llbracket x_i \rrbracket = \llbracket r_i \rrbracket + \epsilon$.

Add: Using the input $\llbracket x \rrbracket, \llbracket y \rrbracket$, parties locally compute the sum as $\llbracket x + y \rrbracket = \llbracket x \rrbracket + \llbracket y \rrbracket$.
Multiply: Using input $\llbracket x \rrbracket, \llbracket y \rrbracket$, parties proceed as follows:

1. Parties retrieve a preprocessed triple $(\llbracket a \rrbracket, \llbracket b \rrbracket, \llbracket c \rrbracket)$ and compute $\llbracket \epsilon \rrbracket = \llbracket x \rrbracket - \llbracket a \rrbracket$, $\llbracket \rho \rrbracket = \llbracket y \rrbracket - \llbracket b \rrbracket$ and execute Π_{Prep}.**Open** to reveal ϵ, ρ.
2. Parties compute $\llbracket z \rrbracket = \llbracket c \rrbracket + \epsilon \llbracket b \rrbracket + \rho a + \epsilon \rho$.

Output: To reveal a share $\llbracket y \rrbracket$, parties perform as follows:

1. Parties execute Π_{Prep}.**Check** on all previously opened values.If verification fails, they terminate the protocol and abort.
2. To reveal and verify $\llbracket y \rrbracket$, parties execute $\Pi_{\textbf{Prep}} \cdot \text{Open}$ and then $\Pi_{\textbf{Prep}} \cdot \textbf{Check}$. If the verification is unsuccessful, they abort, otherwise accept y as the final result.

subsequently executed by a Virtual Machine (VM), providing the output of the computation. In this thesis, we implement two arithmetic circuits that are explained in more detail in Chapter 4.

```
def mean(value, num):
    sum = sint(0)
    for i in range(num):
        sum = sum + value[i]
        Sf = sfix(0)
        Sf.load_int(sum)
        Nf = sfix(0)
        Nf.load_int(num)
    mean = Sf / Nf
    return mean

num_input = 1000

input_values = Array(num_input, sint)

for i in range(num_input):
    input_values[i] = sint.get_input_from(2)

result = mean(input_values, num_input)
```

Listing 3.1: The Python-like code to generate the arithmetic circuit for calculating the average of multiple values

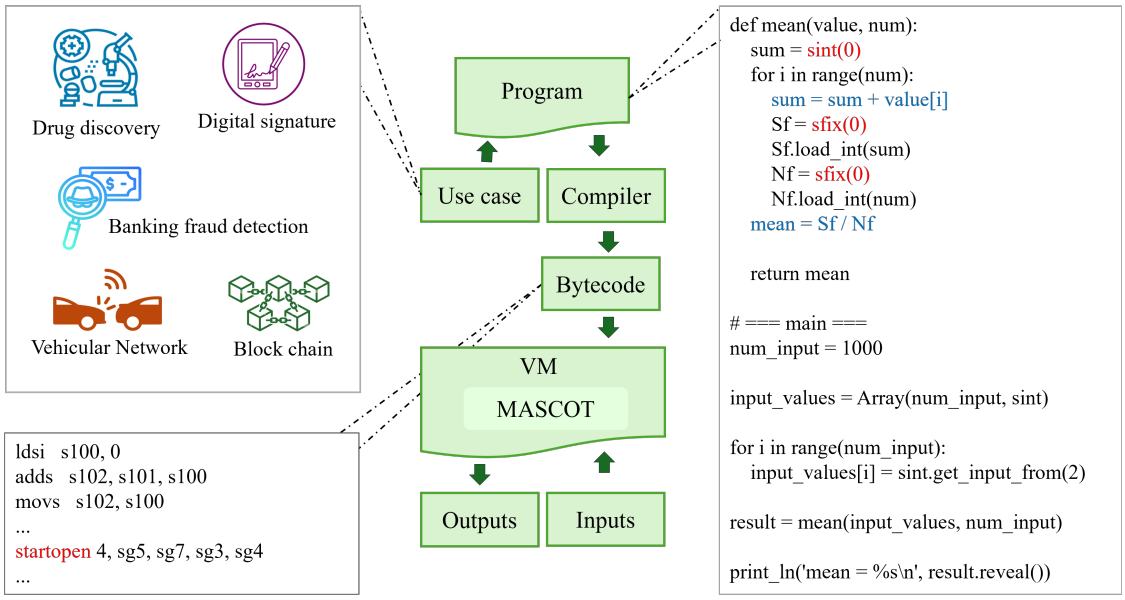


Figure 3.8: Steps to generate and execute an arithmetic circuit within QSMC framework.

3.2 Framework Implementation

We use the MASCOT implementation of the QMP-SPDZ library [131] as our starting point. In the first step, we integrated quantum-generated random numbers into the framework. To this end, we considered that MASCOT relies on AES-128 [132] in counter mode (AES-CTR) as its pseudorandom number generation. This approach is favored for its efficiency and security, as it utilizes the AES New Instructions (AES-NI) [133], which leverages modern processor hardware acceleration to speed up random number generation. The key idea is that AES encryption in CTR mode transforms a predictable, sequential counter into unpredictable, secure random numbers. To this end, a secure seed is initialized by the party along with a 128-bit counter where AES encryption is applied in CTR mode. The seed can come from a number of sources such as the Libsodium PRNG [134] or protocol for joint generation. In the next step, the counter is incremented and AES encryption is repeated to generate a sequence of random numbers as shown in Listing 3.2.

```
AES(seed, counter) -> Random number 1
AES(seed, counter + 1) -> Random number 2
...
AES(seed, counter + n) -> Random number n+1
\label{ls:rng}
```

Listing 3.2: Steps for generating random numbers using AES in CTR mode.

Importantly, the output of AES encryption in CTR mode is deterministic (i.e., the same counter and seed produce the same output.), but without knowing the seed or the counter, the output appears random. The PRNG for AES is implemented in `PRNG::InitSeed()` function to generate a random seed, which is then used to generate new numbers. The pseudocode for PRNG is shown in Listing 3.3.

```
Class PRNG:
    Variables:
```

```
        state          // Internal state array for random data
        cnt            // Counter to track the current position in the state
        initialized    // Boolean to check if PRNG is initialized
        seed           // Seed for initializing the state
        KeySchedule    // AES key schedule for encryption

    Function InitSeed(seed):
        initialized = true
        If AES is enabled:
            aes_schedule(KeySchedule, seed)  // Setup AES key schedule
            state = [0] * RAND_SIZE          // Initialize state to zeros
            For i = 0 to PIPELINES - 1:
                state[i * AES_BLK_SIZE] = i  // Initialize pipeline index
        Else:
            state = seed                     // Directly copy seed to state
        Call hash()                          // Mix the state

    Function hash():
        For each block in state:
            Encrypt block using AES with KeySchedule
        Reset counter (cnt = 0)

    Function get_bytes(buffer, length):
        i = 0
        While i < length:
            If cnt == RAND_SIZE:             // If state is exhausted
                Call hash()                  // Regenerate state
            buffer[i] = state[cnt]           // Fetch random byte
            cnt += 1
            i += 1

    Function get_random_number(length):
        buffer = Allocate array of size length
        Call get_bytes(buffer, length)       // Fill buffer with random bytes
        Return buffer
End Class
```

Listing 3.3: Pseudocode for PRNG in MASCOT.

In the QFL framework, we enhance this process by replacing `PRNG::InitSeed()` with a call to the KMS. Although QRNG can operate as standalone components, providing direct access to random numbers without requiring network connections, their integration into a KMS greatly enhances security through centralized control and improved scalability. This approach is particularly beneficial in large-scale or multiparty systems, where integrating QRNG into the network minimizes infrastructure needs compared to individual devices generating their own randomness. The next step is the incorporation of symmetric and oblivious keys with KMS. To incorporate symmetric keys, we developed the `gen_symmetric_random_key()` method in the `server_example.c` file of the ETSI library [135]. Symmetric keys are essential for AES, which operates with fixed key sizes, typically 128 bits, 192 bits, or 256 bits. To generate an AES key randomly, MASCOT uses PRNG. Once the key is generated, it is distributed among parties using secret sharing techniques. In the QFL framework, we replaced this traditional approach by leveraging quantum-generated symmetric keys through QKD, in compliance with the ETSI 004 standard, offering improved security. The `gen_symmetric_random_key()` method first computes a unique seed value based on the source

and destination Uniform Resource Identifiers (URIs) and the index. The seed is calculated by applying a simple hash function (`simple_hash`) to the source, destination, and index, and combining the results. This seed ensures that the generated key is unique for each pair of source and destination with a given index. The function then initializes the random number generator with the computed seed using `srand (seed)`. Following this, it generates a symmetric key by producing random bytes (from 0 to 255) for the specified key size (`key_size`). The generated key is stored in a dynamically allocated buffer (`key.data`), and the function returns this key. The pseudocode for symmetric key generation is shown in Listing 3.4.

```
FUNCTION gen_pseudorandom_key(source, destination, index, key_size):
    # Step 1: Compute a unique seed
    seed = simple_hash(source) XOR simple_hash(destination) XOR index

    # Step 2: Initialize the random number generator
    srand(seed)

    # Step 3: Generate the pseudorandom key
    key = []  # Initialize an empty array for the key
    FOR i FROM 0 TO key_size - 1:
        key.append(rand() MOD 256)  # Generate a random byte and append to key

    # Step 4: Return the generated key
    RETURN key
```

Listing 3.4: Pseudocode to generate random symmetric keys.

In the next step, we incorporated oblivious keys into the KMS by developing a method called `gen_oblivious_random_key()` [135], in which oblivious keys are retrieved from QOKD protocol. This method follows a similar approach to `gen_symmetric_random_key()`. However, it involves an additional step for modifying the key based on an `oblivious_key_type` parameter. Within this method, if the `oblivious_key_type` is equal to 2, the function applies a series of bitwise operations to the generated key. Specifically, it alternates between modifying even-indexed bytes with the value $0xAA$ and odd-indexed bytes with the value $0x55$ after shifting the bits. These operations introduce an additional level of obfuscation, making the key more oblivious in nature. The modified key is then returned. The pseudocode for oblivious key generation is shown in Listing 3.5.

```
FUNCTION gen_oblivious_pseudorandom_key(source, destination, index, key_size,
    oblivious_key_type):
    # Step 1: Compute a unique seed
    seed = simple_hash(source) XOR simple_hash(destination) XOR index

    # Step 2: Initialize the random number generator
    srand(seed)

    # Step 3: Generate the pseudorandom key
    key = []  # Initialize an empty array for the key
    FOR i FROM 0 TO key_size - 1:
        key.append(rand() MOD 256)  # Generate a random byte and append to key

    # Step 4: Apply oblivious transformations if required
    IF oblivious_key_type == 2:
        FOR i FROM 0 TO key_size - 1:
            IF i MOD 2 == 0:
                key[i] = key[i] XOR 0xAA  # Modify even-indexed bytes
```

```
            ELSE:
                key[i] = (key[i] >> 1) XOR 0x55  # Modify odd-indexed bytes

    # Step 5: Return the generated key
    RETURN key
```

Listing 3.5: Pseudocode to generate random oblivious keys.

After incorporation of quantum keys with the KMS, the program sends an `OPEN_CONNECT` request to the KMS, receiving a Key Stream ID (KSID) in return for each party involved in the computation. Once the setup is complete, the MASCOT protocol is triggered, and the computation begins. During this stage, the program sends `GET_KEY` requests to retrieve the necessary random numbers, oblivious, and symmetric keys. These keys are then provided to the MASCOT protocol, which uses them in conjunction with the QOT protocol to enable secure oblivious transfer and OT extension. After the computation is finished and the results are obtained, the program sends a `CLOSE` request to finalize the session with the KMS. As the OT generation in the MASCOT is handled through the `BaseOT::exec_base()` method in the `OT/BaseOT.cpp` [98], we modified this method to incorporate quantum-generated OTs from the QOT. The pseudocode for the modified `BaseOT` is shown in Listing 3.6 which utilizes the oblivious keys provided by the KMS to enable quantum-generated OT. Notably, in this context, the QOT protocol can be applied in two distinct ways, depending on the available oblivious key rate. First, it can serve as a base OT protocol within an OT extension framework, where it minimizes the need for new oblivious keys by extending a smaller set to meet the protocol's requirements. Alternatively, QOT can be employed as a stand-alone solution, fully replacing other OT implementations in MASCOT. When the quantity of generated oblivious keys is substantially less than the total number of required OTs, integrating QOT with OT extension is the preferred approach, optimizing key usage efficiency. Conversely, if the number of oblivious keys is sufficient, QOT can be directly applied without extension, thereby reducing computational and communication overhead.

```
class BaseOT:
    def __init__(self, k: int, n: int):
        self.k = k  # Security parameter
        self.n = n  # Number of messages
        self.private_key = None
        self.public_key = None

    def keygen(self):
        """
        Generate key pair for the protocol.
        """
        self.private_key = generate_private_key(self.k)
        self.public_key = derive_public_key(self.private_key)

    def sender_input(self, messages):
        """
        Sender inputs n messages.
        """
        assert len(messages) == self.n
        self.messages = messages

    def receiver_input(self, indices):
        """
        Receiver inputs n selection indices.
```

```
        """
        assert len(indices) == self.n
        self.indices = indices

    def transfer(self):
        """
        Execute the transfer protocol.
        """
        encrypted_messages = encrypt(self.messages, self.public_key)
        selected_messages = [
            encrypted_messages[i] for i in self.indices
        ]
        return selected_messages
```

Listing 3.6: BaseOT Protocol Implementation

In the QSMC framework, we extend the number of OTs using Protocol 4. To this end, we exploit AES in counter mode to serve as the PRF required for the OT extension. The final step is to generate the arithmetic circuits that correspond to the functions under computation. We generated two arithmetic circuits, SRD and DSP, for the vehicular networks and drug discovery use cases, respectively. The details of these two use cases are provided in the next chapter.

The implementation code for the QSMC framework is available at the GitHub repository https://github.com/Quantum-SMC.

3.3 Framework Evaluation

In this section, we provide security and efficiency analyses of the proposed QSMC framework. The proposed approach provides unconditional security even against quantum computer attacks. By comparing this framework to its classical counterpart, we highlight the improvements in both security and performance, underscoring the potential of QSMC to enhance SMC practices.

3.3.1 Security Analysis

The QSMC framework is implemented with active security in the malicious adversarial model. In this model, dishonest parties may haphazardly drift from the protocol execution and try to cheat. Protocols that achieve security in the malicious adversary scenario have the highest security level, which means that the only thing that an adversary can do is to cause the honest parties to abort, but can never preclude the privacy of others.

The security of the proposed QSMC framework is fundamentally reinforced by the principles of quantum mechanics through the integration of QRNG, QKD, and QOKD. QKD ensures the generation and distribution of cryptographic keys with unconditional security, leveraging the Heisenberg Uncertainty Principle and the no-cloning theorem to detect any eavesdropping attempts and prevent key interception. QOT provides a robust method for secure data transfer, ensuring that neither party can gain complete knowledge of the other's data, safeguarded by the principles of quantum superposition and entanglement. QRNG enhances the platform's cryptographic strength by generating truly random numbers derived from quantum processes, ensuring unpredictability and eliminating biases inherent in classical random number generators.

While the security of the classical S-OT relies on the computational assumptions of parties, the quantum QOT is unconditionally secure. This is because the key generation in QOT is done using quantum technologies that can be implemented independently of the public-key infrastructure, making it secure against quantum computer attacks. The OT extension is also secure as it is based on symmetric cryptography.

3.3.2 Efficiency Analysis Based on OT

To evaluate our system, we compare the complexities of the quantum and classical protocols. In Tables 3.1 and 3.2, the computation and communication complexity of S-OT, QOT, and OT extension are represented. The values inside parenthesis refer to the cost of OT extension. By communication complexity, we mean the number of bits that are transferred among parties. The computational complexity is the amount of resources required to run the circuit, which particularly focuses on time and memory requirements.

We first consider the operations in the classical S-OT protocol (i.e. random number generation, modular exponentiation, hash evaluation, and encryption operations). We consider the cost of each operation as follows: Sampling a random number (RNG), Computing A or B (modular exponentiation), Computing $g^{ab} = A^b = B^a$ (modular exponentiation), Computing each key using the hash function (hashing), Encryption or decryption of each message (XOR). In total, the S-OT protocol requires two random numbers, five modular exponentiation, three hashing, and three XOR (two for applying the encryption on Alice's messages, and one for decryption of a message by Bob). Also, the communication cost of the S-OT is $2n$ bits for n executions of OT. As explained before, along with S-OT, MASCOT uses an actively secure OT extension protocol [62] that is based on to the passively secure protocol of [64]. The number of extended OTs depends on the number of multiplication triples that are necessary to compute the multiplication gates of the circuit. MASCOT requires 1408 OT extension for each multiplication triple, considering a 128 bit filed with the constant parameter $\tau = 3$ and the statistic and computational security parameters $\kappa = \lambda = 128$ (see section 7.1 of [8]). In [62], the communication and computation complexity of the OT extension protocol used in MASCOT are computed. The results show that OT extension requires $2n + 336$ hashing for computations, and $128n + 10$ *kbit* for communication purposes [62].

Table 3.1 represents the communication complexity of quantum and classical protocols. As it can be seen, the cost of QOT protocol is $3n$ which is always less than classical cost $2n + (128n + 10000)$, regardless of the value of n (number of OTs). Table 3.2 shows the computational complexity of quantum and classical protocols. To fairly compare the costs of the quantum and classic cases, we first need to consider the cost of each operation separately. For example, modular exponentiation is much more expensive than linear operations, such as bitwise XOR. One way to compare these operations is by calculating the execution time of them. Note that since the offline phase of protocols can be done in advance and independently of the user's private inputs, we only measure the execution times for operations in the online phase. Through the online phase, the S-OT requires $3n$ bitwise XOR, while the QOT needs $2n$ bitwise comparisons, $5n$ bitwise XORs, and $3n$ bitwise truncations. Moreover, the S-OT transfers $2n$ bits, while the QOT transfers $3n$ bits for communication during the online phase.

Table 3.1: Comparison of the communication complexities between quantum and classical approaches. n is the number of OTs.

	Quantum	**Classic**
	QOT	S-OT and OT extension
Bit sent	$3n$	$2n + (128n + 10000)$

Table 3.2: Computation complexity of the QOT and S-OT protocols. n is the number of OTs. The dash symbol "-" means the specified operation is not used in the protocol.

	Quantum	**Classic**
Operation	QOT	S-OT and OT extension
Bitwise comparison	$3n$	-
Bitwise truncation	$7n$	-
Bitwise XOR	$5n$	$3n$
RNG	$3n$	$2n$
Hash	-	$3n + (2n + 336)$
Modular exponentiation	-	$5n$
Quantum state preparation	n	-
Quantum state measurement	n	-

3.4 Final Remark

In this chapter, we proposed a QSMC framework based on three advanced quantum communication technologies known as QRNG, QKD and QOKD. To facilitate the management of keys, we developed a KMS for the distribution, storage, and synchronization of keys. The secure computation is managed by MASCOT protocol that offers active security against malicious parties. The QSMC framework facilitates the implementation of diverse SMC use cases across various domains, ranging from vehicular networks to healthcare systems.

Chapter 4

Quantum SMC Services

As the field matured, the application of SMC expanded beyond theoretical frameworks to practical implementations. Within this chapter, we outline the real-life applications of quantum-based SMC in two domains: vehicular networks and privacy-preserving drug discovery.

In Section 4.1, we exploit the proposed QSMC framework (see Chapter 3) to implement SRD use case in vehicular networks. Subsequently, in Section 4.2 we use QSMC in conjunction with FL to implement DSP use case to facilitate privacy-preserving drug discovery. Section 4.3 concludes the chapter.

4.0.1 Related Works

SMC has been widely explored in various domains, including vehicular networks and drug discovery, where privacy and security are paramount. For example, in the context of vehicular networks, in [80], a fully decentralized car-sharing system called Secure and Privacy-Enhancing Protocol for Car Access Provision (SePCAR) is proposed that operates without reliance on a central authority. This system enables secure and private vehicle sharing while safeguarding the data of both car owners and users. Similarly, in [9], an SMC-based protocol for location-based applications called Vehicle Privacy (VPriv) is presented. VPriv facilitates toll collection, speed ticket issuance, and insurance premium calculations without exposing vehicle location data to centralized servers. In [136], a decentralized and location-aware architecture is proposed to address the privacy-preserving issues in blockchain-based traffic management systems in vehicular networks. These solutions highlight the growing role of SMC in ensuring secure and privacy-preserving vehicular communications.

In the pharmaceutical industry, SMC has emerged as a key technology to address data privacy concerns in drug discovery. FL [137] has been widely adopted to enable collaborative model training while keeping data localized, mitigating privacy risks associated with centralized data storage. However, FL remains vulnerable to privacy leaks, as model parameters can inadvertently reveal information about the training data. To counteract this, Differential Privacy (DP) [49] has been employed to introduce controlled noise into model updates. Although effective in enhancing privacy, DP often results in reduced model accuracy due to the introduced perturbations. SMC provides an alternative that ensures strong privacy guarantees without compromising model accuracy. Recent research has demonstrated its efficacy in drug discovery applications [138]. For example, in [14], two novel SMC protocols to predict Drug–Target Interactions (DTI) and Quantitative Structure–Activity Relationships

(QSAR) is introduced using secret-sharing techniques. DTI involves identifying potential interactions between a drug molecule and a specific biological target, such as a protein, which is essential for assessing a drug's efficacy and safety. QSAR, on the other hand, predicts the biological activity of chemical compounds based on their chemical structure and provides insights into how new compounds might behave in biological systems. Additionally, in [82] it is described how multiple pharmaceutical companies participating in the Machine Learning Ledger Orchestration for Drug Discovery (MELLODDY) project used SMC in combination with FL to enhance their classification and regression models. Within MELLODDY project, companies leveraged large datasets containing over 21 million small molecules. Moreover, in [139] a Multiparty Computation-Based Deep Learning Framework for Drug-Drug Interaction (MPCDDI) is proposed which employs secret-sharing techniques to securely aggregate drug-related feature data from multiple institutions, enhancing prediction accuracy. Additionally, in [140], an SMC-friendly framework known as Safe Federated Learning (SAFEFL) is proposed to perform secure FL on the Human Activity Recognition (HAR) dataset [141] for a classification task.

4.1 Vehicular Networks

Vehicular networks are communication platforms that support vehicles to share data among themselves, as well as with roadside infrastructure and other network components, using a combination of wireless communication technologies and protocols. The architecture of the communicating entities in the vehicular network is shown in Fig. 4.1. Vehicle-to-Vehicle (V2V) communication allows vehicles to directly exchange important data, such as speed, location, and direction, which is essential for safe driving. In Vehicle-to-Infrastructure (V2I) communication, vehicles interact with Roadside Unit (RSU) like traffic lights, signs, and toll booths for applications such as toll collection and the dissemination of road condition information. Additionally, Vehicle-to-Pedestrian (V2P) communication enables vehicles to connect with devices carried by pedestrians, such as smartphones, to enhance pedestrian safety. In practice, data transition among network entities is carried out through wireless communication networks such as WiFi, cellular networks (e.g. 5G), and satellite communication [142]. To encrypt data transmission among network entities, PKC protocols such as RSA are employed [143]. These protocols ensure that data exchanged between vehicles and other network participants is protected from unauthorized access. However, these protocols are both inefficient and vulnerable to quantum attacks, highlighting the need for more robust and efficient security measures in vehicular networks.

4.1.1 Use Case: Safe Route Departure

Changing lanes when exiting a highway is one of the main causes of heavy traffic and even sometimes large chain crashes [81]. We propose the SRD service that assists vehicles to switch lanes and exit the highway safely. We consider n numbers of vehicles $\{v_1, v_2, ..., v_p, ..., v_q, ..., v_n\}$ with locations $\{x_1, x_2, ..., x_n\}$ and velocities $\{s_1, s_2, ..., s_n\}$ in an m-lane highway and assume a subset of vehicles $\{v_p, ..., v_q\}$ intend to exist the highway. Therefore, they call for the SRD to help them through the exit process. In the first step, the service asks for the private information of all the vehicles close to the exit point (e.g. vehicles that are within a radius of 5 km of the exit location). The private inputs for each vehicle v_i in the proposed service are location x_i, velocity s_i, lane number l_i, and the exit intention b_i. The exit intention is a Boolean flag

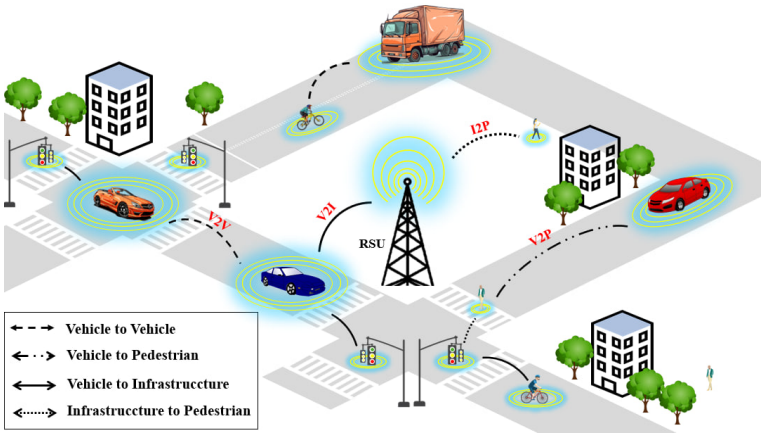


Figure 4.1: An overview of communicating entities in a vehicular network.

b such that 0 represents no exit intention and 1 represents the intention to exit. Therefore, when a vehicle v_i intends to exit the highway, it changes its Boolean flag b_i to 1 and subsequently the protocol is activated. These inputs are then used to compute the proper times for vehicles that intend to exit the highway, considering the density of neighboring vehicles.

Given the exit location x_e as a constant parameter and the private location of the i-th vehicle (x_i), the service computes the distance of each vehicle to the exit ($\Delta x_i = |x_e - x_i|$). As we are considering the highway as a straight path, the distance calculation can be done in one dimension. Note that this is not an unrealistic assumption as we only consider vehicles close to the exit point. Using the distance Δx_i and considering the formula $\Delta x = st$, the service computes the time that each vehicle takes to reach the exit point x_e. Considering the time difference between vehicles, the service evaluates the density of the neighboring vehicles and computes the proper times for the desired vehicles to exit the highway. Through this service, each vehicle is supposed to gradually reduce its lane number step by step to decrease its distance from the exit point. Therefore, the service first checks the lane number l_i of the vehicles intending to exit. If l_i is equal to the exit lane, only one step calculation of $t_i = \Delta x_i / s_i$ is enough to compute the proper time. Otherwise, vehicles have to reduce their lane number to reach the exit lane, and for each lane change, a pair (t_c, l_c) is provided by the service. To compute the pair, the service only considers vehicles in the adjacent lane (l_c). Then, it computes the time difference Δt between the desired vehicle and the next approaching vehicle in the adjacent lane. The computation is continued until Δt reaches a value that is long enough for a vehicle to change its lane (e.g. 30 seconds). Using the same strategy, vehicles change their lanes until they reach the exit lane. Note that, in case the highway was crowded and there was no proper time for a vehicle to change lane, a message would be sent to the approaching vehicles asking them to adjust their speed and provide empty space for the desired vehicle. After performing computation, the service provides the number of exiting vehicles as well as their approximate departure time, to the whole network. These outputs increase safety throughout the network, as vehicles are aware of all the ongoing events around them. The proposed service provides the following outputs:

- Each exiting vehicle receives its unique pair (t_c, l_c), for each lane change.
- The total number of exiting vehicles n_v, as well as their approximate exit time is provided to the whole network, to increase safety.

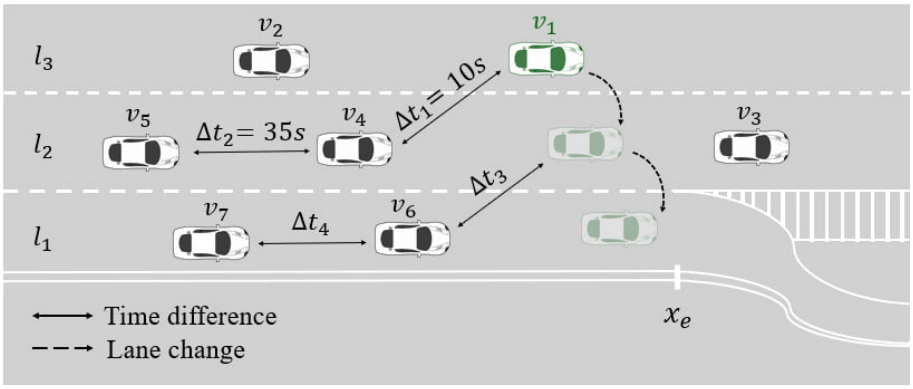


Figure 4.2: Example of the proposed service on a 3-lane highway with seven vehicles, where vehicle v_1 intends to exit. v_i represents each vehicle, Δt_i indicates the time difference between vehicles, l_i is the lane number, and x_e is the exit location. The Δt_i values are hypothetical for illustration purposes.

SRD service can be structured as $f(x_1, v_1, l_1, b_1; ...; x_n, v_n, l_n, b_n) = (n_v, (t_c, l_c)_1, ..., (t_c, l_c)_v)$, where each party provides its private data (location, speed, lane number, exit intention) as inputs and receive the corresponding outputs. Note that the outputs could be announced either publicly or could be sent only to specific vehicles, in order to be kept private. In the SRD some outputs are public while the others are sent only to specific vehicles in an encrypted way.

Figure 4.2 illustrates a simplified example of the proposed service. Suppose that seven vehicles are driving on a highway with three lanes. One of the vehicles, v_1, in lane three (l_3) intends to exit the road, while the others continue on their paths. The service computes the time difference between v_1 and v_4 which is the nearest approaching vehicle in l_2, and obtains $\Delta t_1 = 10$ seconds. Since Δt_1 is too short, the computation is repeated for v_4 and v_5, and the result is $\Delta t_2 = 35$ seconds, which is considered long enough for v_1 to switch to the adjacent lane l_2. At this stage, a message with the content *"Please reduce your speed to ... and switch to l_2 after 10 seconds."* is sent to v_1. The same strategy is applied until v_1 switches to l_1 and exits the highway. Additionally, the other vehicles receive a message with the content *"A vehicle is exiting the highway at ..."*. Note that, through the whole computation process, the vehicles' private information (location, velocity, lane number, exit intention) is not revealed to the other vehicles.

SRD Arithmetic Circuit

To perform secure computation among vehicles using MASCOT, we need to generate an arithmetic circuit for the SRD service. As the proposed service primarily involves arithmetic operations like addition, utilizing MASCOT protocol is highly efficient. A schematic representation for SRD arithmetic circuit involving arithmetic gates is shown in Fig. 4.3. Once the circuit is defined, it must be programmed for integration into the QSMC framework. To achieve this, a high-level Python-like language provided by MP-SPDZ is utilized. The circuit is programmed in a file named `SRD.mpc`, which is located in the `Programs/Source` directory of the QSMC framework. The pseudocode for SRD circuit is shown in Listing 4.1. Finally, by running the program as outlined in the GitHub repository https://github.com/Quantum-

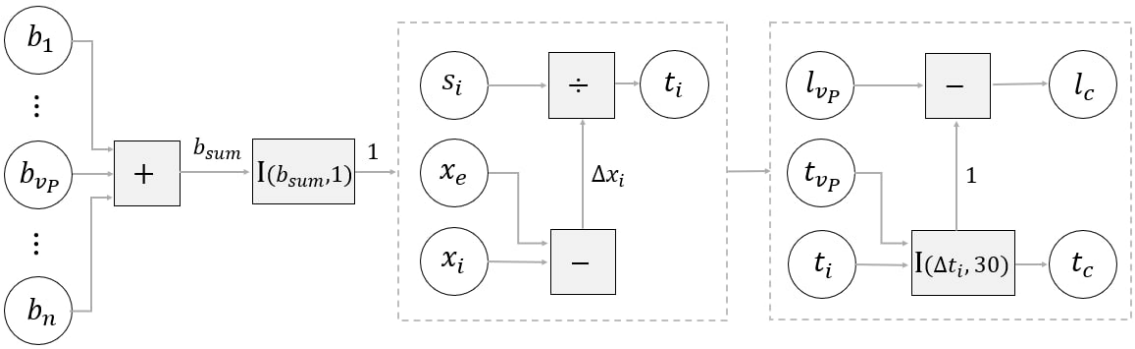


Figure 4.3: A schematic representation for SRD arithmetic circuit. $+$, $-$, $\div$, and I represent addition, subtraction, division, and comparison gates, respectively.

SMC, secure computation among vehicles is executed, and the output of the computation $((t_c, l_c), n_v)$ is generated.

```
for each vehicle i:
  if vehicle.lane[i] == 3:
    if lane 2 is empty:
      print("vehicle", i + 1, "switches to lane 2 now")
      vehicle.lane[i] -= 1
    else:
      for each duration in lane 2:
        if timeDifference >= requiredTimeToChangeLane:
          Print("vehicle" i+1 "switches to lane 2 after X seconds")
          Update vehicle duration
          vehicle.lane[i] -= 1
          break
  else if vehicle.lane[i] == 2:
    if lane 1 is empty:
      Print("vehicle" i+1 "switches to lane 1 now")
      vehicle.lane[i] -= 1
    else:
      for each duration in lane 1:
        if timeDifference >= requiredTimeToChangeLane:
          Print("vehicle" i+1 "switches to lane 1 after X seconds")
          Update vehicle duration
          vehicle.lane[i] -= 1
          break
  else if vehicle.lane[i] == 1:
    Print("vehicle" i+1 "exits the highway after X seconds")
```

Listing 4.1: Pseudocode for SRD arithmetic circuit.

Results and Discussion

In this section, we analyze the complexity by calculating the communication and computation costs for the SRD use case. To this end, we first need to determine the number of OTs required for its circuit execution. For each vehicle, we need to perform one division gate (to compute the relationship $t = \Delta x/s$) and two comparison gates, requiring a total of 5 division operations. According to MASCOT protocol, each division/multiplication requires

2 multiplication triples, and each triple requires 128 base OT as well as 1408 OT extension which leads to

$$n = 10\,(128 \text{ OT} + 1408 \text{ OT extension}). \tag{4.1}$$

Having the required number of OTs, and considering Table 3.1, the amount of bit sent for quantum and classical protocols are

$$\text{Bit sent}_Q = 3n, \tag{4.2}$$

and

$$\text{Bit sent}_C = 2n + 128n. \tag{4.3}$$

We calculate the amount of the transferred bits for the quantum and classical protocols, considering different numbers of vehicles n_p. Figure 4.4 illustrates the amount of data transferred for $n_p = 2, 4, ..., 10$. For instance, with two vehicles on the highway, the communication costs are 5.6 *kbit* for the quantum protocol and 742.2 *kbit* for the classical protocol. As the number of vehicles increases from 2 to 10, the total communication costs rise to 28 *kbit* for the quantum protocol and 3670.8 *kbit* for the classical protocol. On average, the communication cost using the quantum protocol is reduced by 97% compared to the classical protocol. This value is obtained by separately comparing the communication costs of the quantum and classical protocols for each vehicle count and then averaging the results.

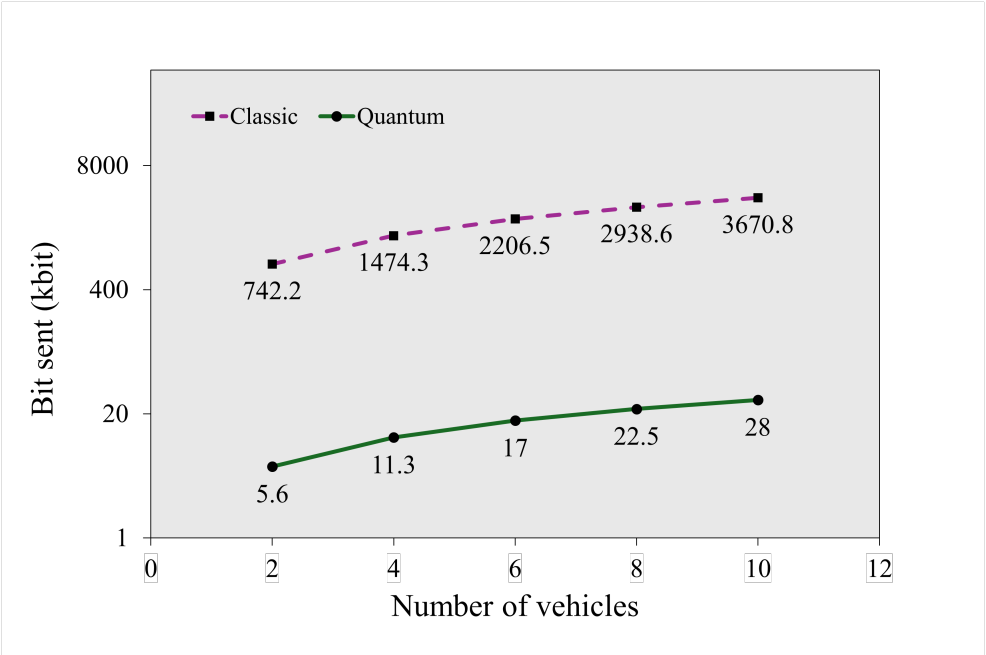


Figure 4.4: The communication cost of the classic and quantum models for different numbers of vehicles. The results are plotted on a logarithmic scale to account for the significant difference in cost values between the two approaches.

To compute the runtime of the SRD service, considering the required number of OTs, we measured the execution time for each operation listed in Table 3.2. As shown in Fig. 4.5, the execution time for quantum is higher than that of the classic by 42%. The runtime values are derived by averaging five measurements for each vehicle count n_p. The substantial

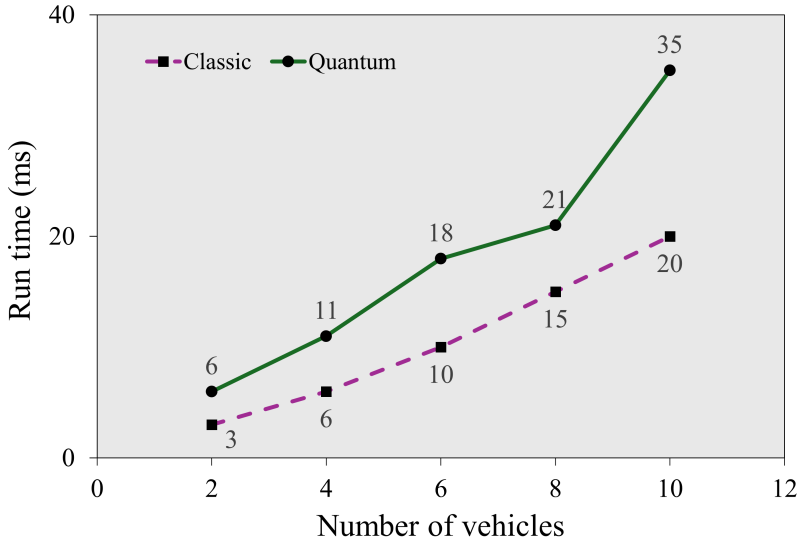


Figure 4.5: Run times of the classic and quantum models across different numbers of vehicles.

reduction in communication cost is particularly valuable for vehicular network applications, where the available spectrum for radio channels is a constrained resource. The results are obtained using C++'s ¡chrono¿ library with high_resolution_clock on an ASUS Zenbook 14 UX425E laptop running Ubuntu 20.04 (64-bit). The machine features an 11th Gen Intel(R) Core(TM) i7-1165G7 processor (2.80 GHz), 4 cores, and 16 GB of RAM.

Limitations and Challenges

We use the proposed QSMC platform to secure the communication among vehicles in SRD service. As mentioned before, we use the QKD protocol to generate quantum-based symmetric keys. Given the need for long-range communication in SRD, DV-QKD is a more suitable option due to its robustness to high mobility and signal losses, common in dynamic vehicular environments. Its proven security models, such as the BB84 protocol with decoy states, offer strong protection against eavesdropping, ensuring the integrity of sensitive data exchanged in-vehicle networks. In vehicular network applications, vehicles transfer data through wireless communications because a physical connection cannot be established among them. Terahertz QKD has recently been demonstrated to be a viable solution for wireless mobile applications [144, 145]. Satellite-based QKD is also a promising approach to transfer secure keys among vehicles in mobile networks [146, 147, 148]. However, a primary approach could be the use of QKD through optical fibers. The advantages of implementing QKD through optical fiber over wireless communications include improved reliability, lower cost, greater bandwidth, and higher security. As an example of a real-life situation, we can pre-share the keys using QKD, when the vehicles are being electrically charged.

4.2 Drug Discovery

Drug discovery involves identifying and developing new pharmaceutical compounds to treat diseases. The process typically spans over a decade and requires significant financial investment, often totaling billions of dollars, before a potential drug candidate can be brought to market [149]. Drug discovery typically encompasses the following steps:

- *Target Identification and Validation:* involves identifying a molecular target, such as a protein or nucleic acid, that plays a key role in a disease process. Once identified, the target must be validated to ensure that modulating its activity will lead to the desired therapeutic effect.
- *Lead Discovery (Hit Identification):* involves screening extensive molecular libraries to identify compounds that effectively interact with the target of interest. These compounds are known as hits and serve as starting points for further optimization.
- *Lead Optimization:* involves modifying the leads to improve their potency, selectivity, and other pharmacological properties while minimizing potential side effects.
- *Preclinical Development:* involves preclinical testing of leads to assess their safety, efficacy, pharmacokinetics, and toxicity in vitro (in cells) and in vivo (in animal models). Preclinical study data guide decisions on whether a candidate compound should progress to clinical trials.
- *Clinical Development:* involves testing candidate drugs in humans through a series of clinical trials as follows:
  - *Phase 1:* determining the safety profile, pharmacokinetics, and dosage of the drug in a small number of healthy individuals.
  - *Phase 2:* assessing the drug's effectiveness and further safety in a larger number of individuals who have the condition being targeted.
  - *Phase 3:* checking efficacy, observing adverse reactions, and make a comparison with the existing treatments in large-scale clinical trials.
  - *Phase 4:* post-market monitoring of the long-term safety and efficacy of the drug in a larger patient population.
- *Regulatory Approval:* involves submiting a New Drug Application (NDA) or Biologics License Application (BLA) to regulatory agencies, such as the Food and Drug Administration (FDA) in the United States or the European Medicines Agency (EMA) in Europe, seeking approval to market the drug.
- *Lifecycle Management:* involves optimizing formulations or refining manufacturing processes to improve the drug's market competitiveness and enhance patient outcomes.

4.2.1 Use Case: Drug Solubility Prediction

Solubility is a measure of how much of a substance (the solute) can dissolve in a solvent at a given temperature and pressure to form a solution. It's usually expressed as the maximum concentration of the solute that can be achieved before the solution becomes saturated. Solubility prediction in drugs is important as poor solubility can limit the absorption of a drug

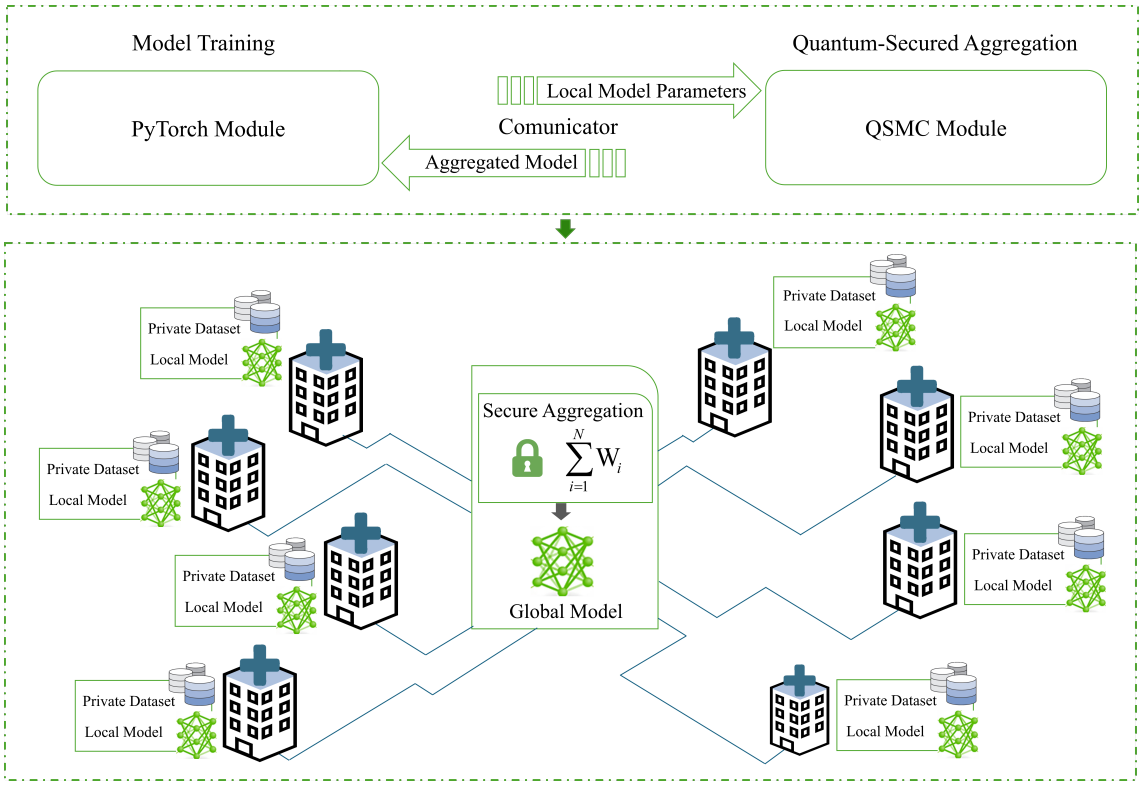


Figure 4.6: The architecture of the proposed QFL framework for the DSP use case.

into the bloodstream, reducing its efficacy. Additionally, solubility affects how a drug can be formulated for delivery (e.g., as a tablet, capsule, or injectable solution). We propose the DSP use case where multiple pharmaceutical companies collaboratively train a GCN model on their private datasets to predict the aqueous solubility of drug molecules. To implement the DSP use case, we develop a QFL platform that consists of two modules: the PyTorch and the QSMC, as illustrated in Fig. 4.6. The PyTorch module is responsible for model training and the QSMC is responsible for quantum-secured aggregation of models. In the first step, using the PyTorch module, each company locally trains a GCN model using its private dataset. After training, a communicator interface [140] is deployed to securely transfer secret shares of locally trained models from PyTorch to QSMC. This interface employs Secure Sockets Layer (SSL) to establish encrypted communication channels. Furthermore, by utilizing secret sharing techniques, the actual model parameters are split into multiple shares distributed across the QSMC module. As a result, even if SSL encryption is compromised, the intercepted data remains unintelligible unless a majority of the shares are also compromised. To enable this functionality, the QSMC module is configured to listen on a designated port, accepting incoming connections from the PyTorch interface. Rather than creating individual connections for each party, we configure PyTorch to operate as a single entity, managing and sharing the locally trained model parameters of all parties with the QSMC. This approach simplifies the setup but can be adapted to allow separate connections for each party. The QSMC module then performs model aggregation using MASCOT protocol and returns the aggregated model to PyTorch for the next training iteration. This process ensures that neither the server nor any other party can access the individual model parameters while still

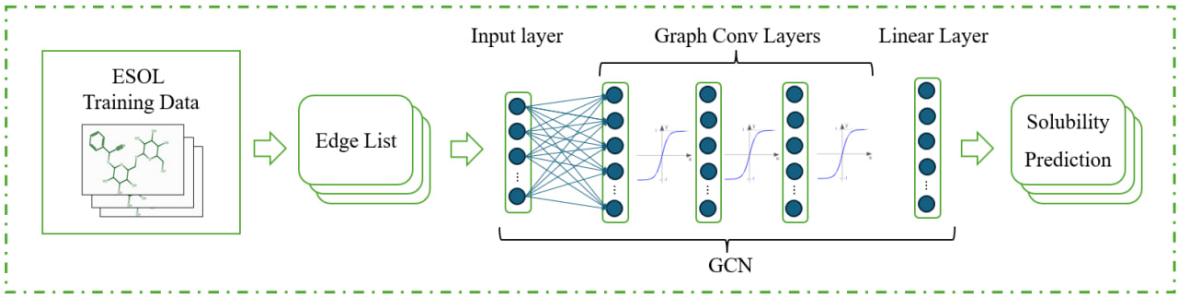


Figure 4.7: Architecture of the GCN model for drug solubility prediction.

allowing the computation of the aggregated model. The aggregated global model is then sent back to each participating company, ensuring that they have a GCN model with enhanced predictive performance.

PyTorch Module

The QSMC module is explained in Chapter 3. In the following, we explain the PyTorch module. This PyTorch module is responsible for performing the local model training on behalf of the parties. Within this module, we employ a GCN model [150] to predict the solubility of chemical compounds along with the ESOL dataset [151], a widely used benchmark for estimating drug solubility.

Gragh Convolutional Networks

A neural network is a machine learning algorithm designed to simulate the human perception, using multiple feature extraction layers. Each layer has some nodes that are connected to the nodes of the next layer through edges that have associated weights. The raw data is fed to the input layer, and the intermediate layers extract features, and the last layer gives the final prediction score or data class. Weights in the network determine the strength and direction of the connections between nodes. Initially, these weights are set to random values. During training, the network learns by adjusting these weights to minimize the error in its predictions. This learning process involves multiple iterations, where the network repeatedly processes the input data and updates the weights based on the computed error. This process is known as forward and backward propagation. In forward path, the input samples are fed to the model, and the output is generated. The error between the predicted output and the actual target value is then calculated using a loss function. In backward propagation, this error is propagated back through the network, and the weights are adjusted using a method called gradient descent. This iterative process continues until the network's predictions are sufficiently accurate.

GCN is a type of neural network specifically designed to operate on graph-structured data. GCNs have become a powerful tool in drug discovery because they can effectively model molecular structures, which are naturally represented as graphs where atoms are considered nodes and bonds are seen as edges. Figure 4.7 demonstrates the deployed GCN [150] for DSP use case. The deployed GCN directly estimates the solubility of compounds based on their chemical structures. This task is formulated as a regression problem, where the model takes the Simplified Molecular Input Line Entry System (SMILES) strings as input and outputs the

solubility as a real value. As in PyTorch Geometric [152], the GCN Convolution (GCNConv) layer is mathematically defined by

$$x_i^{(k)} = \sum_{j \in N(i) \cup \{i\}} \frac{1}{\deg(i) \cdot \deg(j)} (W^T \cdot x_j^{(k-1)}) + \text{bias}, \tag{4.4}$$

where $x_i^{(k)}$ is the feature representation of node i at layer k, $N(i)$ is the set of neighboring nodes of node i, W is the weight matrix that transforms the node features, bias is a vector added to the output, and $\deg(i)$ is the degree of node i, which represents the number of edges connected to it. We employed a GCN with three layers of GCNConv, and each layer is followed by a hyperbolic tangent (tanh) activation function as

$$x^{(k)} = \tanh(x^{(k)}). \tag{4.5}$$

The third GCN layer produces the final output $x^{(3)}$. This output is then passed through a linear layer, represented by

$$y_{\text{pred}} = W_{\text{lin}} \cdot x^{(3)} + b_{\text{lin}}, \tag{4.6}$$

where W_{lin} is the weight matrix of the linear layer and b_{lin} is its bias. The input and output size of the GCN layers are represented in Table 4.1.

Table 4.1: Input and output size of the GCN layers.

Layer	Input shape	Output shape
Initial Graph Convolution	$[32, 9]$	$[32, 64]$
Graph Convolution 1	$[32, 64]$	$[32, 64]$
Graph Convolution 2	$[32, 64]$	$[32, 64]$
Graph Convolution 3	$[32, 64]$	$[32, 64]$
Global Pooling	$[32, 64]$	$[1, 128]$
Output Linear Layer	$[1, 128]$	$[1, 1]$

To evaluate the performance of the model, we use the MSE loss function, which measures the average squared difference between the predicted values y_{pred} and the target values y_{t}. The loss function role is to minimize the model's error in predicting the solubility of drug molecules based on the learned graph representations. The formulation for the MSE loss is given by

$$L_{\text{MSE}} = \frac{1}{N} \sum_{n=1}^{N} (y_{\text{pred},n} - y_{\text{t},n})^2, \tag{4.7}$$

where N is the number of samples.

ESOL Dataset

We use the ESOL dataset [151], which is widely utilized in cheminformatics and machine learning for predicting the solubility of chemical compounds in water. It comprises $1,128$ samples, providing detailed information on the solubility of various organic molecules. In

Table 4.2: A Few Samples from the ESOL dataset.

Compound ID	SMILES	Solubility (mol/L)	Weigh (g/mol)
Fenfuram	Cc1occc1C(=O)Nc2ccccc2	-3.3	201.225
Citral	CC(C)=CCCC(C)=CC(=O)	-2.06	152.237
Picene	c1ccc2c(c1)ccc3c2ccc4c5ccccc5ccc43	-7.87	278.354
Thiophene	c1ccsc1	-1.33	84.143
Benzothiazole	c2ccc1scnc1c2	-1.5	135.191

ESOL, each chemical compound is represented as a graph that encodes the molecule's connectivity and structure, where atoms serve as nodes and bonds as edges. ESOL contains node feature vectors that represent the properties of individual atoms within the molecule including atom type, atomic mass, atomic charges, hybridization, etc. In addition to node features, the ESOL dataset includes edge features that describe the type of chemical bond between connected atoms (e.g., single, double, or triple bonds) and also provides information on bond distances and angles. The graph structure is represented by adjacency matrices or edge lists, which define how atoms are connected through chemical bonds. Table 4.2 shows a few number of samples from the ESOL dataset.

ESOL dataset uses SMILES that is a widely used method for encoding molecular structures as strings. Consider the example of using a SMILES string "CCO" to represent ethanol. In this notation, "C" denotes carbon atoms, and "O" represents oxygen. The sequence encodes the molecular structure, where two carbon atoms are bonded together, with an oxygen atom attached to one of them, forming ethanol. In other words, this string is interpreted as a linear structure of Carbon-Carbon-Oxygen (C-C-O), reflecting the connectivity of the molecule. The graph construction phase involves creating three nodes to represent the atoms (C, C, O) and two edges to represent the bonds (C-C and C-O). Feature initialization assigns specific properties to these nodes and edges. Node features include atom types (C, C, O), hybridization states, and other atomic properties, while edge features denote the bond types (single bonds in this case). This molecular graph is then fed into a GCN, which processes these features through its layers. The GCN learns to map these features to make predictions about the molecule, such as its solubility or binding affinity.

QFL Implementation

As described, the QFL framework consists of PyTorch and QSMC modules. To implement the PyTorch module, we built upon the publicly available code of SAFEFL [140]. We extended their implementation by integrating the GCN model described in Section 4.2.1. Within our setup, an identical GCN model is distributed among parties and the QSMC module. Furthermore, the framework is extended to work with the ESOL dataset described in Section 4.2.1. In this work, we split the ESOL dataset among three pharmaceutical companies and the QSMC module to streamline the experimental setup. Each partition was unique, ensuring no overlap of samples between parties, while maintaining a consistent test set across all participants.

In our setup, the QSMC module fulfills a dual role. Firstly, it functions as a server

that independently trains its own GCN model locally. This allows it to compare the model parameters submitted by the participating parties with its own, thereby strengthening the reliability and trustworthiness of the aggregation process. To this end, exploiting FLTrust [153], the QSMC employs cosine similarity and ReLU-based clipping to detect and flag deviations from expected model parameters. This mechanism effectively safeguards against poisoning attacks, in which malicious participants might attempt to degrade the accuracy of the model or introduce backdoors for targeted manipulation.

The second role of QSMC is to perform secure aggregation using the MASCOT SMC protocol. To enable this functionality, we utilized the MASCOT implementation of the QMP-SPDZ library [131]. Building on this foundation, we deployed the arithmetic circuit provided in [140], which corresponds to the FLTrust aggregation rule. The circuit, scripted in a Python-like language, is placed in the `./Programs/Source` directory of the QSMC module within the QFL framework. The simplified pseudocode for the FLTrust arithmetic circuit is provided in Listing 4.2.

```
Input:
    - input[PARTIES][PARAM_NUM]: Parameter updates from each party.

# Step 1: Compute dot products and norms.
dot_product = [dot(input[i], input[PARTIES - 1]) for i in range(PARTIES - 1)]
norm = [compute_norm(input[i]) for i in range(PARTIES)]

# Step 2: Compute trust scores using ReLU.
trust_score = [ReLU(dot_product[i] / (norm[i] * norm[PARTIES - 1])) for i in
    range(PARTIES - 1)]

# Step 3: Adjust parameters by trust scores and aggregate.
for i in range(PARTIES - 1):
    input[i] = (trust_score[i] / norm[i]) * input[i]
global_model_update = sum(input) * (norm[PARTIES - 1] / sum(trust_score))

Return global_model_update
```

Listing 4.2: Pseudocode for the DSP arithmetic circuit implementing the FLTrust aggregation rule.

Results and Discussion

We designed an experimental setup involving three parties, each assigned 225 samples from the ESOL dataset. Additionally, 225 samples were allocated to the QSMC module to facilitate FLTrust aggregation. A total of 80% of the ESOL dataset was used for training, while the remaining 20% was reserved for testing. The GCN model was trained over 500 local iterations, accompanied by 10 global iterations in the FL setup. The hyperparameters used are summarized in Table 4.3. For the training process, PyTorch Geometric was employed for GCN implementation, and RDKit was used for processing molecular data.

Figure 4.8 shows the loss function for the three parties and the QSMC, decreasing over the training process and reaching values of 0.0105, 0.0030, 0.0003, and 0.0046, indicating improved model performance. To assess the performance of the QFL framework, we conducted a solubility test on the global GCN model using the test samples. The results are depicted in Fig. 4.9, showing the predicted versus actual solubility values. Points closely aligned with the diagonal line indicate higher accuracy and demonstrate the model's precision. Moreover,

Table 4.3: Parameters used in QFL for drug solubility prediction.

Parameters	Value	Description
# parties	3	Pharmaceutical companies involved.
# malicious parties	1	Parties attempting to poison the data.
Local iterations	500	Training iterations performed by each party locally before sending updates.
Global iterations	10	FL rounds between the parties and the QSMC module.
Learning rate	0.0007	Step size for updating model weights during training.
Batch size	64	Samples used in each training epoch.
Size of QSMC dataset	225	Samples used by QSMC for FLTrust aggregation.
Size of each party dataset	225	Samples assigned to each party for local training.

the MSE reached values of 1.2. MSE measures the average squared difference between the estimated and actual values, offering a straightforward interpretation of the prediction accuracy. The communication cost for the QSMC module amounts to 3656.4 GB, contributing significantly to the overall system processing time. The total execution time is 3 hours and 55 minutes, with 99.7% of it specifically dedicated to QSMC execution.

Robustness Against Attacks

Federated learning frameworks, including QFL, are vulnerable to two types of attacks: privacy inference attacks [154] and poisoning attacks [155]. In privacy inference attacks, a malicious adversary who corrupts the model aggregator seeks to extract sensitive information about parties' private data from the updated local models or gradients [156]. Attackers may use statistical analysis, and model inversion techniques to infer details about the training data used by individual parties. To mitigate privacy inference attacks, we deployed the quantum-based SMC techniques to obscure sensitive information in aggregated updates. On the other hand, in poisoning attacks, malicious parties generate fraudulent models by intentionally injecting biased or manipulated data into the training process to compromise the integrity or performance of the global model. Poisoning attacks can aim to degrade model accuracy, introduce vulnerabilities (e.g., backdoors), or bias the model towards specific outcomes that benefit the attacker. To prevent poisoning attacks, we deployed the robust FLTrust aggregation rule, which either excludes potentially corrupt local models from the aggregation process or minimizes their impact using various scoring measures [157].

We tested the proposed QFL framework against four different types of poisoning attacks known as Krum [155], Trim [155], Min-Max [158], and Min-Sum [158]. Krum and Trim-Mean are Byzantine-robust aggregation rules aimed at mitigating adversarial behavior. Krum selects a local update closest to the majority by minimizing the sum of distances to other updates, while Trim-Mean discards a specified number of extreme values for each model parameter before averaging. Attackers exploit these rules by carefully crafting poisoned updates, with Krum attack ensuring malicious updates are close to each other to maximize deviation,

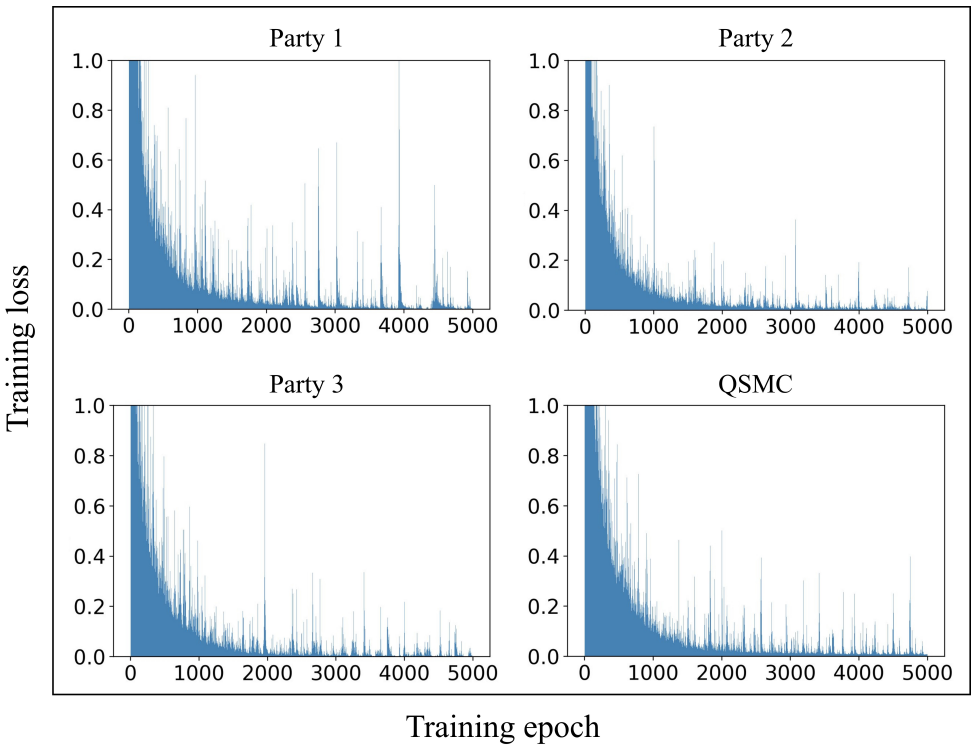


Figure 4.8: The loss function for parties and QSMC over 5000 iterations.

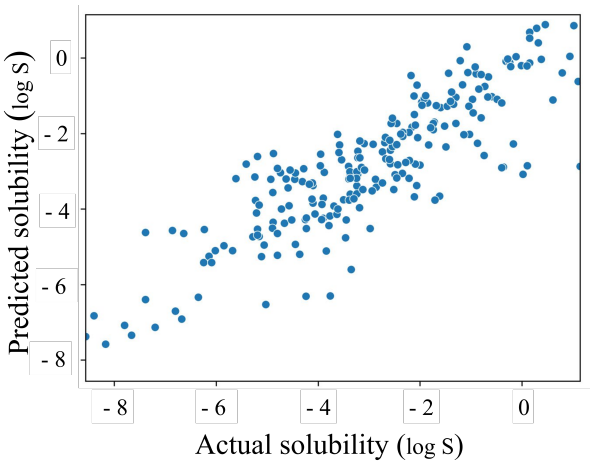


Figure 4.9: The predicted solubility for GCN using ESOL dataset.

Table 4.4: Error and accuracy evaluation for the QFL and SAFEFL frameworks under different attacks for the FLTrust aggregation rule. The first column shows the error evaluation (MSE) for the proposed QFL under different attacks for regression task (solubility prediction) on the ESOL dataset, based on our work. The second column shows the accuracy evaluation for the SAFEFL for classification task on the HAR dataset, with results taken from [140].

	QFL	SAFEFL [140]
Attack type	Regression task	Classification task
No attack	1.2	0.96
Krum [155]	1.3	0.94
Trim [155]	1.1	0.94
Min-Max [158]	1.1	0.92
Min-Sum [158]	1.2	0.94

and Trim attack manipulating values to influence the trimmed mean. Additionally, Min-Max and Min-Sum attacks introduce extreme values or minimize the cumulative effect of updates, aiming to distort the global model or hinder its convergence. These sophisticated attacks highlight the vulnerability of FL systems and the necessity of robust defense mechanisms.

We considered a scenario where up to 33% of the parties (i.e., one out of three) were maliciously corrupted. The MSE was calculated under different data poisoning attacks for the FLTrust aggregation rule. Our observations show that the QFL framework remains robust against these attacks, as the model continues to learn effectively despite the presence of malicious behavior. Table 4.4 compares our results with a similar study [140], in which a classification task was performed on the HAR dataset, containing human activity data (walking, sitting, lying, etc.) collected from smartphones of 30 participants. Their work considers 20% of malicious parties. Both approaches show rather consistent results across different attacks when using the FLTrust aggregation rule. However, a direct comparison of the values is not meaningful, as the task types and datasets used in the two studies differ.

All experiments are run on a machine with an ASUS TUF Dash F15 (FX516PR), featuring an 11th Gen Intel® Core™ i7-11370H processor (8 cores), 16 GB RAM, and running Ubuntu 24.04.1 LTS.

Limitations and Challenges

A key challenge of the QFL framework is the limited availability of quantum infrastructure. Although quantum technologies hold significant promise for enhancing privacy and security in the FL process, access to reliable quantum hardware remains a major barrier. Despite the high TRL of current quantum communication systems, they are still in experimental stages, preventing pharmaceutical companies from fully utilizing these technologies in practical applications. Another limitation is the unavailability of publicly accessible pharmaceutical datasets. As a result, a relatively small dataset had to be used, which was subsequently split across the different parties. The performance of machine learning models, including those based on GCNs, is highly dependent on the size and diversity of the training data. While the model is trained effectively, access to larger and more diverse datasets could further enhance its generalization and accuracy in predicting drug solubility. Another limitation is the computational and communicational complexity of using quantum-based SMC techniques for FL.

The total execution time is significantly impacted by the QSMC module. Efficiency could therefore be a limiting factor, and for very large datasets, utilizing a GPU may be necessary to train the model effectively.

4.3 Final Remark

Exploiting the proposed QSMC, we implemented two real-life use cases of SMC: SRD and DSP. In the context of the vehicular network, for the SRD, the communication cost is reduced by 97%, while the computation cost is increased by 42%, compared to classical systems. Additionally, Within the DSP use case, a GCN model was trained over 5000 iterations to predict the solubility of drug molecules. The global model obtained was tested on a test set, resulting in a MSE of 1.2. The framework was also evaluated against various poisoning attacks, with the model still learning effectively.

Chapter 5

Quantum SMC with Quantum Computing

In the previous chapters, we explored how quantum communication technologies such as QRNG, QKD, and QOKD can enhance the security and efficiency of SMC protocols. In this chapter, we explore how quantum computing can be harnessed to implement SMC, leveraging quantum properties such as entanglement to design novel protocols executable on real quantum computers.

In Section 5.1, we explain the related work. Subsequently, in Section 5.2, we introduce a novel quantum SMC protocol designed to compute Boolean functions utilizing the MBQC method. Subsequently, in Section 5.3, we expand upon this protocol to enable the computation of Boolean functions using single qubits. Section 5.4 concludes the chapter.

5.1 Related Works

A number of quantum-based SMC protocols have been proposed to carry out computations on various types of functions, such as Boolean functions [31, 68], polynomials [32], and arithmetic operations [69, 70, 34, 71]. These protocols typically leverage quantum entanglement, quantum measurements, and other quantum techniques to achieve secure computation. One notable example is the work by [30], which proposed a series of quantum schemes exploiting quantum entanglement in GHZ states to compute symmetric Boolean functions. This approach was later scrutinized in [159], where the authors disproved the security of one scheme, which claimed to achieve SMC for a dishonest majority with a threshold of $n-1$. To address this issue, an improved protocol based on the scheme in [30] was proposed in [160], which rectified the security flaws and enhanced efficiency. Additionally, in [47], a quantum-based SMC protocol was proposed to compute binary Boolean functions resorting to the MBQC technique. In addition to entangled particles, other approaches employed single qubits. Quantum SMC protocols based on entangled particles provide enhanced security and lower communication costs compared to single-qubit approaches. However, they require more quantum resources and specialized equipment for qubit entanglement. In contrast, single-qubit protocols are more resource-efficient but have higher communication overhead due to increased qubit exchange among parties, while maintaining the same computational complexity. Utilizing single qubits, in [31], authors suggested a new approach to compute pairwise AND function by employing single qubit measurements and linear classical computing. Additionally, in [68], a

further investigation of the work outlined in [31] was performed in which a single qubit is used to compute an n-tuple pairwise AND function. Moreover, in [48], a quantum-based SMC protocol to compute binary Boolean function was proposed utilizing single qubits.

5.2 SMC Using Measurement-Based Quantum Computing

In this section, we propose a quantum SMC protocol using the correlations of the GHZ state to compute binary Boolean functions, using MBQC. Our method introduces an additional random Z-phase rotation to the GHZ qubits to increase the protocol's security. Afterward, we implement the proposed scheme on the IBM QisKit platform and validate its feasibility through consistent and reliable results.

5.2.1 Secure NAND Computation

This section recalls the idea initially proposed by [161], to compute the universal NAND function using the entanglements of the GHZ state $|\text{GHZ}\rangle = (|001\rangle - |110\rangle)/\sqrt{2}$. The computation is accomplished in a secure manner which means that three parties (say, Alice, Bob, and Charlie) with input bits a, b, and c compute the NAND(a, b), while ensuring that no information about the individual inputs is disclosed to the other parties. Let us consider a scenario where the three qubits of the GHZ state are divided among three parties with each party holding one qubit of the entangled state. The parties measure the qubits in either σ_x or σ_y according to the input bits a, b, $c \in \{0, 1\}$. The third input is defined as $c = a \oplus b$. If the input bit is 0, they measure the qubit in σ_x basis, and if the bit is 1, the measurement is done in σ_y. There are four independent choices of inputs that form the following stabilizer equations for the GHZ state, initially outlined in [162] as

$$\begin{aligned}
\sigma_x \otimes \sigma_x \otimes \sigma_x \,|\text{GHZ}\rangle &= -\,|\text{GHZ}\rangle\,, \\
\sigma_x \otimes \sigma_y \otimes \sigma_y \,|\text{GHZ}\rangle &= -\,|\text{GHZ}\rangle\,, \\
\sigma_y \otimes \sigma_x \otimes \sigma_y \,|\text{GHZ}\rangle &= -\,|\text{GHZ}\rangle\,, \\
\sigma_y \otimes \sigma_y \otimes \sigma_x \,|\text{GHZ}\rangle &= +\,|\text{GHZ}\rangle\,.
\end{aligned} \tag{5.1}$$

The four equations can be expressed in a more concise way as

$$\sigma_a \otimes \sigma_b \otimes \sigma_{(a\oplus b)} \,|\text{GHZ}\rangle = (-1)^{\text{NAND}(a,b)} \,|\text{GHZ}\rangle\,. \tag{5.2}$$

Equation (5.2) implies that the output of NAND(a, b) is encoded into the eigenvalues of Eq. (5.1). If we assign the eigenvalues $+1$ and -1 with bit values 0 and 1, respectively, we obtain

$$\text{NAND}(a, b) = M_a \oplus M_b \oplus M_{(a\oplus b)}, \tag{5.3}$$

where $\oplus$ is addition modulo 2; M_a, M_b, and $M_{(a\oplus b)} \in \{0, 1\}$ are the measurement outcome of parties. This result implies that if three parties share the GHZ state and perform measurements determined by their inputs, the parity of their measurement outcomes is equal to NAND(a, b). In [163], the idea of [161] was expanded so that instead of using different measurement bases (σ_x and σ_y), parties can perform a pre-rotation operation to the GHZ qubits, based on the values of a, b, and $a \oplus b$. Afterward, by performing the σ_x measurement on the three qubits of the GHZ state, NAND(a, b) is computed. In other words, if we represent the $\pi/2$ rotation along the Z axis of the Bloch sphere by

$$U = R_z(\pi/2) = e^{-i\pi\,\sigma_z/4}, \tag{5.4}$$

then, performing the $U^\dagger$ rotation on the GHZ state will encode the parties' inputs in the resource state, leading to

$$|\psi\rangle = U^{\dagger a} U^{\dagger b} U^{\dagger (a \oplus b)} |\text{GHZ}\rangle \,. \tag{5.5}$$

The execution of the $U^\dagger$ operation on each qubit relies on the inputs of the respective parties (either $a, b, a \oplus b$). Specifically, if the input of a party is 1, the $U^\dagger$ operation is performed on the corresponding qubit. Conversely, if the input is 0, $U^\dagger$ is skipped for the corresponding qubit, resulting in the qubit retaining its initial state. This flexibility allows each party to decide whether or not to rotate its qubit based on the corresponding input. Afterward, by measuring the three qubits of the GHZ state in σ_x basis and performing XOR among the measurement results, NAND$(\vec{a}, \vec{b})$ is obtained as shown in Eq. (5.3).

5.2.2 Boolean Functions

In this section, we explain how binary Boolean functions can be computed using the secure NAND computation technique outlined in Section 5.2.1. We start by considering that any Boolean function $f(\vec{a}, \vec{b}) : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}$, which operates on two n-bit strings $\vec{a}$ and $\vec{b}$ as inputs and returns a single binary output, can be computed by taking the inner product of two vectors $P_i(\vec{a})$ and $K_i(\vec{b})$ as

$$f(\vec{a}, \vec{b}) = \bigoplus_{i=1}^{m} P_i(\vec{a}) \,.\, K_i(\vec{b}), \tag{5.6}$$

where $\vec{a} = (a_1, ..., a_n)$ and $\vec{b} = (b_1, ..., b_n)$ correspond to Alice's and Bob's input data, respectively; P_i represents polynomials depending on $\vec{a} \in \{0,1\}^n$ and K_i represents monomials depending on $\vec{b} \in \{0,1\}^n$ [164]. Equation (5.6) implies that to compute the Boolean function $f(\vec{a}, \vec{b})$, m series of AND operations are required. Therefore, by resorting to the secure NAND computation technique outlined in section 5.2.1 and subsequently converting it to an AND operation through a NOT operator, we can compute $f(\vec{a}, \vec{b})$. The maximum number of terms required for Eq. (5.6), denoted as m, is limited to 2^n, where n represents the input length.

According to Eq. (5.6), to compute $f(\vec{a}, \vec{b})$, three terms should be taken into account: $P_i(\vec{a})$, $K_i(\vec{b})$, and $P_i(\vec{a}).K_i(\vec{b})$. The first two polynomials $P_i(\vec{a})$ and $K_i(\vec{b})$ can be calculated locally by Alice and Bob, respectively. To compute the third term $P_i(\vec{a}).K_i(\vec{b})$, we use the scheme presented in Section 5.2.1. All that is needed is to find the result of NAND$\big(P_i(\vec{a}), K_i(\vec{b})\big)$ for each value of i. Afterward, we obtain $P_i(\vec{a}).K_i(\vec{b}) = \neg\text{NAND}(P_i(\vec{a}), K_i(\vec{b}))$ by performing a NOT.

Note that the particular form of polynomials in Eq. (5.6) depends on the Boolean function being evaluated. For example, let us obtain the polynomials that are required to compute OR$(\vec{a}, \vec{b})$. The 2-bit OR function can be represented as

$$OR(\vec{a}, \vec{b}) = \vec{a} + \vec{b} + \vec{a} \,.\, \vec{b} \,, \tag{5.7}$$

leading to

$$\begin{aligned}
OR(\vec{a},\vec{b}) =&(a_1\, OR\, b_1)\,.\,(a_2\, OR\, b_2)\\
=&(a_1+b_1+a_1.\,b_1)\,.\,(a_2+b_2+a_2.\,b_2)\\
=&a_1a_2+a_1b_2+a_1a_2b_2+b_1a_2+b_1b_2+b_1a_2b_2+a_1b_1a_2+a_1b_1b_2+a_1b_1a_2b_2\\
=&\underbrace{a_1a_2}_{P_1}.\underbrace{1}_{K_1}+\underbrace{(a_1+a_1a_2)}_{P_2}.\underbrace{b_2}_{K_2}+\underbrace{(1+a_1+a_2+a_1a_2)}_{P_3}.\underbrace{b_1b_2}_{K_3}+\underbrace{(a_2+a_1a_2)}_{P_4}.\underbrace{b_1}_{K_4}\\
=&\sum_{i=1}^{4}P_i(a_1,a_2).K_i(b_1,b_2). \qquad (5.8)
\end{aligned}$$

In equations (5.7) and (5.8), the symbols '+' and '.' represent the XOR and the logical AND, respectively [164]. Equation (5.8) indicates that a 2-bit OR$(\vec{a},\vec{b})$ function can be computed using the following vector of polynomials

$$P(\vec{a})=\begin{bmatrix} a_1a_2\\ a_1+a_1a_2\\ 1+a_1+a_2+a_1a_2\\ a_2+a_1a_2 \end{bmatrix},\quad K(\vec{b})=\begin{bmatrix} 1\\ b_2\\ b_1b_2\\ b_1 \end{bmatrix}. \qquad (5.9)$$

5.2.3 Boolean Function Computation using MBQC

Built upon the methodology outlined in [30], we introduce a quantum-based two-party protocol to calculate binary Boolean functions, by involving a third party. Using the proposed protocol, two parties, referred to as Alice and Bob can compute a binary Boolean function without disclosing any information about their private inputs. In [165], it was proven that attaining unconditionally secure two-party computations is not feasible. Consequently, the participation of a third party, referred to as Charlie, becomes necessary [165]. To address this security requirement, our protocol, originally designed for two-party scenarios, is extended to include the collaborative participation of Charlie.

In [30], a nearly private computation protocol (Scheme A) was proposed in which three parties share a GHZ state to perform SMC. Their scheme is described as "nearly private" because there are certain circumstances in which the third party, Charlie, can acquire information about the inputs of other participants. The primary source of information leakage in this scheme is that Charlie can simultaneously learn about the parity of Alice and Bob's inputs as well as the outcome of the protocol. The security of this protocol can be improved by preventing Charlie from gaining knowledge of the parity of Alice and Bob's private inputs, or the final output of the protocol, or both. We utilize a technique where we introduce an additional random Z-phase rotation on the GHZ qubits to obscure the outcome from Charlie. Using this technique, we reach a higher level of security, while using the same quantum resources and maintaining the existing complexity.

The proposed protocol advances through the following steps. First, Alice and Bob agree on the specific Boolean function and compute the required polynomial vectors P and K locally, using their inputs. Afterward, the protocol is executed over m rounds. The number of rounds corresponds to the size of polynomial vectors P and K. In each round i $(1 \leq i \leq m)$, the secure computation of the AND operation between P_i and K_i is carried out as follows. First, three qubits that form a GHZ state are distributed among the three parties.

Protocol 9 Quantum SMC Protocol Based on MBQC

Inputs: Input strings $\vec{a}$ for Alice and $\vec{b}$ for Bob.
Outputs: $f(\vec{a},\vec{b})$ for Alice and Bob.

1. Starting from $i=1$, repeat steps 2-9 for each term.
2. Given the particular function being computed, Alice and Bob locally calculate $P_i(\vec{a})$ and $K_i(\vec{b})$.
3. In order to generate a privately shared random bit r_i, a Bell state $|\varphi_i\rangle = (|00\rangle + |11\rangle)/\sqrt{2}$ is distributed between Alice and Bob. Subsequently, each measures a qubit of the Bell state and stores the result as r_i.
4. Alice and Bob send to Charlie the bit values $P_i(\vec{a}) \oplus r_i$ and $K_i(\vec{b}) \oplus r_i$, through a secure classical channel, respectively.
5. Charlie computes $(P_{i(\vec{a})} \oplus r_i) \oplus (K_{i(\vec{b})} \oplus r_i) = P_i(\vec{a}) \oplus K_i(\vec{b})$.
6. A three-qubit GHZ state is distributed among the parties as $|\mathrm{GHZ}_i\rangle = (|001\rangle - |110\rangle)/\sqrt{2}$.
7. Alice, Bob, and Charlie individually apply the operations ${\sigma_z}^{r_i} U^{\dagger P_i(\vec{a})}$, $U^{\dagger K_i(\vec{b})}$, and $U^{\dagger P_i(\vec{a}) \oplus K_i(\vec{b})}$ to their respective qubits in the GHZ state.
8. Next, parties measure their qubits in Pauli-X basis and store the measurement results m_{P_i}, m_{K_i}, and $m_{(P_i \oplus K_i)}$.
9. The three parties perform a NOT operator on the classical results to compute $\neg m_{P_i}$, $\neg m_{K_i}$, and $\neg m_{(P_i \oplus K_i)}$.
10. Once $i=m$, Alice and Bob individually perform the XOR operation on their measurement results ($M_1 = \bigoplus_{i=1}^{m} \neg m_{P_i}$ and $M_2 = \bigoplus_{i=1}^{m} \neg m_{K_i}$) and send them to Charlie.
11. Charlie then sums the XOR of Alice and Bob's outcomes with his own measurement results. He then reveals the value of $f'(\vec{a},\vec{b}) = M_1 \oplus M_2 \oplus M_3$, where $M_3 = \bigoplus_{i=1}^{m} \neg m_{(P_i \oplus K_i)}$.
12. Alice and Bob perform the last XOR operation to retrieve the result of the computation as $f(\vec{a},\vec{b}) = r \oplus f'(\vec{a},\vec{b})$, with $r = \bigoplus_{i=1}^{m} r_i$.

An intriguing characteristic of the GHZ state is that the computation can take place even when the qubits are located in different places, allowing for a secure computation among the distributed parties. Next, each party performs $U^\dagger$ (i.e. a $-\pi/2$ rotation around the Z axis of the Bloch sphere) on its qubit considering P_i, K_i, and $(P_i \oplus K_i)$, leading to

$$|\psi_i\rangle = U^{\dagger P_i} \otimes U^{\dagger K_i} \otimes U^{\dagger (P_i \oplus K_i)} |\mathrm{GHZ}_i\rangle \,. \tag{5.10}$$

Since all the information is encoded in the phase of the quantum state, performing an additional Pauli-Z rotation on one of the qubits will obscure the outcome of the computation. Therefore, if Alice and Bob intend to obscure the output from Charlie, one of them (say Alice) has to perform a Pauli-Z rotation on its qubit considering a random bit. To share a random bit, a Bell state $|\varphi_i\rangle = (|00\rangle + |11\rangle)/\sqrt{2}$ is distributed between the two parties. Afterward, each of them measures a qubit of the Bell states and stores the result in r_i. Next, Alice performs the Z-rotation on the first qubit leading to

$$\left|\psi_i'\right\rangle = {\sigma_z}^r \, U^{\dagger P_i} \otimes U^{\dagger K_i} \otimes U^{\dagger (P_i \oplus K_i)} \left|\mathrm{GHZ}_i\right\rangle . \tag{5.11}$$

If $r_i = 0$, the σ_z operator is not applied to the qubit, whereas if $r = 1$, σ_z is applied to the qubit. The additional Z rotation enhances security by concealing the protocol outcome from Charlie. However, Alice and Bob can easily decode the actual output by executing a bit-flip. In the next step, parties measure their qubit in Pauli-X basis and store the results in classical bits m_{P_i} , m_{K_i}, and $m_{(P_i \oplus K_i)}$. Subsequently, the parties apply a NOT operator to their results and proceed to the next round. Once the protocol is executed for m rounds, the three parties compute $M_1 = \bigoplus_{i=1}^{m} \neg m_{P_i}$, $M_2 = \bigoplus_{i=1}^{m} \neg m_{K_i}$ and $M_3 = \bigoplus_{i=1}^{m} \neg m_{(P_i \oplus K_i)}$.

Alice and Bob send their result to Charlie, who sums up all the classical bits as

$$f'(\vec{a}, \vec{b}) = M_1 \oplus M_2 \oplus M_3, \tag{5.12}$$

and sends the results to Alice and Bob. Following this, Alice and Bob derive the actual output by executing XOR between the random bit $r = \bigoplus_{i=1}^{m} r_i$ and the classical bit received from Charlie, as

$$f(\vec{a}, \vec{b}) = r \oplus f'(\vec{a}, \vec{b}). \tag{5.13}$$

As outlined in the protocol description, the demand for quantum resources increases with n, which means that as the length of the parties' input bits extends, a greater amount of quantum resources becomes necessary. Protocol 9 provides an overview of these procedures. In the next section, the computation of a 2-bit OR function is explained through an implementation in the IBM QisKit to illustrate the protocol.

5.2.4 QisKit Implementation

Universal fault-tolerant quantum computers are not available. Therefore, simulation platforms such as IBM QisKit [99] are employed for the design and implementation of quantum algorithms. QisKit is a software framework developed by IBM that enables users to simulate and execute quantum programs on both simulation platforms and real quantum computers. In this section, we design a circuit for the proposed protocol and explain its implementation in the QisKit IBM platform for a particular scenario involving a 2-bit OR$(\vec{a}, \vec{b})$ function [1].

Figure 5.1 depicts the quantum circuit implementing the proposed protocol. Circuit preparation includes three steps: A, B, and C. In the first step, A, a GHZ state is prepared starting from three qubits q_0, q_1, and q_2 with the initial state $|0\rangle$. In step B, the qubits are rotated according to the parties' private inputs and a random bit r. The rotational operations $R_z(\pi)$ and $R_z(-\pi/2)$ correspond to the V and $U^\dagger$ operations, respectively. Next, in step C, qubits

[1]The implementation code for the proposed quantum SMC protocol is accessible in GitHub repository "Quantum-SMC," located at https://github.com/Quantum-SMC.

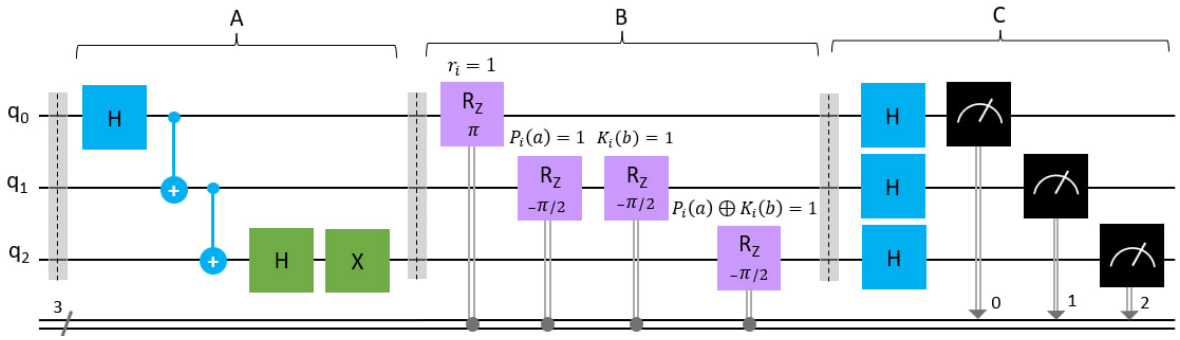


Figure 5.1: Quantum circuit for the proposed protocol in each round i. The q_0, q_1, and q_2 are three initial qubits with state $|0\rangle$. The label A represents the preparation of the GHZ state. Label B indicates the rotation of qubits with respect to the bits r, P_i, K_i, and $P_i \oplus K_i$. Note that the rotation gates in label B are exclusively applied when the bit values are equal to 1; otherwise, they are omitted from the circuit. Label C represents qubit measurements on the Hadamard basis. Labels 'H', '+', Z, 'X', and 'R_z' identify the Hadamard, controlled-X, Pauli-Z, Pauli-X, and Z-rotation gates, respectively.

---

are measured in the Hadamard basis. The default measurement in QisKit is performed in the Z-basis. However, by incorporating an H gate prior to the measurement operator, we can measure the qubit in the X-basis. The measurement result of each qubit is stored in a classical register of the QisKit environment. To store the measurement outcomes, three classical registers, C_0, C_1, and C_2, are used to store the measurement results of q_0, q_1, and q_2, respectively.

In our simulated experiments, we assume with no loss of generality that the three parties, Alice, Bob, and Charlie, with 2-bit inputs $\vec{a} = (1,0)$, $\vec{b} = (1,0)$, and $\vec{a} \oplus \vec{b} = (0,0)$ intend to compute $OR(\vec{a},\vec{b}) = OR\,(OR(1,1), OR(0,0))$, which yields to output 1. Alice and Bob share random bits $\mathbf{\vec{r}} = (0,1,1,0)$ leading to $r = \bigoplus_{i=1}^{4} r_i = 0$. Considering Eq. (5.8), Alice and Bob compute the polynomials $\vec{P} = (1,0,1,0)$ and $\vec{K} = (0,1,1,0)$, which correspond to the OR function. Figure 5.2, illustrates the measurement outcomes in four rounds of execution. For a three-qubit GHZ state, the possible measurement outcomes are 000, 001, 010, 011, 100, 101, 110, 111. Note that by performing measurement, the qubits collapse from a superposition state into a classical state with the highest probability (either 0 or 1). The eight potential outcomes occur with nearly equal probabilities, a logical outcome of the randomness of quantum measurement. To compute 2-bit OR, four rounds of computation are carried out, and within each round 1000 shots are executed. As shown in Fig. 5.2, the measurement outcomes with the highest probabilities for rounds 1 to 4 are 001, 011, 110, and 000, respectively, leading to

$$\text{Outcomes} \begin{cases} M_1 = \neg 0 \oplus \neg 0 \oplus \neg 1 \oplus \neg 0 = 1 & Alice \\ M_2 = \neg 0 \oplus \neg 1 \oplus \neg 1 \oplus \neg 0 = 0 & Bob \\ M_3 = \neg 1 \oplus \neg 1 \oplus \neg 0 \oplus \neg 0 = 0 & Charlie \end{cases} . \tag{5.14}$$

Alice and Bob send their summation bits to Charlie, who then performs an XOR operation on the obtained bits, resulting in $f'(\vec{a},\vec{b}) = 1 \oplus 0 \oplus 0 = 1$. This outcome is then transmitted to Alice and Bob, who calculate the final result by performing an XOR operation between the

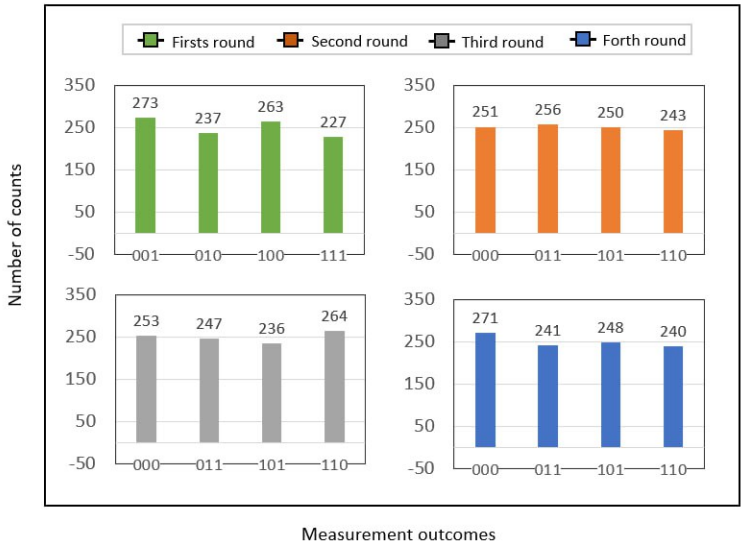


Figure 5.2: Measurement results of the quantum circuit demonstrated in Fig. 5.1, considering a particular scenario involving a 2-bit OR$(\vec{a}, \vec{b})$ function. The simulation is performed on *'qasm_simulator'* simulator. The circuit is run over 4 rounds with 1000 shots.

---

received classical bit and the shared random bit, yielding $f(\vec{a}, \vec{b}) = 1 \oplus 0 = 1$. The obtained result confirms the correctness of the protocol. The simulation results were performed on the *'qasm_simulator'* within Google Colab, Ubuntu 20.04.6 LTS, Python 3.10.12, and QisKit-0.43.2. Implementations are carried out on the ASUS Zenbook 14 UX425E laptop with 4 cores and an 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80 GHz processor, and 16 GB of RAM.

5.2.5 Result and Discussion

In this section, we provide security, privacy, and efficiency analyses of the proposed protocol. Additionally, we provide a comparative analysis between quantum SMC protocols and the protocol proposed in this work.

Privacy Analysis

To evaluate the privacy of the proposed protocol, we examine the data leakage in each step as follows:

1. Computation of $P(\vec{a})$ and $K(\vec{b})$: as the computation is conducted locally, there is no disclosure of any information regarding the inputs.

2. Qubit measurement by parties: no information is leaked.

3. Transmission of $(P_i \oplus r_i)$ and $(K_i \oplus r_i)$ to Charlie: resorting to the random bit r ensures that Charlie remains unaware of any details regarding the inputs. Despite using a random bit, Charlie gains knowledge about the parity of the inputs, at this stage.

4. Qubit rotation by parties: no information is leaked.

Once the value of the function is announced, participating parties remain uninformed regarding each other's inputs.

Security Analysis

The security of our protocol is derived directly from the principles of secure NAND computation outlined in [161] which rely on fundamental principles of quantum mechanics, such as the no-cloning theorem and the inability to measure certain quantum properties without disturbing the system. These features make it extremely difficult for an adversary to extract information from quantum systems without leaving detectable traces. As a result, entangled states and their quantum correlations offer unique opportunities for achieving secure communication. Consider the security against the attack from a party (for example Alice). If Alice wants to learn about Bob's input, she needs to intercept the bit value $K_i(\vec{b}) \oplus r_i$ that is transmitted between Bob and Charlie at step 4 of Protocol 9. However, since the transmission occurs over a secure classical channel, Alice fails to acquire any information about the bit value. Furthermore, if Charlie aims to retrieve the function output, his attempt will be unsuccessful due to his lack of knowledge concerning the random classical bits r_i.

The protocol lacks security against a coalition attack because Charlie possesses knowledge of the parity of input bits at each stage. This implies that if Charlie forms a coalition with either Alice or Bob, they can acquire information about the input of the other party. Consequently, the protocol can only be considered secure with a threshold of $th = 1$. The protocol is passively secure, which means that while the adversary can attempt to gather information from others, they are not permitted to deviate from the specified protocol execution.

Efficiency Analysis

Efficiency analysis involves three factors: quantum resources, communication complexity, and round complexity of the protocol. To compute Boolean functions as described in Eq. (5.6), two types of quantum resources are required: Bell and GHZ states. The application of Bell states can be substituted with a standard Quantum Key Distribution protocol to enable the sharing of random bits between Alice and Bob. The necessity for these quantum resources aligns with the requirements of the SMC protocol introduced in [30], which similarly emphasizes Boolean function evaluation. However, our scheme surpasses security level compared to the protocol outlined in [30] due to the use of an additional Z-phase rotation technique, concealing the output from Charlie. The communication cost, that is the number of bits transmitted among parties, is $2n$ bits, for each round of protocol execution. The round complexity of our protocol which refers to the number of rounds required for the execution of the protocol is 4.

Table 5.1 shows the functions to be computed, the required quantum resources, and the communication and the round complexity for various SMC protocols. While some SMC protocols listed in Table 5.1 utilize fewer quantum resources (single qubit), this research focuses on a different and more general type of functions. Furthermore, the use of a single qubit resulted in increased communication costs when compared to the GHZ state. Although the use of quantum resources, such as GHZ state, can increase with the input size n, the communication cost remains minimal.

Table 5.1: Comparison of different quantum SMC protocols. ℓ indicates the number of monomials. RoundCx and CommCx denote round complexity and communication complexity, respectively.

Quantum SMC protocols	Computed function	RoundCx	CommCx	Quantum resources
Ref. [30]	Boolean functions	4	$\mathcal{O}(2n)$	GHZ state + Bell state (Prot. A)
Ref. [31]	Pairwise AND	2	$\mathcal{O}(2n^2)$	Single qubit
Ref. [68]	N-tuple pairwise AND	2	$\mathcal{O}(2n^2)$	Single qubit
Ref. [32]	N-variable polynomials	3	$\mathcal{O}(\ell n^2)$	Single qudit (Prot. Γ_1) Entangled state (Prot. Γ_2)
Ref. [69]	Summation function	1	$\mathcal{O}(1)$	Entangled state
Our protocol	Boolean functions	4	$\mathcal{O}(2n)$	GHZ state + Bell state

5.3 SMC Using Single Qubits

In this section, we propose a quantum SMC protocol designed for evaluating binary Boolean functions using single qubits. We design the corresponding quantum circuit and implement the proposed protocol on the IBM QisKit platform, yielding reliable outcomes.

5.3.1 Pairwise AND Computation

This section reviews the private computation of pairwise AND initially proposed by [31] which serves as the basis for our protocol. This approach allows multiple participants to collectively compute the pairwise AND of their inputs without exposing any information about their inputs. Consider the pairwise AND function

$$f(x_1, ..., x_n) = \bigoplus_{j=1}^{n-1} \Big(x_{j+1} \cdot (\bigoplus_{i=1}^{j} x_i) \Big), \tag{5.15}$$

where $\bigoplus$ is addition modulo 2 (XOR) and "." denotes the AND operation. This function computes the pairwise AND operation for each pair of values in the sequence and then performs a bitwise XOR on the results.

Suppose that n parties with input bits $x_1, x_2, ..., x_n$ want to compute pairwise AND of their private inputs with the assistance of a server. In the initial step, a secret shared random bit $r = \overset{n}{\underset{i=1}{\oplus}} r_i$ is distributed among parties such that each party i holds r_i. Let us specify the $-\pi/2$ rotation and the π rotation around the y axis of the Bloch sphere as

$$U = R_y(\pi/2) = e^{-i\pi\sigma_y/4}, \tag{5.16}$$

and

$$V = R_y(\pi) = e^{-i\pi\sigma_y/2}. \tag{5.17}$$

Initially, the server prepares a qubit $|0\rangle$ and sends it to the first party. Party P_1 performs the operations $V^{r_1}U^{x_1}$ on the qubit based on the input x_1 and random bit r_1. The rotations are carried out in such a way that if the bit value is 0, the operation U (V) is applied to the qubit, and if the bit value is 1, the operation U (V) is omitted and the qubit remains unchanged. The modified qubit is then forwarded to the subsequent party P_2, where the

rotations $V^{r_2}U^{x_2}$ are applied. This sequence of actions iteratively repeats until all the parties have applied their rotations to the qubit. Employing an XOR routine detailed at [31], the parties compute the XOR of their peers' private inputs $\oplus_i x_i$. Subsequently, one of the parties performs the $(U^\dagger)^{\oplus_i x_i}$ operation on the qubit resulting in

$$|r \oplus f\rangle = (U^\dagger)^{\oplus_i x_i} \underbrace{V^{r_n}U^{x_n}}_{\mathcal{P}_n} ... \underbrace{V^{r_2}U^{x_2}}_{\mathcal{P}_2} \underbrace{V^{r_1}U^{x_1}}_{\mathcal{P}_1} |0\rangle . \tag{5.18}$$

Afterward, the qubit is returned to the server that measures it in the computational basis, revealing the classical outcome $(r \oplus f)$. Exploiting the XOR routine, parties locally compute r by XORing all the random bits r_i of their peers and retrieve the final output as

$$f(x_1, ..., x_n) = r \oplus (r \oplus f). \tag{5.19}$$

In the next section, we use the result of Eq. (5.19) to compute binary Boolean functions in a secure multiparty manner.

5.3.2 Boolean Function Computation using Single Qubits

In this section, we propose a quantum-based SMC protocol to compute binary Boolean functions using single qubits. As outlined in Eq. (5.6), a Boolean function can be computed as

$$f(\vec{a}, \vec{b}) = \bigoplus_{i=1}^{m} P_i(\vec{a}) \,.\, K_i(\vec{b}). \tag{5.20}$$

The right-hand side of Eq. (5.6) indicates that $f(\vec{a}, \vec{b})$ can be computed using the secure AND computation method explained in Section 5.3.1. Depending on the particular Boolean function under examination, the polynomials described by Eq. (5.6) are computed as follows: consider the Equivalence function $EQ(\vec{a}, \vec{b})$ in which the output of the computation is true if the two statements or conditions are equivalent. The polynomials needed for a 2-bit Equivalence function $EQ(\vec{a}, \vec{b})$ can be computed as

$$EQ(\vec{a}, \vec{b}) = 1 + \vec{a} + \vec{b}\,, \tag{5.21}$$

therefore,

$$\begin{aligned} EQ(\vec{a}, \vec{b}) =& EQ(a_1, b_1) \,.\, EQ(a_2, b_2) \\ =& (1 + a_1 + b_1) \,.\, (1 + a_2 + b_2) \\ =& 1 + 1.a_2 + 1.b_2 + a_1.1 + a_1a_2 + a_1b_2 + b_1.1 + b_1a_2 + b_1b_2 \\ =& \underbrace{(1 + a_1 + a_2 + a_1a_2)}_{P_1} . \underbrace{1}_{K_1} + \underbrace{1}_{P_2} . \underbrace{b_1b_2}_{K_2} + \underbrace{(1 + a_2)}_{P_3} . \underbrace{b_1}_{K_3} + \underbrace{(1 + a_1)}_{P_4} . \underbrace{b_2}_{K_4} \\ =& \sum_{i=1}^{4} P_i(a_1, a_2).K_i(b_1, b_2). \end{aligned} \tag{5.22}$$

In Eqs. (5.21) and (5.22), addition and multiplication are the XOR, and the logical AND, respectively [164]. Equation (5.22) indicates that a 2-bit Equivalence function $EQ(\vec{a}, \vec{b})$ can

be computed using vectors of polynomials

$$P(\vec{a}) = \begin{bmatrix} 1 + a_1 + a_2 + a_1 a_2 \\ 1 \\ 1 + a_2 \\ 1 + a_1 \end{bmatrix}, \quad K(\vec{b}) = \begin{bmatrix} 1 \\ b_1 b_2 \\ b_1 \\ b_2 \end{bmatrix}. \tag{5.23}$$

Note that the size of P and K can grow with n implying a greater demand for quantum resources to compute the desired function.

The proposed SMC protocol progresses through the following steps. Initially, Alice and Bob decide on a Boolean function and independently calculate the necessary polynomial vectors P and K based on their respective inputs. The protocol is executed over m rounds, corresponding to the number of elements in the polynomial vectors P and K. Each round i $(1 \leq i \leq m)$ proceeds as follows. Initially, a Bell state $|\varphi_i\rangle = (|00\rangle + |11\rangle)/\sqrt{2}$ is distributed between Alice and Bob, with each party possessing a qubit. Utilizing this Bell state, the two parties share a secret random bit r_i known only to them. Afterward, Charlie prepares a qubit at state $|0\rangle$ and sends it to Alice. Note that although the proposed protocol is tailored for two-party computation with independent inputs, the involvement of a third party, referred to as Charlie, is essential. This is because unconditionally secure two-party computation is not achievable, as outlined in [165].

However, Charlie's input is not independent of the inputs from the other parties. It is determined by the parity of the two other inputs $(P \oplus K)$. Next, Alice receives the qubit and performs the operation $V^{r_i} U^{P_i(\vec{a})}$ on the qubit, considering the input $P_i(\vec{a})$ and the random bit r_i. Note that, the objective of the U rotation is to encrypt Alice's input, whereas the V rotation is employed to obscure the function's output from untrusted parties. Afterward, Alice sends the altered qubit to Bob who performs $V^{r_i} U^{K_i(\vec{b})}$ on the received qubit. Bob then sends the qubit to Charlie who performs $U^{\dagger(P_i(\vec{a}) \oplus K_i(\vec{b}))}$ on the qubit leading to

$$|f_i'\rangle = \overbrace{U^{\dagger(P_i(\vec{a}) \oplus K_i(\vec{b}))}}^{Charlie} \overbrace{V^{r_i} U^{K_i(\vec{b})}}^{Bob} \overbrace{V^{r_i} U^{P_i(\vec{a})}}^{Alice} |0\rangle \,. \tag{5.24}$$

The state $|f_i'\rangle$ contains the value of $\mathrm{AND}(P_i, K_i)$ up to a bit flip r_i. Charlie then measures the qubit on a computation basis and stores the classical result. Once the protocol is executed over m rounds, Charlie performs an XOR among all the measurement outcomes to obtain $f' = \bigoplus_{i=1}^{m} f_i'$. Charlie then sends the result to Alice and Bob, who retrieve the final output by XORing the received classical bit f' and the random bit $r = \bigoplus_{i=1}^{m} r_i$ as

$$f(\vec{a}, \vec{b}) = r \oplus f'. \tag{5.25}$$

Protocol 10 provides an overview of the procedural steps. Utilizing the proposed protocol, in the next section, we compute the 2-bit Equivalence function using the IBM QisKit platform.

5.3.3 QisKit Implementation

In this section, we design a quantum circuit for the proposed protocol and explain its implementation in IBM QisKit under both ideal and noisy conditions. We compute a special

Protocol 10 Quantum SMC Protocol using Single Qubits

Inputs: Inputs $\vec{a} = (a_1, a_2, ..., a_n)$ for Alice, and $\vec{b} = (b_1, b_2, ..., b_n)$ for Bob.
Outputs: $f(\vec{a}, \vec{b})$ for Alice and Bob.

1. For $1 \leq i \leq m$, repeat steps 2-9 for each term.
2. Alice and Bob compute the associated polynomials $P_i(\vec{a})$ and $K_i(\vec{b})$ based on the specific function being calculated.
3. Alice and Bob are provided with two qubits, constituting a Bell state $|\varphi_i\rangle = (|00\rangle + |11\rangle)/\sqrt{2}$. Subsequently, each of them measures a qubit of the Bell state and records the outcome as r_i.
4. Using a secure classical channel, Alice and Bob send to Charlie the bits $P_i(\vec{a}) \oplus r_i$ and $K_i(\vec{b}) \oplus r_i$, respectively.
5. Charlie obtains the parity of parties' inputs by computing $(P_i(\vec{a}) \oplus r_i) \oplus (K_i(\vec{b}) \oplus r_i) = P_i(\vec{a}) \oplus K_i(\vec{b})$. The resulting value corresponds to Charlie's input.
6. Charlie provides a qubit in state $|0\rangle$ and sends it to Alice.
7. Alice performs $V^{r_i}U^{P_i(\vec{a})}$ on the qubit, considering the values of P_i and r_i, and sends the qubit to Bob.
8. Bob performs $V^{r_i}U^{K_i(\vec{b})}$ on the qubit and then sends it to Charlie.
9. Charlie performs $U^{\dagger(P_i(\vec{a})\oplus K_i(\vec{b}))}$ on the qubit and measure it in computational basis.
10. After these steps are repeated over m rounds (where $i = m$), Charlie performs an XOR among all the measurement outcomes and sends the result to Alice and Bob.
11. Alice and Bob retrieve the final output as $f(\vec{a}, \vec{b}) = r \oplus f'(\vec{a}, \vec{b})$, with $r = \bigoplus_{i=1}^{i} r_i$.

case of the 2-bit EQ$(\vec{a}, \vec{b})$ function. Our code is accessible in the GitHub repository `https://github.com/Quantum-SMC`.

Figure 5.3 illustrates the corresponding quantum circuit where a qubit in the initial state $|0\rangle$ is prepared. To encrypt the inputs and output, we apply $R_y(\pi)$, $R_y(\pi/2)$, and $R_y(-\pi/2)$ operations to the qubit, corresponding to V, U, and $U^\dagger$, respectively. Afterward, the qubit is measured in the computational basis, and the classical outcome is stored in the classical register of the QisKit environment. In this circuit, the classical register C_0 is used to store the measurement result of q_0. Let us consider an example where the three parties, Alice, Bob, and Charlie, with input bits $\vec{a} = \{1, 0\}$, $\vec{b} = \{1, 0\}$, and $\vec{a} \oplus \vec{b} = \{0, 0\}$ aim to compute function $EQ(\vec{a}, \vec{b}) = (1\, EQ\, 1)\, EQ\, (0\, EQ\, 0))$, which should yield the output 1. Alice and Bob share random bits $\mathbf{\vec{r}} = (0, 1, 1, 0)$ leading to $r = \bigoplus_{i=1}^{4} r_i = 0$. Considering Eq. (5.22), Alice and Bob compute $P = (1, 0, 1, 0)$ and $K = (0, 1, 1, 0)$, which correspond to the EQ function. Afterward, parties execute the circuit for four rounds. In each round, 100 shots were applied. Figure 5.4 illustrates the measurement outcomes for four rounds of circuit execution under (a)

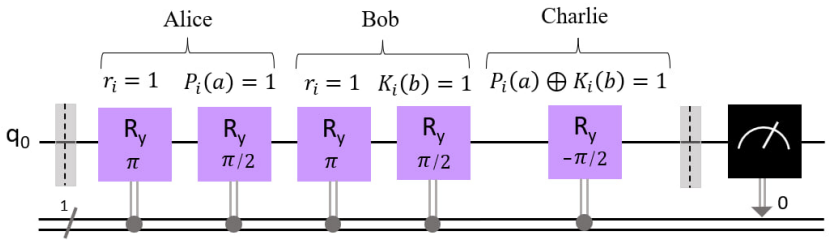


Figure 5.3: Quantum circuit for the proposed protocol for round i. The qubit labeled as q_0 is the initial qubit with state $|0\rangle$. R_y indicates the rotation operation of the qubit along the y-axis of the Bloch sphere, with respect to the classical bits r, P_i, K_i, and $P_i \oplus K_i$. Note that the rotation gates are exclusively applied when the bit values are equal to 1; otherwise, for bit values equal to 0, they are omitted from the circuit.

---

an ideal noiseless setting and (b) a noisy setting. The noise model includes bit-flip, phase-flip, amplitude damping, phase damping, and depolarizing errors, each with a probability of 0.1. This model was applied to both quantum gates and measurement operations. Our results indicate that the probability of obtaining the correct answer is, on average, 80.25%. This reduction in accuracy is due to quantum errors, which can be mitigated with error correction techniques. The most frequent measurement outcomes for rounds 1 to 4 are 0, 1, 0, and 0, respectively. Subsequently, Charlie's outcome is derived by XORing the measurement results from each round, yielding $0 \oplus 1 \oplus 0 \oplus 0 = 1$. This outcome is then transmitted to Alice and Bob, who retrieve the actual output of the EQ function by XORing the received classical bit and the private random bit (i.e., $1 \oplus 0 = 1$). The simulation results were obtained using the *'AerSimulator'* with 100 shots per round, conducted in Google Colab on Ubuntu 20.04.6 LTS, with Python 3.10.12 and QisKit 0.43.2. We carried out the implementations on an ASUS Zenbook 14 UX425E laptop with 4 cores, an 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80 GHz processor, and 16 GB of RAM.

5.3.4 Result and Discussion

This section provides security, privacy, and complexity analyses of the proposed protocol. Additionally, we compare our work with another SMC protocol specifically designed for Boolean function computation.

Privacy Analysis

To validate the privacy of the proposed scheme, we assess the data leakage in each step as follows. In step 2 of the protocol, where the computation of $P(\vec{a})$ and $K(\vec{b})$ occurs locally, no information is disclosed regarding the inputs. During the transmission of $(P_i \oplus r_i)$ and $(K_i \oplus r_i)$ Charlie remains uninformed about parties' private inputs due to the use of a random bit. However, Charlie gains knowledge about the parity of the inputs at this stage. In the qubit transmission among parties, no information is revealed. Even if an eavesdropper successfully intercepts the particle transferred from one party to another, they are unable to measure it on the appropriate measurement basis. Qubit measurement and qubit rotation are done without the leakage of information.

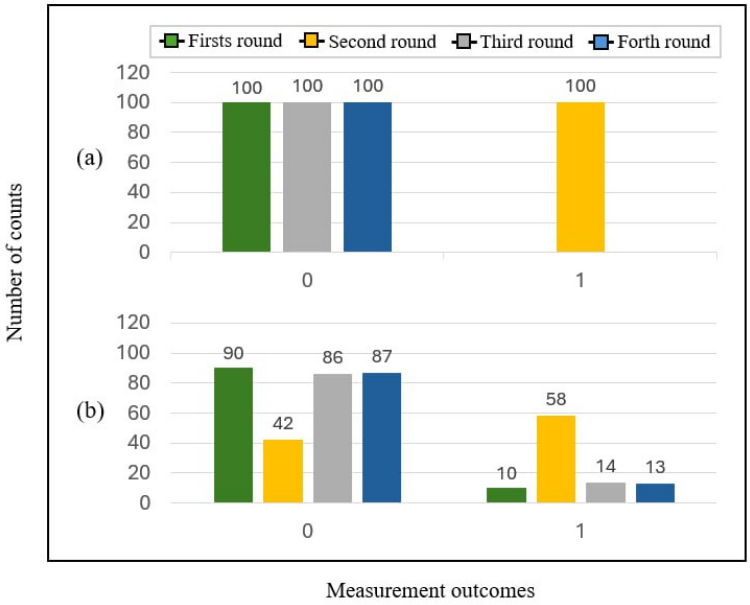


Figure 5.4: The measurement outcomes of the corresponding quantum circuit with panel (a) illustrating the ideal noiseless results and panel (b) showing the results affected by quantum noise. Both simulations utilized the *'AerSimulator'* backend and were conducted over four rounds, each with 100 shots.

Security Analysis

The security of this scheme is derived from the fundamental principles of quantum mechanics, which makes it difficult for an adversary to extract information from quantum systems without leaving detectable traces. Consider security against potential attacks from Charlie. If Charlie aims to obtain any information about a party's private input, for instance, Alice, he needs to intercept the particle transmitted from Alice to Bob, and measure it in the right measurement basis ($|0\rangle$, $|1\rangle$). Nevertheless, Charlie cannot determine the correct measurement basis because he lacks information about the unitary operation $V^{r_i}U^{P(\vec{a})}$ and Alice's input bit. Secondly, if Charlie wants to extract the output of the function, he fails because he knows nothing about random bit r_i.

The protocol's vulnerability to a coalition attack arises from Charlie's awareness of the parity of input bits at every stage. This signifies that if Charlie collaborates with either Alice or Bob, they can gain insights into the input of the other party. As a result, the protocol's security can only be assured with a threshold of $th = 1$. The protocol maintains passive security, indicating that although the adversary can try to extract information from others, any deviation from protocol execution is prohibited.

Efficiency Analysis

The efficiency of SMC protocols is crucial for practical applications. Various factors influence the efficiency of these protocols, such as computation and communication overheads, as well as the amount of required quantum resources. To obtain the computation complexity, we consider the required operations at each step: polynomials (10 XOR), parity of inputs

Table 5.2: Comparison of different quantum SMC protocols for Boolean function computation. m indicates the number of rounds. CompCx and CommCx denote computation and communication complexity, respectively. U and V specify the $-\pi/2$ rotation and the π rotation around the y axis of the Bloch sphere as defined in the paper. M represents the qubit measurement.

Quantum SMC Prot.	CompCx	CommCx	Quantum resources	Quantum operations
Ref. [30]	$(6m+12)$ XOR	$\mathcal{O}(2m)$	GHZ + Bell	$(5m)M$
Ref. [47]	$(6m+14)$XOR+$(3m)$NOT	$\mathcal{O}(2m)$	GHZ + Bell	$(m)\sigma_z$+ $(3m)U$+$(3m)M$
This work	$(6m+12)$ XOR	$\mathcal{O}(5m)$	Single qubit+Bell	$(2m)V + (3m)U$ + $(3m)M$

($3m$ XOR), quantum operations ($(2m)VU$, mU, and $(3m)M$, with M representing qubit measurement), Charlie's outcome (m XOR), and the final output by parties ($2(m+1)$ XOR). Overall, the computational cost of our protocol is $(12 + 6m)$ XOR, $(2m)$ instances of VU operation, m instances of U operation, and $(3m)$ qubit measurement M. The communication complexity of our protocol is $\mathcal{O}(5m)$, reflecting the number of bits exchanged during each round m.

To evaluate our scheme, in Table 5.2, we compare the complexity of the proposed protocol with other SMC protocols that emphasize on secure Boolean function computation. As shown in Table 5.2, two types of quantum resources are required to compute Boolean functions within our approach: Bell state and single qubit. The necessity for these quantum resources is reduced to one-third (66.7%) compared to [30], in which three-qubit GHZ states are employed via MBQC approach. Although the computation complexity remains consistent compared to [30], the communication overhead in our protocol scales as $\mathcal{O}(5m)$ which is 40% higher than that of other approaches. This outcome is anticipated since the exchange of single qubits among parties inherently elevates communication requirements. In contrast, the MBQC approach avoids qubit exchanges by using distributed entangled particles, though this method results in a higher demand for quantum resources and a costly process of entangling particles. Furthermore, while our approach involves more quantum operations, this is justified by the enhanced security it provides.

5.4 Final Remark

We proposed two quantum-based SMC protocols for computing binary Boolean functions. The first protocol leverages entanglement in the GHZ state for computation, while the second utilizes single qubits. Although these protocols primarily focus on a limited number of parties, extending them to support an arbitrary number of n participants presents a promising avenue for advancing SMC protocols in quantum computing. We implemented our protocols on the IBM QisKit platform and obtained experimental results confirming the feasibility and practicality of our approach.

Chapter 6

Conclusion and Future Work

This chapter outlines our key findings derived from the research conducted within the framework of this thesis. The study primarily explored the implementation of quantum-based SMC through two main avenues: quantum communication and quantum computing.

Section 6.1 provides an overview of the principal conclusions drawn from the research. Section 6.2 addresses the existing challenges and constraints encountered during the study. Finally, Section 6.3 explores potential directions for future research that could build upon the findings presented in this work.

6.1 Conclusion

Classical SMC protocols rely on PKC, which is vulnerable to attacks from powerful quantum computers capable of efficiently breaking traditional cryptographic schemes. In this thesis, we have addressed the challenges faced by classical SMC in terms of security and efficiency. We investigated two primary approaches for achieving quantum-based SMC: quantum communication and quantum computing. Using the first approach, we proposed a QSMC framework which leverages quantum communication technologies including QRNG, QKD, and QOKD along with a robust KMS and the MASCOT SMC protocol. The key advantage of the proposed framework over classical methods is its independence from PKC and its resilience against quantum computer attacks. Using the proposed QSMC framework, we developed two real-life applications of SMC: SRD and DSP. The SRD service assists vehicles in safely exiting a highway by determining the optimal timing for lane changes, ensuring efficient traffic flow and enhanced safety, all while maintaining the privacy of each vehicle's data. We evaluated the complexity of this service and the result shows that the communication cost of the quantum method is reduced by 97% while the execution time increased by 42% compared to the classical method. The substantial reduction in communication cost is particularly valuable for vehicular network applications, where the available spectrum for radio channels is a constrained resource. In DSP, the QSMC framework, combined with FL, enabled multiple pharmaceutical companies to collaboratively train a GCN model for accurately predicting drug molecule solubility. This approach achieved an average loss value of 0.0046 and an MSE of 1.2 on the test set.

The quantum communication approach has achieved a higher TRL due to its successful deployment in various real-world contexts, particularly for secure key exchange over fiber and satellite networks. The QKD protocol, for example, is robust against eavesdropping, and its

practical implementations demonstrate strong security for long-distance data transmission in networked environments. The largest practical QKD system to date has been achieved by a team from China, reaching over 4,600 kilometers through a hybrid system of fiber-optic links and satellites. The backbone of this impressive network is the Micius satellite, launched by China in 2016. In addition to these achievements, fiber-based QKD networks have achieved distances up to around 600 kilometers by leveraging trusted nodes, where each intermediate node must securely manage and relay the key between endpoints, as true end-to-end QKD is limited by signal loss in optical fibers. This makes it suitable for secure multiparty applications like vehicular networks or medical data sharing where real-time data privacy is crucial.

In the subsequent chapter, we achieved quantum SMC using the quantum computing approach that deploys quantum computers and quantum processors for algorithm execution. This method primarily harnesses key quantum phenomena, including entanglement and random measurements. Using this approach, we propose a two-party protocol for the secure computation of Boolean functions exploiting MBQC. Within this method, a highly entangled GHZ state is generated and distributed among participants to compute the desired Boolean function while encrypting their inputs. We implemented our protocol in an IBM QisKit simulator and validated its correctness. The complexity of the protocol is $2m$ bits for each round of protocol execution which is the same as other approaches. However, we achieved higher security by performing a rotation gate that hides the output of computation from untrusted parties. We extended this protocol to use single qubits instead of highly entangled GHZ state. The new protocol requires less quantum resources by 66%. However, the communication cost was increased to $5m$ which is 40% higher. This outcome is anticipated since the exchange of single qubits among parties inherently elevates communication requirements. We tested this protocol on both ideal and noisy settings. The noise model includes bit-flip, phase-flip, amplitude damping, phase damping, and depolarizing errors, each with a probability of 0.1. This model was applied to both quantum gates and measurement operations. Our results indicated that the probability of obtaining the correct answer is, on average, 80.25%. This reduction in accuracy is due to quantum errors, which can be mitigated with error correction techniques.

The quantum computing-based approach utilizes methods such as trapped ions, superconducting qubits, and quantum dots to build quantum computers. Currently, the largest quantum computer based on qubit count is IBM's $1,000+$ qubit superconducting chip, marking a significant milestone in scaling quantum processors. Meanwhile, trapped-ion systems, such as Quantinuum's 56-qubit model, are also progressing, with a strong performance in coherence and error rates, aiming toward scalable, fault-tolerant systems by 2030. Quantum computing systems increasingly integrate Quantum Processing Unit (QPU) as a specialized hardware component designed to perform quantum computations by manipulating qubits, for parallelized tasks in quantum simulation and machine learning. SMC within this framework can potentially provide even higher levels of efficiency and computational power. This approach is promising for applications requiring significant computational resources, such as large-scale cryptographic protocols. However, quantum computing for SMC is less ready for immediate deployment.

6.2 Limitation and Challenges

Quantum communication systems face significant limitations related to distance constraints. The effectiveness of QKD and QOKD diminishes over long-range distances because of signal loss in optical fibers, necessitating the use of trusted nodes to relay keys securely. This reliance on intermediary nodes introduces potential vulnerabilities, as each node must be trusted to ensure the confidentiality and integrity of the key exchange. Furthermore, the implementation of fiber-based QKD and QOKD systems often requires substantial infrastructure investments and can be hampered by environmental elements, such as temperature changes and physical obstructions, which may impact the quality of the quantum signals. As a result, while quantum communication offers robust security features, its practical deployment is limited by these logistical and technical challenges.

Quantum computing, while promising for SMC, is currently hindered by several technical restrictions. A major obstacle to overcome is qubit coherence, as qubits are highly sensitive to decoherence and noise from the environment, resulting in computational errors. These error rates can greatly affect the reliability of quantum algorithms, necessitating the development of complex error correction techniques that further complicate the execution of quantum protocols. Moreover, the present state of quantum hardware remains in its early stages, with issues such as limited qubit connectivity and scalability posing obstacles to the realization of large-scale quantum systems. As researchers work to enhance qubit performance and design better quantum architectures, the immediate deployment of quantum computing for practical applications remains a challenging endeavor.

6.3 Future Directions:

As the domains of quantum communication and quantum computing continue to evolve, several promising future directions emerge that could enhance their applicability and effectiveness in SMC. In this section, we cover some of the future work that may be interesting to be explored.

- The QSMC framework proposed in Chapter 3 could be expanded to include other SMC protocols such as Yao. This extension will allow users to exploit both Boolean and arithmetic circuits. To this end, other SMC protocols need to be studied and integrated into the framework. Additionally, while the QSMC framework ensures active security by preventing any malicious party from learning about the output of others, a more comprehensive evaluation can be obtained by implementing various types of attacks within the framework.

- To enhance the sophistication of the SRD service implemented in Chapter 4, a two-dimensional representation of locations and the road architecture derived from actual maps can be considered. Additionally, in the DSP use case, instead of utilizing a classical GCN, a hybrid classical-quantum neural network could be deployed to improve efficiency and security during the local training process. To this end, parameterized quantum circuits can be integrated as a hidden layer within a neural network.

- The two-party Boolean function computation protocols proposed in Chapter 5 can be extended to accommodate more than two parties. Additionally, the implementation of more sophisticated functions such as addition or multiplication can be investigated.

While the proposed protocols have been implemented and tested in both ideal and noisy simulators, their evaluation could be significantly enhanced through the use of real quantum devices equipped with QPUs.

References

[1] Maurício J. Ferreira, Nuno A. Silva, Armando N. Pinto, and Nelson J. Muga. Characterization of a quantum random number generator based on vacuum fluctuations. *Applied Sciences*, 11(16), 2021.

[2] Sara Mantey, Nuno Silva, Armando Pinto, and Nelson Muga. Design and implementation of a polarization-encoding system for quantum key distribution. *Journal of Optics*, 26(7):075704, 2024.

[3] Pam Carter, Graeme T Laurie, and Mary Dixon-Woods. The social licence for research: why care.data ran into trouble. *Journal of Medical Ethics*, 41(5):404–409, 2015.

[4] Margaret McCartney. Care.data doesn't care enough about consent. *British Medical Journal*, 348:g2831, 2014.

[5] Paraskevas Vezyridis and Stephen Timmons. Understanding the care.data conundrum: New information flows for economic growth. *Big Data and Society*, 4(1):1–12, 2017.

[6] Andrew C. Yao. Protocols for secure computations. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, pages 160–164, 1982.

[7] Wenliang Du and Mikhail J. Atallah. Secure multi-party computation problems and their applications: A review and open problems. In *Proceedings of the 2001 Workshop on New Security Paradigms*, pages 13–22, 2001.

[8] Marcel Keller, Emmanuela Orsini, and Peter Scholl. Mascot: Faster malicious arithmetic secure computation with oblivious transfer. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 830–842, 2016.

[9] Raluca Ada Popa, Hari Balakrishnan, and Andrew J. Blumberg. Vpriv: Protecting privacy in location-based vehicular services. In *Proceedings of the 18th Conference on USENIX Security Symposium*, pages 335–350, 2009.

[10] Zeinab Rahmani, Luis Barbosa, and Armando N. Pinto. Collision warning in vehicular networks based on quantum secure multiparty computation. In *II Workshop de Comunicação e Computação Quântica WQuantum*, pages 19–24, 2022.

[11] Manuel B. Santos, Ana C. Gomes, Armando N. Pinto, and Paulo Mateus. Quantum secure multiparty computation of phylogenetic trees of sars-cov-2 genome. In *Proceedings of the Telecoms Conference*, pages 1–5, 2021.

[12] Brian Knott, Shobha Venkataraman, Awni Hannun, Shubho Sengupta, Mark Ibrahim, and Laurens van der Maaten. Crypten: Secure multi-party computation meets machine learning. In *Advances in Neural Information Processing Systems*, volume 34, pages 4961–4973, 2021.

[13] Vaikkunth Mugunthan, Antigoni Polychroniadou, David Byrd, and Tucker Hybinette Balch. Smpai: Secure multi-party computation for federated learning. In *Proceedings of the NeurIPS 2019 Workshop on Robust AI in Financial Services*, volume 21, 2019.

[14] Rong Ma, Yi Li, Chenxing Li, Fangping Wan, Hailin Hu, Wei Xu, and Jianyang Zeng. Secure multiparty computation for privacy-preserving drug discovery. *Bioinformatics*, 36(9):2872–2880, 2020.

[15] Armando N. Pinto, Laura Ortiz, Manuel Santos, Ana C. Gomes, Juan P. Brito, Nelson J. Muga, Nuno A. Silva, Paulo Mateus, and Vicente Martin. Quantum enabled private recognition of composite signals in genome and proteins. In *Proceedings of the 22nd International Conference on Transparent Optical Networks*, pages 1–4, 2020.

[16] David Byrd and Antigoni Polychroniadou. Differentially private secure multi-party computation for federated learning in financial applications. In *Proceedings of the 1st ACM International Conference on AI in Finance*, pages 1–9, 2020.

[17] Dan Bogdanov, Marko Jõemets, Sander Siim, and Meril Vaht. How the estonian tax and customs board evaluated a tax fraud detection system based on secure multi-party computation. In *Financial Cryptography and Data Security*, pages 227–234, 2015.

[18] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *Society for Industrial and Applied Mathematics*, 41(2):303–332, 1999.

[19] Ronald L. Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the Association for Computing Machinery*, 21(2):120–126, 1978.

[20] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. In *Democratizing Cryptography: The Work of Whitfield Diffie and Martin Hellman*, pages 365–390, 2022.

[21] René Schoof. Elliptic curves over finite fields and the computation of square roots mod p. *Mathematics of computation*, 44(170):483–494, 1985.

[22] Mariano Lemus, Mariana F Ramos, Preeti Yadav, Nuno A Silva, Nelson J Muga, André Souto, Nikola Paunković, Paulo Mateus, and Armando N Pinto. Generation and distribution of quantum oblivious keys for secure multiparty computation. *Applied Sciences*, 10(12):4080, 2020.

[23] Daniel J. Bernstein and Tanja Lange. Post-quantum cryptography. *Nature*, 549(7671):188–194, 2017.

[24] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. Ntru: A ring-based public key cryptosystem. In *Algorithmic Number Theory*, pages 267–288, 1998.

[25] Robert J. McEliece. A public-key cryptosystem based on algebraic. *Coding Theory*, 4244:114–116, 1978.

[26] Leslie Lamport. Constructing digital signatures from a one way function. Technical Report CSL-98, 1979.

[27] Zeinab Rahmani, Luis S. Barbosa, and Armando N. Pinto. Quantum privacy-preserving service for secure lane change in vehicular networks. *IET Quantum Communication*, 4(3):103–111, 2023.

[28] Armando N. Pinto, Manuel B. Santos, Nuno A. Silva, Nelson J. Muga, and Paulo Mateus. Oblivious keys for secure multiparty computation obtained from a cv-qkd. In *Proceedings of the 23rd International Conference on Transparent Optical Networks*, pages 1–4, 2023.

[29] Manuel B. Santos, Ana C. Gomes, Armando N. Pinto, and Paulo Mateus. Private computation of phylogenetic trees based on quantum technologies. *IEEE Access*, 10:38065–38088, 2022.

[30] Klearchos Loukopoulos and Daniel E. Browne. Secure multiparty computation with a dishonest majority via quantum means. *Physical Review A*, 81:062336, 2010.

[31] Marco Clementi, Anna Pappa, Andreas Eckstein, Ian A. Walmsley, Elham Kashefi, and Stefanie Barz. Classical multiparty computation using quantum resources. *Physical Review A*, 96:062317, 2017.

[32] Changbin Lu, Fuyou Miao, Junpeng Hou, Zhaofeng Su, and Yan Xiong. Secure multiparty computation with a quantum manner. *Journal of Physics A: Mathematical and Theoretical*, 54(8):085301, 2021.

[33] Stefanie Barz, Vedran Dunjko, Florian Schlederer, Merritt Moore, Elham Kashefi, and Ian A. Walmsley. Enhanced delegated computing using coherence. *Physical Review A*, 93:032339, 2016.

[34] ZhaoXu Ji, HuanGuo Zhang, HouZhen Wang, FuSheng Wu, JianWei Jia, and WanQing Wu. Quantum protocols for secure multi-party summation. *Quantum Information Processing*, 18:1–19, 2019.

[35] Maurício J. Ferreira, André Carvalho, Nuno A. Silva, Armando N. Pinto, and Nelson J. Muga. Probable prime generation from a quantum randomness source. In *Proceedings of the 23rd International Conference on Transparent Optical Networks*, pages 1–4, 2023.

[36] Charles H. Bennett and Gilles Brassard. Quantum cryptography: Public key distribution and coin tossing. *Theoretical Computer Science*, 560:7–11, 2014.

[37] Hans J. Briegel, David E. Browne, Wolfgang Dür, Robert Raussendorf, and Maarten Van den Nest. Measurement-based quantum computation. *Nature Physics*, 5(1):19–26, 2009.

[38] Michael A. Nielsen. Cluster-state quantum computation. *Reports on Mathematical Physics*, 57(1):147–161, 2006.

[39] Heng-Yue Jia, Qiao-Yan Wen, Ting-Ting Song, and Fei Gao. Quantum protocol for millionaire problem. *Optics Communications*, 284(1):545–549, 2011.

[40] Wen Liu and Wei Zhang. A quantum protocol for secure manhattan distance computation. *IEEE Access*, 8:16456–16461, 2020.

[41] Run-Hua Shi. Quantum multiparty privacy set intersection cardinality. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 68(4):1203–1207, 2021.

[42] Robert Raussendorf, Daniel E. Browne, and Hans J. Briegel. Measurement-based quantum computation on cluster states. *Physical Review A*, 68:022312, 2003.

[43] Yiqing Zhou, E. Miles Stoudenmire, and Xavier Waintal. What limits the simulation of quantum computers? *Physical Review X*, 10:041038, 2020.

[44] Zhenyu Cai, Ryan Babbush, Simon C. Benjamin, Suguru Endo, William J. Huggins, Ying Li, Jarrod R. McClean, and Thomas E. O'Brien. Quantum error mitigation. *Reviews of Modern Physics*, 95:045005, 2023.

[45] Francesco Battistel, Christopher Chamberland, Kauser Johar, Ramon WJ Overwater, Fabio Sebastiano, Luka Skoric, Yosuke Ueno, and Muhammad Usman. Real-time decoding for fault-tolerant quantum computing: progress, challenges and outlook. *Nano Futures*, 7(3):032003, 2023.

[46] Zeinab Rahmani, Armando N. Pinto, and Luis S. Barbosa. Quantum-secured federated learning for solubility prediction in drug molecules. pages 1–27, Under review.

[47] Zeinab Rahmani, Armando N. Pinto, and Luis Barbosa. Secure two-party computation via measurement-based quantum computing. *Quantum Information Processing*, 23(6):221, 2024.

[48] Zeinab Rahmani, Armando N. Pinto, and Luis S. Barbosa. Private computation of boolean functions using single qubits. In *Proceedings of the 15th International Conference on Parallel Processing and Applied Mathematics*, pages 1–12, 2024.

[49] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game, or a completeness theorem for protocols with honest majority. In *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, pages 307—328. 2019.

[50] Adi Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, 1979.

[51] Shafi Goldwasser, Silvio Micali, and Chales Rackoff. The knowledge complexity of interactive proof-systems. In *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, pages 203–225, 2019.

[52] Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols. In *Proceedings of the 22nd annual ACM symposium on Theory of computing*, pages 503–513, 1990.

[53] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, pages 351–371. 2019.

[54] Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. Semi-homomorphic encryption and multiparty computation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 169–188, 2011.

[55] Ivan Damgård, Valerio Pastro, Nigel Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In *Advances in Cryptology – CRYPTO 2012*, pages 643–662, 2012.

[56] Ivan Damgård, Marcel Keller, Enrique Larraia, Valerio Pastro, Peter Scholl, and Nigel P. Smart. Practical covertly secure mpc for dishonest majority – or: Breaking the spdz limits. In *Computer Security – ESORICS 2013*, pages 1–18, 2013.

[57] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*, pages 169–178, 2009.

[58] Ran Canetti. Universally composable security: a new paradigm for cryptographic protocols. In *Proceedings of 42nd IEEE Symposium on Foundations of Computer Science*, pages 136–145, 2001.

[59] Jesper Buus Nielsen, Peter Sebastian Nordholt, Claudio Orlandi, and Sai Sheshank Burra. A new approach to practical active-secure two-party computation. In *Advances in Cryptology – CRYPTO 201*, pages 681–700, 2012.

[60] Enrique Larraia, Emmanuela Orsini, and Nigel P. Smart. Dishonest majority multi-party computation for binary circuits. In *Advances in Cryptology – CRYPTO 2014*, pages 495–512, 2014.

[61] Sai Sheshank Burra, Enrique Larraia, Jesper Buus Nielsen, Peter Sebastian Nordholt, Claudio Orlandi, Emmanuela Orsini, Peter Scholl, and Nigel P. Smart. High-performance multi-party computation for binary circuits based on oblivious transfer. *Journal of Cryptology*, 34(3):472, 2021.

[62] Marcel Keller, Emmanuela Orsini, and Peter Scholl. Actively secure ot extension with optimal overhead. In *Advances in Cryptology - CRYPTO 2015*, pages 724–741, 2015.

[63] Donald Beaver. Correlated pseudorandomness and the complexity of private computations. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, pages 479–488, 1996.

[64] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In *Advances in Cryptology - CRYPTO 2003*, pages 145–161, 2003.

[65] Thijs Laarhoven. Sieving for shortest vectors in lattices using angular locality-sensitive hashing. In *Advances in Cryptology – CRYPTO 2015*, pages 3–22, 2015.

[66] Maurício J. Ferreira, Nuno A. Silva, Armando N. Pinto, and Nelson J. Muga. Statistical validation of a physical prime random number generator based on quantum noise. *Applied Sciences*, 13(23), 2023.

[67] Xiu-Bo Chen, Gang Xu, Yi-Xian Yang, and Qiao-Yan Wen. An efficient protocol for the secure multi-party quantum summation. *International Journal of Theoretical Physics*, 49:2793–2804, 2010.

[68] Hao Cao, Wenping Ma, Ge Liu, Liangdong Lü, and Zheng-Yuan Xue. Quantum secure multiparty computation with symmetric boolean functions. *Chinese Physics Letters*, 37(5):050303, 2020.

[69] Hui-Yi Yang and Tian-Yu Ye. Secure multi-party quantum summation based on quantum fourier transform. *Quantum Information Processing*, 17(6):129, 2018.

[70] Run-hua Shi, Yi Mu, Hong Zhong, Jie Cui, and Shun Zhang. Secure multiparty quantum computation for summation and multiplication. *Scientific reports*, 6(1):19655, 2016.

[71] Xiao-Qiu Cai, Tian-Yin Wang, Chun-Yan Wei, and Fei Gao. Cryptanalysis of secure multiparty quantum summation. *Quantum Information Processing*, 21(8):285, 2022.

[72] Robert Raussendorf and Hans J. Briegel. A one-way quantum computer. *Physical Review Letters*, 86:5188–5191, 2001.

[73] Michael A. Nielsen and Christopher M. Dawson. Fault-tolerant quantum computation with cluster states. *Physical Review A*, 71:042323, 2005.

[74] Richard Jozsa. An introduction to measurement based quantum computation. *Quantum Information Processing-From Theory to Experiment*, 199:137–158, 2006.

[75] Hans J. Briegel and Robert Raussendorf. Persistent entanglement in arrays of interacting particles. *Physical Review Letters*, 86:910–913, 2001.

[76] Michael A Nielsen. Quantum computation by measurement and quantum memory. *Physics Letters A*, 308(2-3):96–100, 2003.

[77] Anne Broadbent, Joseph Fitzsimons, and Elham Kashefi. Universal blind quantum computation. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 517–526, 2009.

[78] Peter Bogetoft, Dan Lund Christensen, Ivan Damgard, Martin Geisler, Thomas Jakobsen, Mikkel Krøigaard, Janus Dam Nielsen, Jesper Buus Nielsen, Kurt Nielsen, Jakob Pagter, Michael Schwartzbach, and Tomas Toft. Multiparty computation goes live. Cryptology ePrint Archive, 2008/068, 2008.

[79] Dakshnamoorthy Manivannan, Shafika Showkat Moni, and Sherali Zeadally. Secure authentication and privacy-preserving techniques in vehicular ad-hoc networks (vanets). *Vehicular Communications*, 25:100247, 2020.

[80] Iraklis Symeonidis, Abdelrahaman Aly, Mustafa Asan Mustafa, Bart Mennink, Siemen Dhooghe, and Bart Preneel. Sepcar: A secure and privacy-enhancing protocol for car access provision. In *Computer Security – European Symposium on Research in Computer Security*, pages 475–493, 2017.

[81] Michael Lee and Travis Atkison. Vanet applications: Past, present, and future. *Vehicular Communications*, 28:100310, 2021.

[82] Wouter Heyndrickx, Lewis Mervin, Tobias Morawietz, Noé Sturm, Lukas Friedrich, Adam Zalewski, Anastasia Pentina, Lina Humbeck, Martijn Oldenhof, Ritsuya Niwayama, Peter Schmidtke, Nikolas Fechner, Jaak Simm, Adam Arany, Nicolas Drizard, Rama Jabal, Arina Afanasyeva, Regis Loeb, Shlok Verma, Simon Harnqvist, Matthew Holmes, Balazs Pejo, Maria Telenczuk, Nicholas Holway, Arne Dieckmann, Nicola Rieke, Friederike Zumsande, Djork-Arné Clevert, Michael Krug, Christopher Luscombe, Darren Green, Peter Ertl, Peter Antal, David Marcus, Nicolas Do Huu, Hideyoshi Fuji, Stephen Pickett, Gergely Acs, Eric Boniface, Bernd Beck, Yax Sun, Arnaud Gohier, Friedrich Rippmann, Ola Engkvist, Andreas H. Göller, Yves Moreau, Mathieu N. Galtier, Ansgar Schuffenhauer, and Hugo Ceulemans. Melloddy: Cross-pharma federated learning at unprecedented scale unlocks benefits in qsar without compromising proprietary information. *Journal of Chemical Information and Modeling*, 64(7):2331–2344, 2024.

[83] Haris Smajlović, Ariya Shajii, Bonnie Berger, Hyunghoon Cho, and Ibrahim Numanagić. Sequre: a high-performance framework for secure multiparty computation enables biomedical data sharing. *Genome Biology*, 24(1):5, 2023.

[84] Anh-Tu Tran, The-Dung Luong, Jessada Karnjana, and Van-Nam Huynh. An efficient approach for privacy preserving decentralized deep learning models based on secure multi-party computation. *Neurocomputing*, 422:245–262, 2021.

[85] Suhel Sayyad. Privacy preserving deep learning using secure multiparty computation. In *Proceedings of the 2nd International Conference on Inventive Research in Computing Applications*, pages 139–142, 2020.

[86] Anders Dalskov, Daniel Escudero, and Marcel Keller. Fantastic four: Honest-majority four-party secure computation with malicious security. In *Proceedings of the 30th USENIX Security Symposium*, pages 2183–2200, 2021.

[87] Hanshu Hong and Zhixin Sun. A secure peer to peer multiparty transaction scheme based on blockchain. *Peer-to-Peer Networking and Applications*, 14(3):1106–1117, 2021.

[88] Zhitao Guan, Xiao Zhou, Peng Liu, Longfei Wu, and Wenti Yang. A blockchain-based dual-side privacy-preserving multiparty computation scheme for edge-enabled smart grid. *IEEE Internet of Things Journal*, 9(16):14287–14299, 2022.

[89] Hanrui Zhong, Yingpeng Sang, Yongchun Zhang, and Zhicheng Xi. Secure multi-party computation on blockchain: An overview. In *Parallel Architectures, Algorithms and Programming*, pages 452–460, 2020.

[90] Jun Zhou, Shiying Chen, Kim-Kwang Raymond Choo, Zhenfu Cao, and Xiaolei Dong. Epns: Efficient privacy-preserving intelligent traffic navigation from multiparty delegated computation in cloud-assisted vanets. *IEEE Transactions on Mobile Computing*, 22(3):1491–1506, 2023.

[91] Jun Zhou, Zhenfu Cao, Zhan Qin, Xiaolei Dong, and Kui Ren. Lppa: Lightweight privacy-preserving authentication from efficient multi-key secure outsourced computation for location-based services in vanets. *IEEE Transactions on Information Forensics and Security*, 15:420–434, 2020.

[92] Alghamdi, Wajdi, Salama, Reda, Sirija, M., Abbas, Ahmed Radie, and Dilnoza, Kholmurodova. Secure multi-party computation for collaborative data analysis. *E3S Web of Conferences*, 399:04034, 2023.

[93] Dan Bogdanov, Margus Niitsoo, Tomas Toft, and Jan Willemson. High-performance secure multi-party computation for data mining applications. *International Journal of Information Security*, 11:403–418, 2012.

[94] Ivan Damgård, Kasper Damgård, Kurt Nielsen, Peter Sebastian Nordholt, and Tomas Toft. Confidential benchmarking based on multiparty computation. In *Financial Cryptography and Data Security*, pages 169–187, 2017.

[95] Chi Zhang, Sotthiwat Ekanut, Liangli Zhen, and Zengxiang Li. Augmented multi-party computation against gradient leakage in federated learning. *IEEE Transactions on Big Data*, pages 1–10, 2022.

[96] Assaf Ben-David, Noam Nisan, and Benny Pinkas. Fairplaymp: a system for secure multi-party computation. In *Proceedings of the 15th ACM Conference on Computer and Communications Security*, page 257–266, 2008.

[97] Martin Franz, Andreas Holzer, Stefan Katzenbeisser, Christian Schallhart, and Helmut Veith. Cbmc-gc: An ansi c compiler for secure two-party computations. In *Compiler Construction*, pages 244–249, 2014.

[98] Marcel Keller. Mp-spdz: A versatile framework for multi-party computation. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1575–1590, 2020.

[99] Gadi Aleksandrowicz, Thomas Alexander, Panagiotis Barkoutsos, Luciano Bello, Yael Ben-Haim, David Bucher, Francisco Jose Cabrera-Hernández, Jorge Carballo-Franquis, Adrian Chen, Chun-Fu Chen, Jerry M. Chow, Antonio D. Córcoles-Gonzales, Abigail J. Cross, Andrew Cross, Juan Cruz-Benito, Chris Culver, Salvador De La Puente González, Enrique De La Torre, Delton Ding, ..., and Christa Zoufal. Qiskit: An open-source framework for quantum computing, 2019.

[100] Cirq Developers. Cirq, 2023.

[101] Bar Ilan University Cryptography Research Group. LIBSCAPI - The Secure Computation API.

[102] Abdelrahaman Aly, Karl Cong, Daniele Cozzo, Marcel Keller, Emmanuela Orsini, Dragos Rotaru, Oliver Scherer, Peter Scholl, Nigel P Smart, and Titouan Tanguy. Scale–mamba v1. 14: Documentation. 2021.

[103] Daniel Demmler, Thomas Schneider, and Michael Zohner. Aby - a framework for efficient mixed-protocol secure two-party computation. In *Network and Distributed System Security Symposium*, pages 1–15, 2015.

[104] Alexandra Institute. FRESCO - a FRamework for Efficient Secure COmputation. `https://github.com/aicis/fresco`.

[105] Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. *Communications of the Association for Computing Machinery*, 28(6):637–647, 1985.

[106] Miguel Herrero-Collantes and Juan Carlos Garcia-Escartin. Quantum random number generators. *Reviews of Modern Physics*, 89:015004, 2017.

[107] Davide G. Marangon, Giuseppe Vallone, and Paolo Villoresi. Random bits, true and unbiased, from atmospheric turbulence. *Scientific Reports*, 4(1):5490, 2014.

[108] Pierre Bayon, Lilian Bossuet, Alain Aubert, Viktor Fischer, François Poucheret, Bruno Robisson, and Philippe Maurine. Contactless electromagnetic active attack on ring oscillator based true random number generator. In *Constructive Side-Channel Analysis and Secure Design*, pages 151–166, 2012.

[109] Michael Gude. Concept for a high performance random number generator based on physical random phenomena. *Frequenz*, 39(7-8):187–190, 1985.

[110] Hai-Qiang Ma, Yuejian Xie, and Ling-An Wu. Random number generation based on the time of arrival of single photons. *Applied Optics*, 44(36):7760–7763, 2005.

[111] AE Ivanova, SA Chivilikhin, and AV Gleim. Using of optical splitters in quantum random number generators, based on fluctuations of vacuum. In *Journal of Physics: Conference Series*, volume 735, 2016.

[112] Nelson J Muga, Mário FS Ferreira, and Armando N Pinto. Qber estimation in qkd systems with polarization encoding. *Journal of Lightwave Technology*, 29(3):355–361, 2010.

[113] Nuno A Silva and Armando N Pinto. Effects of losses and nonlinearities on the generation of polarization entangled photons. *Journal of lightwave technology*, 31(8):1309–1317, 2013.

[114] Wen-Ye Liang, Shuang Wang, Hong-Wei Li, Zhen-Qiang Yin, Wei Chen, Yao Yao, Jing-Zheng Huang, Guang-Can Guo, and Zheng-Fu Han. Proof-of-principle experiment of reference-frame-independent quantum key distribution with phase coding. *Scientific Reports*, 4(1):3617, 2014.

[115] P. C. Sun, Y. Mazurenko, and Y. Fainman. Long-distance frequency-division interferometer for communication and quantum cryptography. *Optics Letters*, 20(9):1062–1064, 1995.

[116] Frédéric Bouchard, Duncan England, Philip J. Bustard, Khabat Heshami, and Benjamin Sussman. Quantum communication with ultrafast time-bin qubits. *Physical Review X Quantum*, 3:010332, 2022.

[117] Nicolas Gisin, Grégoire Ribordy, Wolfgang Tittel, and Hugo Zbinden. Quantum cryptography. *Reviews of Modern Physics*, 74(1):145–195, 2002.

[118] Alberto Boaron, Gianluca Boso, Davide Rusca, Cédric Vulliez, Claire Autebert, Misael Caloz, Matthieu Perrenoud, Gaëtan Gras, Félix Bussières, and Ming-Jun Li. Secure quantum key distribution over 421 km of optical fiber. *Physical review letters*, 121(19):190502, 2018.

[119] Stefano Pirandola, Ulrik L Andersen, Leonardo Banchi, Mario Berta, Darius Bunandar, Roger Colbeck, Dirk Englund, Tobias Gehring, Cosmo Lupo, and Carlo Ottaviani. Advances in quantum cryptography. *Advances in Optics and Photonics*, 12(4):1012–1236, 2020.

[120] Mikołaj Lasota, Radim Filip, and Vladyslav C Usenko. Robustness of quantum key distribution with discrete and continuous variables to channel noise. *Physical Review A*, 95(6):062312, 2017.

[121] Paul Jouguet, Sébastien Kunz-Jacques, Anthony Leverrier, Philippe Grangier, and Eleni Diamanti. Experimental demonstration of long-distance continuous-variable quantum key distribution. *Nature Photonics*, 7(5):378–381, 2013.

[122] Duan Huang, Peng Huang, Dakai Lin, and Guihua Zeng. Long-distance continuous-variable quantum key distribution by controlling excess noise. *Scientific reports*, 6(1):1–9, 2016.

[123] Charles H. Bennett, Gilles Brassard, and N. David Mermin. Quantum cryptography without bell's theorem. *Physical Review Letters*, 68:557–559, 1992.

[124] Govind P Agrawal. *Lightwave technology: components and devices*, volume 1. John Wiley & Sons, 2004.

[125] Elaine Barker. *Recommendation for key management: part 1 - general*. National Institute of Standards and Technology, 2020.

[126] ETSI. Quantum key distribution (qkd); components and internal interfaces. Technical Report ETSI GS QKD 004 V2.1.1, European Telecommunications Standards Institute, 2020.

[127] Niv Gilboa. Two party rsa key generation. In *Advances in Cryptology - CRYPTO '99*, volume 1666, pages 116–129, 1999.

[128] Tung Chou and Claudio Orlandi. The simplest protocol for oblivious transfer. In *Progress in Cryptology – LATINCRYPT 2015*, pages 40–58, 2015.

[129] Ivan Damgård and Claudio Orlandi. Multiparty computation for dishonest majority: From passive to active security at low cost. In *Advances in Cryptology – CRYPTO 2010*, pages 558–576, 2010.

[130] Donald Beaver. Efficient multiparty protocols using circuit randomization. In *Advances in Cryptology — CRYPTO '91*, pages 420–432, 1992.

[131] Diogo Matos. QMP-SPDZ: Quantum Multi-Party Computation with SPDZ Framework. `https://github.com/diogoftm/QMP-SPDZ`, 2025. Accessed: 2025-01-03.

[132] Joan Daemen and Vincent Rijmen. Aes proposal: Rijndael, 1999.

[133] Eslam G. AbdAllah, Yu Rang Kuang, and Changcheng Huang. Advanced encryption standard new instructions (aes-ni) analysis: Security, performance, and power consumption. In *Proceedings of the 12th International Conference on Computer and Automation Engineering*, pages 167–172, 2020.

[134] Daniel J Bernstein and Frank Denis. Libsodium-a modern, portable, easy to use crypto library, 2019.

[135] Diogo Matos. Minimal etsi qkd 004. `https://github.com/diogoftm/minimal-etsi-qkd-004/tree/main`, 2025. Accessed: 2025-01-23.

[136] Wanxin Li, Hao Guo, Mark Nejad, and Chien-Chung Shen. Privacy-preserving traffic management: A blockchain and zero-knowledge proof inspired approach. *IEEE Access*, 8:181733–181743, 2020.

[137] Jakub Konečný, H. McMahan, Felix Yu, Peter Richtárik, Ananda Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. In *Neural Information Processing Systems Workshop on Private Multi-Party Machine Learning*, 2016.

[138] Aljoša Smajić, Melanie Grandits, and Gerhard F Ecker. Privacy-preserving techniques for decentralized and secure machine learning in drug discovery. *Drug Discovery Today*, 28(12):103820, 2023.

[139] Xia Xiao, Xiaoqi Wang, Shengyun Liu, and Shaoliang Peng. Mpcddi: A secure multi-party computation-based deep learning framework for drug-drug interaction predictions. In *Bioinformatics Research and Applications*, pages 263–274, 2022.

[140] Till Gehlhar, Felix Marx, Thomas Schneider, Ajith Suresh, Tobias Wehrle, and Hossein Yalame. SafeFL: MPC-friendly framework for private and robust federated learning. In *Proceedings of the IEEE Security and Privacy Workshops*, pages 69–76, 2023.

[141] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge Luis Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. In *The European Symposium on Artificial Neural Networks*, 2013.

[142] Haixia Peng, Le Liang, Xuemin Shen, and Geoffrey Ye Li. Vehicular communications: A network layer perspective. *IEEE Transactions on Vehicular Technology*, 68(2):1064–1078, 2019.

[143] Maxim Raya and Jean-Pierre Hubaux. The security of vehicular ad hoc networks. In *Proceedings of the 3rd ACM Workshop on Security of Ad Hoc and Sensor Networks*, pages 11–21, 2005.

[144] Carlo Ottaviani, Matthew J. Woolley, Misha Erementchouk, John F. Federici, Pinaki Mazumder, Stefano Pirandola, and Christian Weedbrook. Terahertz quantum cryptography. *IEEE Journal on Selected Areas in Communications*, 38(3):483–495, 2020.

[145] Neel Kanth Kundu, Soumya P. Dash, Matthew R. McKay, and Ranjan K. Mallik. Mimo terahertz quantum key distribution. *IEEE Communications Letters*, 25(10):3345–3349, 2021.

[146] Robert Bedington, Juan Mantilla, and Alexander Ling. Progress in satellite quantum key distribution. *npj Quantum Information*, 3, 2017.

[147] Imran Khan, Bettina Heim, Andreas Neuzner, and Christoph Marquardt. Satellite-based qkd. *Optics and Photonics News*, 29(2):26–33, 2018.

[148] Sheng-Kai Liao, Wen-Qi Cai, Wei-Yue Liu, Liang Zhang, Yang Li, Ji-Gang Ren, Juan Yin, Qi Shen, Yuan Cao, and Zheng-Ping Li. Satellite-to-ground quantum key distribution. *Nature*, 549(7670):43–47, 2017.

[149] Joseph A DiMasi, Ronald W Hansen, and Henry G Grabowski. The price of innovation: new estimates of drug development costs. *Journal of Health Economics*, 22(2):151–185, 2003.

[150] Peter McHale. Predicting molecule solubility using graph neural networks. `https://github.com/petermchale/gnn`, 2025.

[151] John S. Delaney. Esol: Estimating aqueous solubility directly from molecular structure. *Journal of Chemical Information and Computer Sciences*, 44(3):1000–1005, 2004.

[152] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

[153] Xiaoyu Cao, Minghong Fang, Jia Liu, and Neil Zhenqiang Gong. Fltrust: Byzantine-robust federated learning via trust bootstrapping. *arXiv preprint arXiv:2012.13995*, 2020.

[154] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE Symposium on Security and Privacy*, pages 691–706, 2019.

[155] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. Local model poisoning attacks to byzantine-robust federated learning. In *Proceedings of the 29th USENIX Conference on Security Symposium*, pages 1605–1622, 2020.

[156] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE Symposium on Security and Privacy*, pages 739–753, 2019.

[157] Yaniv Ben-Itzhak, Helen Möllering, Benny Pinkas, Thomas Schneider, Ajith Suresh, Oleksandr Tkachenko, Shay Vargaftik, Christian Weinert, Hossein Yalame, and Avishay Yanai. ScionFL: Efficient and robust secure quantized aggregation. In *IEEE Conference on Secure and Trustworthy Machine Learning*, pages 490–511, 2024.

[158] Virat Shejwalkar and Amir Houmansadr. Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning. In *Network and Distributed System Security Symposium*, pages 1–19, 2021.

[159] Yan-bing Li, Qiao-yan Wen, and Su-juan Qin. Comment on "secure multiparty computation with a dishonest majority via quantum means". *Physical Review A*, 84:016301, 2011.

[160] Yan-Bing Li, Qiao-Yan Wen, and Su-Juan Qin. Improved secure multiparty computation with a dishonest majority via quantum means. *International Journal of Theoretical Physics*, 52:199–205, 2013.

[161] Janet Anders and Dan E. Browne. Computational power of correlations. *Physical Review Letters*, 102:050502, 2009.

[162] David Mermin. Quantum mysteries revisited. *American Journal of Physics*, 58(8):731–734, 1990.

[163] Vedran Dunjko, Theodoros Kapourniotis, and Elham Kashefi. Quantum-enhanced secure delegated classical computing. *Quantum Info. Comput.*, 16(1–2):61–86, 2016.

[164] Wim van Dam. Implausible consequences of superstrong nonlocality. *Natural Computing*, 12(1):9–12, 2012.

[165] Hoi-Kwong Lo. Insecurity of quantum secure computations. *Physical Review A*, 56:1154–1162, 1997.