UNIVERSIDADE DE LISBOA
INSTITUTO SUPERIOR TÉCNICO

Quantum assisted Secure Multiparty Computation

Manuel Batalha dos Santos

Supervisor: Doctor Paulo Alexandre Carreira Mateus
Co-Supervisor: Doctor Armando Nolasco Pinto

Thesis specifically prepared to obtain the PhD Degree in Mathematics

Draft

January 2023

Abstract

Start with no indent.

Then you can write another paragraph.

Key-words: quantum cryptography, quantum oblivious transfer, quantum obliious linear evaluation, secure multiparty computation.

Resumo

Escrever a mesma coisa que está no Abstract, mas em Português.

Palavras-chave: criptografia quântica, passeios quânticos, memórias quânticas, transições de fase topológicas, estados de fronteira

Acknowledgments

Write the acknowledgments here.

I acknowledge Fundação para a Ciência e a Tecnologia (FCT, Portugal) for its support through the PhD grant SFRH/BD/ 144806/2019 in the context of the Doctoral Program in the Information Security (IS). I also acknowledge support from SQIG (Security and Quantum Information Group) in the Instituto de Telecomunicações (IT), Lisbon, namely through UID/EEA/50008/2013. (CHECK THIS)

I dedicate this thesis to my loving wife Teresinha and my two children Henrique and Helena who came to life during this journey to help me finish it.

Contents

List of Figures

List of Tables

List of Abbreviations

A – Alice

B – Bob

BCS – Bardeen-Cooper-Schrieffer

BG – Boltzmann-Gibbs

CS – Chiral symmetry

DTQW – Discrete-time quantum walk

DQPT – Dynamical quantum phase transition

E – Eve

EB – Entanglement based

LE – Loschmidt Echo

MDM – Massive Dirac model

PHS – Particle-hole symmetry

PT – Phase transition

PM – Prepare and measure

SSH – Su-Schrieffer-Heeger

TI – Topological insulator

TSC – Topological superconductor

TRS – Time-reversal symmetry

QKD – Quantum Key Distribution

QW – Quantum walk

Chapter 1

Introduction

The emerging fields of Data Mining and Data Analysis have deeply benefited from the increasing power of computers [5]. However, its need for a massive and methodical collection of data can lead to the complete or partial leak of private sensitive information, such as in the case of the genomics field [6–9]. As a consequence, the aggregation of data from different sources is most of the times blocked due to legally imposed regulations such as the General Data Protection Regulation (GDPR) [10]. Although this has the benefit of protecting people's privacy, it also has the downside of preventing honest players from accessing data necessary to tackle some of the most important issues in our society.

Secure Multiparty Computation

To overcome the privacy-related issues described above, several privacy-enhancing technologies have been proposed [11–13]. One important area of research is Secure Multiparty Computation (SMC). This technology allows a set of n parties P_i to jointly compute some function $f(x_1, ..., x_n) = (y_1, ..., y_n)$ without disclosing their inputs to the other parties. The security requirements of SMC are equivalent to an ideal case, where every party P_i sends his inputs to some independent and trusted third party, who computes $f()$ and sends back to each party their corresponding output.

Since Yao seminal work [14], several SMC protocols have been developed, rendering different framework implementations [15–17]. However, they can generally be separated into two types according to the circuit logic being used: boolean or arithmetic. In each case, the efficiency and security of SMC heavily rely on the efficiency and security of important cryptographic primitives. Boolean-based SMC protocols rely on Oblivious Transfer (OT) [18] and arithmetic-based rely on Oblivious Linear Evaluation (OLE) [19]. Impagliazzo and Rudich [20] proved that both OT and OLE protocols require public cryptography and cannot just rely on symmetric cryptography. This is an unfortunate result both from an efficiency and security perspective. Indeed, symmetric cryptography is lighter than asymmetric cryptography and requires less computational assumptions. Moreover, with the emergence of quantum computers, Shor's algorithm [21] jeopardizes all the current public-key methods based on RSA, Elliptic Curves or Diffie-Hellman, in which many OT and OLE implementations rely on. This puts at risk the deployment of classical OT and OLE, which ultimately leads to the exposure of the SMC parties' private inputs. Thus, it is essential to develop SMC methods secure against quantum computers while not compromising state-of-the-art performance levels.

A Quantum Era

We are now in the beginning of what is known to be the second quantum revolution. Quantum technology has evolved to a point where we can integrate quantum exotic features into complex engineering systems. Most of the applications lie in the field of quantum cryptography, where one thrives to find protocols that offer some advantage over their classical counterparts. As analysed in [22, 23], these advantages can be of two types:

1. Improve the security requirements, rendering protocols that are information-theoretically secure or require fewer computational assumptions;
2. Achieve new primitives that were previously not possible just with classical techniques.

Despite the most famous use-case of quantum cryptography being quantum key distribution (QKD), other primitives play an important role in this quest. Some examples of these cryptographic tasks are bit commitment [24], coin flipping [25], delegated quantum computation [26], position verification [27], and password-based identification [28, 29].

Also, the intrinsic randomness provided by quantum phenomena is an ideal resource to develop quantum communication protocols for oblivious transfer (OT) [30]. Remarkably, there is a distinctive difference between classical and quantum OT from a security standpoint, as the latter is proved to be possible assuming only the existence of quantum-hard one-way functions [31, 32]. This means quantum OT can be based only on symmetric cryptography, requiring weaker security assumptions than classical OT. Moreover, these quantum protocols frequently have a desirable property that guarantees information-theoretic security after the execution of the protocol. This property is commonly called everlasting security. This greatly improves the security of SMC protocols, allowing them to have their security based on symmetric cryptography alone and with this important feature of everlasting security. Regarding oblivious linear evaluation (OLE) primitive, it is known that it can be reduced to OT [33] through classical methods that do not require further assumptions. Therefore, it seems natural to use quantum OT to generate quantum-secure OLE instances.

Contributions and Outline

Despite the many advances, the adoption of quantum cryptography by secure multiparty computation (SMC) systems is still reduced. This is due to the efficiency challenges imposed by quantum technology and the need of high throughput of both OT and OLE primitives in boolean- and arithmetic-based SMC, respectively.

The overall goal of this dissertation is to give one step closer to the adoption of quantum cryptography by SMC systems. We do this with three contributions. In our first contribution, we start the studying of comparing the efficiency of both classical and quantum protocols. Our second contribution is the first quantum OLE protocol which does not rely on OT. Our last contribution is an implementation of a special-purpose SMC system applied to genomics analysis assisted with quantum OT. Along the way, we produced a review dedicated to quantum OT protocols alone. Usually, its analysis is integrated into more general surveys under the topic of "quantum cryptography", leading to a less in-depth exposition of the topic.

We describe the contributions in a bit more detail.

Efficiency of classical and quantum OT protocols. To the best of our knowledge, there is no comparative study on the efficiency of quantum and classical approaches. This is mainly caused by two reasons. From a theoretical perspective, the use of different types of information (quantum and classical) makes it difficult to make a fair comparison based on the protocols' complexity. Also, from a practical standpoint, there is still a discrepancy in the technological maturity between quantum and classical techniques. Quantum technology is still in its infancy, whereas classical processors and communication have many decades of development.

Despite these constraints, we compare the complexity and operations efficiency of classical and quantum protocols. To achieve this, we realize that both classical and quantum protocols can be divided into two phases: offline and online. The offline phase is characterize by the fact that it is independent of the parties' inputs. This means that, from a practical point-of-view, this phase produces the resources necessary to use during the online phase, where we take into consideration the parties' inputs. It can be argued that the offline phase is not so hungry-efficient as the online phase. As a consequence, for comparison purposes, we can focus on the online phase. Fortunately, the online phase of quantum OT is solely based on classical communications. Therefore, it is possible and fair to compare the online phase of both classical and quantum protocols.

We make a detailed comparison between the complexity of the online phase of two state-of-the-art classical OT protocols [3, 4] and an optimized quantum OT protocol. We conclude that the online phase of quantum OT competes with its classical counterparts and has the potential to be more efficient.

Convetion issue: use precomputation phase instead of offline; use transfer phase instead of online.

Quantum oblivious linear evaluation protocol. Our second contribution is a quantum protocol for OLE with quantum universally composable (quantum-UC) security in the $\mathcal{F}_{\mathbf{COM}}-$hybrid model, i.e. when assuming the existence of a commitment functionality, $\mathcal{F}_{\mathbf{COM}}$. To obtain a secure protocol, we take advantage of the properties of Mutually Unbiased Bases in high-dimensional Hilbert spaces with prime and prime-power dimension. Such a choice is motivated by recent theoretical and experimental advances that pave the way for the development and realization of new solutions for quantum cryptography [34–38].

To the best of our knowledge our protocol is the first quantum-UC secure quantum OLE proposal. Moreover, it is not based on any quantum OT implementation which would be the standard approach. We consider the static corruption adversarial model with both semi-honest and malicious adversaries. We develop a weaker version of OLE, which may be of independent interest. We also modify the proposed protocol to generate quantum-UC secure vector OLE (VOLE). We give bounds on the possible size of VOLE according to the security parameters.

Quantum assisted secure multiparty computation. Individuals' privacy and legal regulations demand genomic data be handled and studied with highly secure privacy-preserving techniques. In this contribution, we propose a feasible secure multiparty computation (SMC) system assisted with quantum cryptographic protocols that is designed to compute a phylogenetic tree from a set of private genome sequences. This system adapts several distance-based methods (Unweighted Pair Group Method with Arithmetic mean, Neighbour-Joining, Fitch-Margoliash) into a private setting where the sequences owned by each party are not disclosed to the other members present in the protocol. We do not apply a generic implementation of SMC to the problem of phylogenetic trees. Instead, we develop a tailored private protocol for this use case in order to improve efficiency.

We theoretically evaluate the performance and privacy guarantees of the system through a complexity analysis and security proof and give an extensive explanation about the implementation details and cryptographic protocols. We also implement a quantum-assisted secure phylogenetic tree computation based on the Libscapi implementation of the Yao protocol, the PHYLIP library and simulated keys of two quantum systems: quantum oblivious key distribution and quantum key distribution.[1]. This demonstrates its effectiveness and practicality. We benchmark our implementation against a classical-only solution and we conclude that both approaches render similar execution times. The only difference between the quantum and classical systems is the time overhead taken by the oblivious key management system of the quantum-assisted approach.

The results are presented as follows. We start presenting SMC protocols based on OT and OLE at Chapter 2. Then, at Chapter 3 we introduce some quantum information concepts and security definitions used throughout the thesis. Chapter 4 is devoted to quantum oblivious transfer protocols. Then, in Chapter 5 we compare classical and quantum approaches for OT. In Chapter 6 we present out quantum OLE protocol along with its security proof. Finally, in Chapter 7, we presented our implementation of quantum-assisted SMC system applied to phylogeny analysis. rewrite the chapters

Published research

This thesis is based on research published in various journals. During my PhD I was involved in the following projects.

- [39] Manuel B. Santos, Paulo Mateus, and Armando N. Pinto. "Quantum Oblivious Transfer: A Short Review". In: Entropy 24.7 (July 2022), p. 945.
- [40] Manuel B. Santos, Armando N. Pinto, and Paulo Mateus. "Quantum and classical oblivious transfer: A comparative analysis". In: IET Quantum Communication 2.2 (May 2021), pp. 42–53.
- [41] Manuel B. Santos, Paulo Mateus, and Chrysoula Vlachou. Quantum Universally Composable Oblivious Linear Evaluation. 2022. DOI: 10.48550/ARXIV.2204.14171. Poster at QCrypt2022.
- [42] Manuel B. Santos et al. "Private Computation of Phylogenetic Trees Based on Quantum Technologies". In: IEEE Access 10 (2022), pp. 38065–38088.
- [43] Manuel B. Santos et al. "Quantum Secure Multiparty Computation of Phylogenetic Trees of SARS-CoV-2 Genome". In: 2021 Telecoms Conference (ConfTELE). IEEE, Feb. 2021.

[1] The code can be accessed at the following repo: https://github.com/manel1874/QSHY/tree/dev-cq-phylip

- [44] Armando N. Pinto et al. “Quantum Enabled Private Recognition of Composite Signals in Genome and Proteins”. In: 2020 22nd International Conference on Transparent Optical Networks (ICTON). IEEE, July 2020.

Chapter 4 is based on [39]. Chapter 5 is based on the work developed on both [40] and [42]. Chapter 6 presents all the results from [41]. Finally, Chapter 7 is the combination of [42–44]

Chapter 2

Technical overview

In this chapter, the basic mathematical and information-theoretical concepts used throughout this thesis are presented. The chapter is divided into four main sections. We start with a brief overview on some same mathematical notation elements. Then, the basic formalism of quantum information and the universal composability framework in the quantum setting are introduced. Finally, an informal description of secure multiparty computation is given.

2.1 Mathematical preliminaries

We denote by $\gcd(a, b), a, b \in \mathbb{Z}$ the greatest common divisor between integers a and b. $\mathbb{Z}_q$ denotes the set of integers $a \mod q$. $\mathbb{Z}_q^*$ is the set of integers $a \in \mathbb{Z}_q$ that are coprime with q, i.e. $\gcd(a, q) = 1$. For q prime, $\mathbb{Z}_q^*$ forms a multiplicative group of order $q-1$ and $\mathbb{Z}_q$ a finite field of order q. A generator g of some multiplicative group $\mathbb{G}$ is an element in $\mathbb{G}$ such that $\forall\, a \in \mathbb{G}, \exists r : g^r = a$. The discrete logarithm base g of some element $a \in \mathbb{G}$, denoted by $\log_g a$, is the power r of g such that $g^r = a$.

We use $|I|$ to denote the size of a set I. We use the notation $s \leftarrow_\$ I$ to describe a situation where an element s is drawn uniformly at random from the set I. Vectors $\boldsymbol{v} = (v_1, \ldots, v_n)$ are denoted in bold. Given a set J, $\boldsymbol{v}_{|J}$ denotes the subvector of $\boldsymbol{v}$ restricted to the indices $i \in J$. Let $\mathbb{Z}_q$ denote the finite field with order q. Then, for $m \in \mathbb{Z}_q$, $[m]$ is the ordered set $\{1, 2, \ldots, m\}$. Also, for $m, n \in \mathbb{Z}_q$ such that $m < n$, $[m, n] = \{m, m+1, \ldots, n-1, n\}$. For $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}_q^n$, $r_H(\boldsymbol{x}, \boldsymbol{y}) = d_H(\boldsymbol{x}, \boldsymbol{y})/n$ is the relative Hamming distance, with the Hamming distance given by $d_H(\boldsymbol{x}, \boldsymbol{y}) = |\{i : x_i \neq y_i\}|$.

We use the big-$\mathcal{O}$ notation to denote the fastest-growing term of the number of operations with respect to some security parameter n. A negligible function $\mu(n)$ is a function such that $\mu(n) < 1/p(n)$ for some polynomial $p(n)$ and sufficiently large n.

2.2 Secure multiparty computation

Let us consider a scenario with n parties, P_i, each with input x_i, $i \in \{1, ..., n\}$. Simply put, secure multiparty computation (SMC) allows these n parties to jointly compute some function, $f(x_1, ..., x_n) = (y_1, ..., y_n)$, without revealing their inputs. The only information received by party i is the output y_i of the function $f()$, which, depending on the function, may reveal some information about the parties' inputs. This functionality is designed to be equivalent to a scenario where every party, P_i, sends his inputs, x_i, to some independent and trusted third party, P_{TTP}, who computes $f(x_1, ..., x_n)$ and sends back to each party their corresponding output, y_i.

We stress that SMC may not fully hide the inputs of the parties, even with a perfectly maliciously secure protocol. This comes from the security guarantees of the ideal scenario. Indeed, there could be the case where a perfectly legitimate SMC protocol (e.g. using a trusted third party) leaks all the inputs of the parties. This happens when one of the parties can use his outputs and inputs to invert the function $f()$. We illustrate this case with the following example. Imagine that two people want to compute the average of their weight. It is straightforward to see that both parties can use their weight and the average value to compute the other party's weight. This always happens whenever $f()$ is bijective with the adversaries' inputs fixed. In this scenario, SMC does not improve the privacy of the computation.

The following are some properties of SMC which we informally describe:

1. Correctness: if all the parties do not deviate from the protocol, the protocol evaluates the correct output according to $f()$ and the parties' inputs $x_1, \ldots, x_n$;

2. Passive security: even if the adversaries do not deviate from the protocol, they do not learn the inputs of the honest parties. Throughout this thesis, we refer to adversaries who do not deviate from the protocol as semi-honest parties. In the literature, these are also called honest-but-curious adversaries.

3. Active security: even if the adversaries deviate arbitrarily from the protocol (dishonest parties), they do not learn the inputs of the honest parties. In active security there are two types of protocols that react differently with respect to the adversarial behaviour. They can be robust against the advarsaries, meaning the honest parties will still receive the correct answer. Or honest parties abort the protocol when there exists some malicious activity.

Regarding the corruption strategy of the adversaries, they can be of two types: static or adaptive. Static security guarantees that the protocol is secure against an adversary who only corrupts parties before the execution of the protocol. Adaptive security is an harder property to attain as it assumes that the adversary can choose throughout the protocol the party to be corrupted. Also, it is interesting to note that there is a fundamental difference between the adversarial structure of encryption methods and SMC methods. In encryption methods, the adversary is not part of the protocol. It is considered as an external party (usually called Eve) that interfeers with the communication between the protocol parties. In the case of SMC methods, the adversaries are a subset of the protocol parties.

Next, we present two common approaches used for SMC protocols: garbled circuit and secret sharing approaches. The garbled circuit approach is based on boolean circuits and follows from the techniques developed by Yao [13]. The secret sharing approach is commonly based on arithmetic circuits (although it can also be used with boolean circuits) and follows from the properties of secret sharing [45, 46]. We note that throughout this thesis we will focus on two-party protocols. For this reason, we name these parties Alice and Bob. Also, we follow the convention that Alice plays the role of the protocol's sender and Bob plays the role of the receiver.

2.2.1 Garbled circuit approach

The garbled circuit approach is based on Yao's seminal work [13], who proposed a technique to "encrypt" boolean circuits in such a way that preserves the security requirements of both parties. This "encrypted" version is called garbled circuit and we present it in this section along with the Yao protocol description. Usually, this approach suits best scenarios with higher latency. This comes from the fact that it typically requires a fixed number communication rounds, independently from the complexity of the function to be evaluated. On the other hand, for large circuits one needs high bandwidth [47].

Before looking at the Yao protocol, there is one crucial primitive that has to be presented: oblivious transfer.

Oblivious transfer

The study of oblivious transfer (OT) has been very active since its first proposal in 1981 by Rabin [48]. The importance of OT comes from its wide number of applications. More specifically, one can prove that OT is equivalent to the secure two-party computation of general functions [14, 18], i.e. one can implement a secure two-party computation using OT as its building block. Additionally, this primitive can also be used for secure multi-party computation (SMC) [33], private information retrieval [49], private set intersection [50], and privacy-preserving location-based services [51].

The OT functionality can be presented in many flavours. In this thesis, when we refer

$\mathcal{F}_{\mathbf{OT}}$ functionality

- **Input phase:** Alice sends $(m_0, m_1) \in \{0,1\}^l$ (two messages) to $\mathcal{F}_{\mathbf{OT}}$ and Bob sends $b \in \{0,1\}$ (bit choice) to $\mathcal{F}_{\mathbf{OT}}$.
- **Output phase:** Alice receives nothing $\perp$ from the functionality and Bob receives m_b.

Figure 2.1: OT functionality.

to OT, we mean the 1-out-of-2 OT that is specified in Figure 2.1. Consequently, we have that OT must satisfy the following security requirements:

- Concealing: Alices knows nothing about Bob's bit choice b.
- Obliviousness: Bob knows nothing about the message $m_{b\oplus 1}$.

OT can be generalized to the case of k-out-of-N OT, where Alice owns N messages, and Bob can choose k of them. For $k = 1$, this is commonly called private database query (PDQ). Also, we call random OT when both parties' inputs are random.

Yao protocol

A solution to SMC was given for the first time by Yao [13] and its main idea resides on the fact that every function has a boolean circuit representation. From this, Yao developed the concept of garbled circuits which is one of the key elements for secure computation. The Yao's garbled circuit protocol is constrained to only two parties but its generalization was achieved by GMW [15] and BMR [52]. Also, some implementation optimizations on Yao protocol were later developed in order to improve its performance: point-and-permute [52], row reduction [53, 54], FreeXOR [55] and half gates [56].

As we said before, the main idea of Yao protocol is to represent the desired function $f()$ as a boolean circuit C, i.e. by a sequence of logical gates interconnected with wires. After the generation of the circuit C, each party will have two very different roles. Generally speaking, Alice (usually called garbler) randomly generates keys to each input bit, encrypts each circuit's gate and sends both elements to Bob (called evaluator). This procedure masks Alice's inputs from Bob. Then, through the OT functionality, Bob receives the keys corresponding to his input bits. So, OT allows to mask Bob's inputs from Alice. Finally, since the evaluator has all the input keys, he can decrypt every gate, i.e. evaluate the circuit. Let us see in more detail how the protocol works using a four input boolean circuit description of the Millionaires' problem. This problem can be described by the following expression:

$$f(a,b) = \begin{cases} 1 & \text{if } a > b, \\ 0 & \text{otherwise,} \end{cases} \tag{2.1}$$

for $a, b \in \{0,1\}^2$. In summary, it allows two parties to discover who has the largest value without revealing them.

The protocol goes as follows:

1. *Circuit generation:* The garbler Alice generates a boolean circuit of function (2.1):

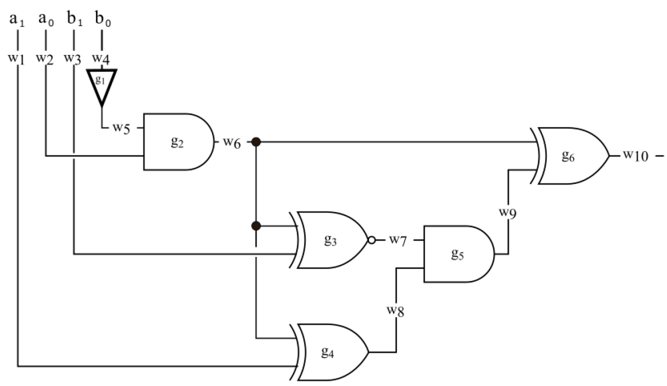

Figure 2.2: Boolean circuit of the Millionaires' Problem. Optimized circuit according to the construction in [1].

   In this case, the circuit contains one NOT gate (g_1), two AND gates (g_2, and g_5), two XOR gate (g_4 and g_6), one XNOR gate (g_3) and four input wires (w_1 and w_2 belonging to Alice and w_3 and w_4 to Bob).

2. *Wire encryption:* Alice uses a random number generator to generate two keys k_i^0 and k_i^1 for each wire w_i, $i \in \{1, ..., 10\}$. These keys correspond to the possible values (0 or 1) on the wire. Note that this is done to prevent Bob from knowing the true value of the wires during the evaluation process.

3. *Gate encryption:* For every gate g_l in the circuit with corresponding input wires w_i and w_j and output wire w_s, Alice creates the following table:

$\mathsf{Enc}_{k_i^0}\left(\mathsf{Enc}_{k_j^0}\left(k_s^{g_l(0,0)}\right)\right)$
$\mathsf{Enc}_{k_i^0}\left(\mathsf{Enc}_{k_j^1}\left(k_s^{g_l(0,1)}\right)\right)$
$\mathsf{Enc}_{k_i^1}\left(\mathsf{Enc}_{k_j^0}\left(k_s^{g_l(1,0)}\right)\right)$
$\mathsf{Enc}_{k_i^1}\left(\mathsf{Enc}_{k_j^1}\left(k_s^{g_l(1,1)}\right)\right)$

where $g_l(a,b)$ is the output of gate g_l for inputs $a,b \in \{0,1\}$. So, we could think of each row as a locked box that requires two keys to be opened. If the two correct keys are used, it outputs the key corresponding to the desired output value given by g_l. After encrypting each gate, Alice permutes the rows of the corresponding table, otherwise, it would be easy to know the real value of the input keys. Then, she sends to Bob the garbled tables along with Alice's input keys.

As an example, we can easily see that if we use input keys k_i^0 and k_j^1 (corresponding to real values 0 and 1), we would only be able to decipher the second row of the table, $\mathsf{Enc}_{k_i^0}(\mathsf{Enc}_{k_j^1}(k_s^{g_l(0,1)}))$, and get $k_s^{g_l(0,1)}$.

4. *Oblivious Transfer:* At this stage of the protocol, the evaluator Bob knows the garbled circuit and Alice's input keys but he does not know the keys corresponding to his real inputs. However, since Bob wants to keep his input value private he cannot directly ask for those keys. At this point, the OT functionality enables the evaluator to receive his input keys without compromising neither the evaluator's nor garbler's security. In fact, for every input wire, both parties perform an OT where Alice plays the role of the sender and Bob plays the role of the receiver.

   Let us assume Alice's input keys to be k_1^0 and k_2^1 (corresponding to the real value 01) and Bob's input bits to be 11. This means that Bob must use the respective input keys (k_3^1 and k_4^1) in order to correctly evaluate the circuit. So, they will execute two OT protocols where:

   - Alice inputs: (k_3^0, k_3^1) and (k_4^0, k_4^1);
   - Bob inputs: $b_1 = 1$ and $b_2 = 1$.

5. *Evaluation:* Once the evaluator has all the necessary elements, he can proceed with the circuit evaluation. In this step, he simply has to decipher the correct rows of the garbled tables sent by Alice with the corresponding keys. Since the rows of the tables are shuffled, the evaluator does not know which row is the correct one. This small issue can be solved by simple techniques (Point-and-Permute or encryption with a certain number of 0 padded) which, for the sake of brevity, we will not explore here. At the end of the evaluation, the evaluator receives the key that corresponds to the result. Finally, the evaluator sends the resulting key to the garbler and the garbler tells him the final bit.

   According to our Millionaires' problem, the evaluation yields the following results for $a = 01$ and $b = 11$: $g_1(k_4^1) = k_5^0$, $g_2(k_5^0, k_2^1) = k_6^0$, $g_3(k_6^0, k_3^1) = k_7^0$, $g_4(k_6^0, k_1^0) = k_8^1$, $g_5(k_7^0, k_8^1) = k_9^0$, $g_6(k_6^0, k_9^0) = k_{10}^0$. Actually, the desired result is 0.

The Yao protocol has its security based on two main building blocks: garbled circuits and oblivious transfer. Although garbled circuits can be generated with symmetric encryption (i.e. using double AES encryption), OT protocols cannot be classically achieved with symmetric cryptography alone [57]. Thus, it is crucial to find efficient protocols for a quantum-resistant OT.

2.2.2 Secret sharing approach

The secret sharing approach was initiated by two concurrent works, known as BGW [45] and CCD [46]. This approach does not generate an "encrypted" version of the circuit. Instead, the parties use some secret sharing scheme in order to evaluate the circuit. In this approach we use simple operations (addition and multiplication) but the communication rounds will depend on the size of circuit being evaluated. We start by defining an important primitive for secret sharing based protocols: oblivious linear evaluation.

Oblivious linear evaluation

Oblivious linear evaluation (OLE) can be seen as a generalization of oblivious transfer (OT) [48]. OT has been shown to be complete for secure multi-party computation [18], i.e., any such task, including OLE, can be achieved given an OT implementation. A compelling reason to study OLE protocols is that they can serve as building blocks for the secure evaluation of arithmetic circuits [58–61], just like OT allows the secure evaluation of boolean circuits [62]. Specifically, OLE can be used to generate multiplication triples which are the basic tool for securely computing multiplication gates [61]. Besides that, OLE has applications in more tasks for two-party secure computation [63–67] and private set intersection [68].

$\mathcal{F}_{\mathbf{OLE}}$ functionality

- **Input phase:** Alice sends $(a, b) \in \mathbb{Z}_d^2$ (two field elements) to $\mathcal{F}_{\mathbf{OLE}}$ and Bob sends $x \in \mathbb{Z}_d$ to $\mathcal{F}_{\mathbf{OLE}}$.
- **Output phase:** Alice receives nothing $\perp$ from the functionality and Bob receives $f(x) := ax + b$.

Figure 2.3: OLE functionality.

The OLE functionality specification is presented in Figure 2.3. Similarly, we have that OLE must satisfy the following security requirements:

- Concealing: Alices knows nothing about Bob's field element x.
- Obliviousness: Bob knows nothing about the function $f()$ other than its evaluation at x, i.e. $f(x)$.

We can also generalize the OLE functionality to a vectorized version. The vector OLE (VOLE) functionality is presented in Figure 2.4. Note that Bob only inputs one field element x and Alice inputs two vectors.

$\mathcal{F}_{\mathbf{VOLE}}$ functionality

- **Input phase:** Alice sends $(\boldsymbol{a}, \boldsymbol{b}) \in \mathbb{Z}_d^{2n}$ (two vectors of field elements) to $\mathcal{F}_{\mathbf{VOLE}}$ and Bob sends only $x \in \mathbb{Z}_d$ to $\mathcal{F}_{\mathbf{VOLE}}$.
- **Output phase:** Alice receives nothing $\perp$ from the functionality and Bob receives $\boldsymbol{f}(x) := \boldsymbol{a}x + \boldsymbol{b}$.

Figure 2.4: VOLE functionality.

Basic operations

To highlight the importance of OLE in secret sharing based SMC protocols, we go through a passively secure protocol [69]. We consider the two party case (Alice and Bob) where the parties own additive shares of the secret. So, for some secret value x, Alice owns x_A, Bob owns x_B and $x = x_A + x_B$. The operations used in the protocol according to the circuit are as follows:

- **Input**. For Alice to secret share her input value x, she randomly chooses x_B and sends it to Bob. Alice defines x_A as $x_A = x - x_B$;
- **Addition**. There are two scenarios to consider:
  - **Scalar**. For Alice and Bob to add a scalar to a secret x ($z = a + x$), Alice computes $z_A = a + x_A$ and Bob sets $z_B = x_B$.
  - **Shares**. For Alice and Bob to add secrets x and y ($z = x+y$), they individually add their corresponding shares, i.e. $z_A = x_A + y_A$ and $z_B = x_B + y_B$.
- **Multiplication**. There are two scenarios to consider:
  - **Scalar**. For Alice and Bob to multiply a secret by a scalar x ($z = a \cdot x$), Alice computes $z_A = a \cdot x_A$ and Bob computes $z_B = a \cdot x_B$.

– **Shares**. Observe that, for Alice and Bob to multiply secrets x and y ($z = x \cdot y$), they require some sort of communication to compute cross terms:

$$\begin{aligned} x \cdot y &= (x_A + x_B) \cdot (y_A + y_B) && (2.2) \\ &= x_A \cdot y_A + x_A \cdot y_B + x_B \cdot y_A + x_B \cdot y_B && (2.3) \end{aligned}$$

At this point, Alice and Bob can execute two OLE s to secret share the cross terms $x_A \cdot y_B$ and $x_B \cdot y_A$. Indeed, if Alice inputs $(x_A, -s_A)$ and $(y_A, -s'_A)$ for random values s_A, s'_A and Bob inputs y_B and x_B, Bob will output $s_B = x_A \cdot y_B - s_A$ and $s'_B = y_A \cdot x_B - s'_A$. Thus, we have that $s_A + s_B = x_A \cdot y_B$ and $s'_A + s'_B = y_A \cdot x_B$. So, Alice share is $z_A = x_A \cdot y_A + s_A + s'_A$ and Bob share is $z_B = s_B + s'_B + x_B \cdot y_B$.

- **Output**. For Alice to receive the output value x of some output wire, Bob simply sends x_B to Alice. Alice outputs $x = x_A + x_B$.

2.3 Quantum information

Quantum information theory studies the implications of using quantum systems as the medium of information. Consequently, the information carriers are governed by the laws of quantum mechanics, allowing to exploit properties not present amid classical methods. In this section, we present the basic elements of quantum information that will be used by the presented quantum protocols and the corresponding security proofs.

As common practice in quantum information theory, a quantum system A is described by some Hilbert space $\mathcal{H}_A$. In this thesis, we will only consider finite-dimensional Hilbert spaces, i.e. $\dim \mathcal{H}_A = d < \inf$. So, we can identify $\mathcal{H}_A \cong \mathbb{C}^d \cong \mathcal{H}_A^*$, where $\mathcal{H}_A^*$ is the corresponding dual space. We use the Dirac bra-ket notation to describe the states in a physical quantum system A. So, a quantum (pure) state is described by a normalized vector $|\psi\rangle_A \in \mathcal{H}_A$ and its dual vector as $\langle\psi|_A \in \mathcal{H}_A^*$. In case it is clear from context, we may omit to which Hilbert space the state belongs to. Since we have the isomorphism $\mathcal{H}_A \cong \mathbb{C}^d$, we can identify the standard basis of $\mathbb{C}^d$ to the standard basis (also known as computational basis) of $\mathcal{H}$, i.e. $\{|i\rangle\}_{i=0}^{d-1}$. Also, given different subsystems $\mathcal{H}_1, \ldots, \mathcal{H}_n$, the joint system can be described by their tensor product, i.e. $\mathcal{H}_1 \otimes \ldots \otimes \mathcal{H}_n$. The vectors in this joint system are described as $|\boldsymbol{x}\rangle = |x_1\rangle \otimes \ldots \otimes |x_n\rangle$, where $\boldsymbol{x} \in \mathbb{Z}_d^n$.

We can generate quantum pure states $|\psi_i\rangle \in \mathcal{H}$ according to some probability distribution $\{p_i\}$. This situation is described by a density operator given by $\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|$, which is commonly called a mixed state. The density operators are positive semi-definite Hermitian operators with unitary trace, i.e. $\rho \geq 0$ and $\operatorname{tr} \rho = 1$. We denote by $\operatorname{Herm}(\mathcal{H})$,

$\text{Pos}(\mathcal{H})$ and $\mathcal{P}(\mathcal{H})$ the sets of Hermitian, positive-semi definite and density operators on $\mathcal{H}$, respectively.

A mixed state is classical if it is of the form $\rho_{\mathcal{X}} = \sum_{x\in\mathcal{X}} P_X(x)\,|x\rangle\langle x|$, where $\mathcal{X}$ is some finite set and P_X is a probability distribution over $\mathcal{X}$. In particular, we denote the uniform distribution over $\mathcal{X}$ as $\tau_{\mathcal{X}} = \frac{1}{|\mathcal{X}|}\sum_{x\in\mathcal{X}} |x\rangle\langle x|$, where $|\mathcal{X}|$ is the size of $\mathcal{X}$. The identity operator is denoted by $\mathbb{1}$. Also, for a bipartite quantum state ρ_{XB}, we say that ρ_{XB} is a classical-quantum state (cq-state for short) if it is of the form $\rho_{XB} = \sum_{x\in\mathcal{X}} P_X(x)\,|x\rangle\langle x| \otimes \rho_B^x$, where P_X is a probability distribution over the finite set $\mathcal{X}$.

2.3.1 Trace distance

To prove the security of quantum protocols, it is fundamental to have a procedure that allows to distinguish quantum states. Luckly, there is a useful metric that expresses how well two quantum states $\sigma, \rho \in \mathcal{P}(\mathcal{H})$ can be distinguished by any (potentially inefficient) procedure. This metric is commonly called *trace distance* and is defined as [70]

$$\delta(\rho,\sigma) := \frac{1}{2}||\rho-\sigma||_1,$$

where $||\cdot||_1$ is the $1-$Schatten norm in the space of bounded operators acting on a Hilbert space. Its name comes from the fact that we can write it using the trace operator as follows

$$||\rho-\sigma||_1 = \text{Tr}\left\{\sqrt{(\rho-\sigma)^\dagger(\rho-\sigma)}\right\}.$$

In this work, we will be working with completely positive trace preserving (CPTP) maps. By definition, a CPTP map preserves the normalization of the input states (hence being trace preserving) and it maps positive operators to positive operators (hence being completely positive). These two properties ensure that density operators are mapped to density operators. In summary, these operators describe all possible physically operations and will be extensively used in Chapter 5 (quantum oblivious linear evaluation protocol).

It can be shown that CPTP maps cannot be used to improve the distinguishability between quantum states. In other words, CPTP maps do not increase the trace distance. This is summarised in Lemma 1.

Lemma 1 (Lemma 7, [70])**.** *The trace distance has the following properties:*

1. *For any CPTP map $\mathcal{E}$ and any $\sigma, \rho \in \mathcal{P}(\mathcal{H})$ we have that*

$$\delta(\mathcal{E}(\sigma),\mathcal{E}(\rho)) \leq \delta(\sigma,\rho)$$

2. Let $\sigma, \sigma' \in \mathcal{P}(\mathcal{H})$ and $\rho \in \mathcal{P}(\mathcal{H}')$. Then,

$$\delta(\sigma \otimes \rho, \sigma' \otimes \rho) = \delta(\sigma, \sigma')$$

Although the following lemma is not directly related to the trace distance, it will be used in Chapter 5 to bound the trace distance between two states.

Lemma 2 (Corollary 4, [**?**]). *Let $X := \{x_1, \cdot, x_n\}$ be a list of (not necessarily distinct) values in $[0,1]$ with the average $\mu_X := \frac{1}{n}\sum_{i=1^n} x_i$. Let $\mathcal{T}$ of size t be a random subset of X with the average $\mu_T := \frac{1}{t}\sum_{i \in T} x_i$. Then, for any $\epsilon > 0$, the set $\bar{\mathcal{T}} = X \setminus T$ with average $\mu_{\bar{\mathcal{T}}} = \frac{1}{n-t}\sum_{i \in K} x_i$ satisfies*

$$Pr\left[\mu_{\bar{\mathcal{T}}} - \mu_T \geq \sqrt{\frac{n(t+1)}{2(n-t)t^2}\log\frac{1}{\epsilon}}\right] \leq \epsilon.$$

2.3.2 Entropy

How unpredictable is some random variable? How much information do we gain when we observe some system? To answer these questions, several measures under the name of entropy were developed. The most simple classical entropy measure was developed by Shannon [71]. Shannon found a way to express the amount of information a random variable has. Intuitivelly, this should express the following behaviour: when some event is more predictable, it conveys less information. Indeed, the more scandalous and unpreditable the news are, the more informative they are. Shannon started by proposing the following function to describe the amount of information an event A has:

$$I(A) = -\log\left(P(A)\right),$$

where $P(A)$ is the probability of the event A. From this, he defined the entropy to be the average information value of all the possible events. More formally, for a discrete random variable X

$$H(X) = \mathbb{E}\left[I(X)\right].$$

In the special case where we consider a distribution $P(X)$ over $\{0,1\}$ that selects 1 with probability p and 0 with probability $1-p$, we have the binary entropy given as

$$H(X) = -p\log_2 p - (1-p)\log_2(1-p).$$

Throughout our analysis, we will also use several times the so-called $d-$ary entropy function, a generalization of the standard binary entropy function. However, this entropy measure does not have such an operational meaning such as the binary entropy.

Definition 1. *For $d \geq 2$, the d-ary entropy function $h_d : [0,1] \to \mathbb{R}$ is given by*

$$h_d(x) = x \log_d(d-1) - x \log_d x - (1-x)\log_d(1-x).$$

The $d-$ary entropy is specially useful to bound the size of an important object in coding theory, the Hamming ball. The Hamming ball with radius μ centered at some point $\boldsymbol{r}$ is defined to be the set of vectors $\boldsymbol{z}$ at a distance μ from $\boldsymbol{r}$, when the distance is given by the Hamming distance d_H. So, we have the following Lemma.

Lemma 3 (Lemma 5, [72])**.** *For an integer $d \geq 2$ and $\mu \in [0, 1-\frac{1}{d}]$,*

$$|\{\boldsymbol{z} \in \mathbb{Z}_d^n : d_H(\boldsymbol{z}, \boldsymbol{r}) \leq \mu n\}| \leq d^{h_d(\mu)n}.$$

To prove the security of protocols is more important to understand the worst-case scenario instead of the average behaviour. Therefore, standard binary entropy style definitions are not enough for securing protocols and we need a new one. This is achieved with min-entropy. Classically, for a finite random variable X, where $P(X = x) = p_x$, the min-entropy is given by the following expression:

$$H_{\min}(X) = -\log \max p_x.$$

Operationally, this gives the probability of guessing the element drawn from X when choosing the element x with maximum probability,, i.e. $P_{\text{guess}} = \max p_x$. We can extend this definition to cq-states $\rho_{XB} \in \mathcal{P}(\mathcal{H}_A \otimes \mathcal{H}_B)$ as given by Definition 2.

Definition 2. *Let $\rho_{XB} \in \mathcal{P}(\mathcal{H}_X \otimes \mathcal{H}_B)$ be a cq-state. The conditional min-entropy is given by*

$$H_{\min}(X|B)_\rho = -\log P_{guess}(X|B),$$

where $P_{guess}(X|B)$ is given by

$$P_{guess}(X|B) = \max_{\{M_x\}_x} \sum_x p_x \operatorname{tr}\left[M_x \rho_x^B\right],$$

where the maximization is taken over all positive operator-valued measures (POVM), i.e. $\{M_x \geq 0 : \sum_x M_x = \mathbb{1}\}$.

In Definition 2, $P_{\text{guess}}(X|B)$ has the operational meaning of being the probability of guessing x having access to system B. Also, the maximization is taken over the most general kind of measurements allowed in quantum mechanics. Next, we present Lemma 4 that states how min-entropy changes when applying a fixed bijective function to the classical subsystem of a cq-state. This Lemma will be important to prove the security of the quantum oblivious linear evaluation protocol presented in Chapter 5.

Lemma 4. *Let $\rho_{XB} \in \mathcal{P}(\mathcal{H}_X \otimes \mathcal{H}_B)$ be a cq-state and let $f : \mathcal{X} \to \mathcal{X}$ be a fixed bijective function. Then,*

$$H_{\min}(X|B)_\rho \leq H_{\min}(f(X)|B)_\rho.$$

Proof. Consider the unitary operator,

$$U = \sum_x |f(x)\rangle\langle x| \, .$$

We check that U is indeed unitary:

$$UU^\dagger = \left(\sum_x |f(x)\rangle\langle x| \right) \left(\sum_{x'} |x'\rangle\langle f(x')| \right) = \sum_x |f(x)\rangle\langle f(x)| = I,$$

where in the last step we used the fact that the function f is a bijection. The same holds for $U^\dagger U = I$.

Now, observe the following,

$$\begin{aligned} H_{\min}(f(X)|B) &= -\log \max_{\{M_x\}_x} \sum_x p_x \operatorname{tr}\left[M_x \rho^B_{f(x)}\right] \\ &= -\log \max_{\{M_x\}_x} \sum_x p_x \operatorname{tr}\left[M_x U \rho^B_x U^\dagger\right] \\ &= -\log \max_{\{M_x\}_x} \sum_x p_x \operatorname{tr}\left[U^\dagger M_x U \rho^B_x\right] . \end{aligned}$$

Note that $\{N_x\}_x = \left\{U^\dagger M_x U\right\}_x$ is also a POVM: they are all positive semidefinite operators and it sums up to unity. Therefore, we have that $\left\{U^\dagger M_x U\right\}_x$ can only decrease the space of possible POVMs. So,

$$\max_{\{M_x\}_x} \sum_x p_x \operatorname{tr}\left[U^\dagger M_x U \rho^B_x\right] \leq \max_{\{M_x\}_x} \sum_x p_x \operatorname{tr}\left[M_x \rho^B_x\right] .$$

This means that,

$$H_{\min}(f(X)|B) \geq -\log \max_{\{M_x\}_x} \sum_x p_x \operatorname{tr}\left[M_x \rho_x^B\right] = H_{\min}(X|B).$$

□

The conditional min-entropy can be generalized to the fully quantum case where both systems are quantum (Definition 3).

Definition 3. *Let $\rho_{AB'} \in \mathcal{P}(\mathcal{H}_A \otimes \mathcal{H}'_B)$ and $\sigma_{B'} \in \mathcal{P}(\mathcal{H}'_B)$. The min-entropy of $\rho_{AB'}$ relative to $\sigma_{B'}$ is given by*

$$H_{min}(A|B')_{\rho|\sigma} = -\log \min\{\lambda : \lambda \cdot id_A \otimes \sigma_{B'} \geq \rho_{AB'}\}$$

and

$$H_{min}(A|B')_{\rho} = \sup_{\sigma_{B'}} H_{min}(A|B')_{\rho|\sigma}$$

Now, consider again the superposition state $|\phi\rangle_{AB'} = \sum_{z \in B} \alpha_z |z\rangle |\psi^z\rangle$ for some set $\mathcal{B}$ and arbitrary coefficients α_z. We define $\rho_{AB'} = |\phi\rangle\langle\phi|_{AB'}$ and the mixture $\tilde{\rho}_{AB'} = \sum_{z \in \mathcal{B}} |\alpha_z|^2 |z\rangle\langle z| \otimes |\psi^z\rangle\langle\psi^z|$. The following lemma gives lower bound on the min-entropy of $\rho_{AB'}$ in terms of the min-entropy of $\tilde{\rho}_{AB'}$.

Lemma 5 (Lemma 3.1.13, [74])**.** *Let $\rho_{AB'}$ and $\tilde{\rho}_{AB'}$ be defined as above. Then,*

$$H_{min}(A|B')_{\rho} \geq H_{min}(A|B')_{\tilde{\rho}} - \log|\mathcal{B}|$$

It will be crucial in the security proof of the quantum oblivious linear evaluation protocol presented in Chapter 5 to understand how min-entropy changes when a completely positive (CP) map is applied. It is known that, for a unital CP map $\mathcal{M}$ (i.e. $\mathcal{M}(\mathbb{1}) = \mathbb{1}$), the conditional min-entropy does not decrease, i.e. $H_{\min}(\mathcal{M}(A)|B) \geq H_{\min}(A|B)$. However, this result is not enough to achieve useful min-entropy bounds. To achieve a meaningful bound for certain operators $\mathcal{M}$, we can use Theorem 20 along with some other two results (Lemma 7 and Lemma 8). Note that, for readability purposes, the theorem uses the notation presented in Chapter 5.

Lemma 6 (Theorem 1, [73])**.** *Let $\mathbf{X}$ denote a system with n qudits, and $\mathcal{M}_{\mathbf{X}\to\mathbf{FY}}$ be a CP map such that $((\mathcal{M}^\dagger \circ \mathcal{M})_{\mathbf{X}} \otimes id_{\bar{\mathbf{X}}})(\Phi_{\mathbf{X}\bar{\mathbf{X}}}) = \sum_{(\boldsymbol{a},\boldsymbol{b}) \in \mathbb{Z}_d^{2n}} \lambda_{(\boldsymbol{a},\boldsymbol{b})} \Phi_{(\boldsymbol{a},\boldsymbol{b})}$. Then, for any partition of $\mathbb{Z}_d^{2n} = \mathfrak{S}_+ \cup \mathfrak{S}_-$ into subsets $\mathfrak{S}_+$ and $\mathfrak{S}_-$, and $\mathcal{M}(\sigma_{\mathbf{X}E}) = \sigma_{\mathbf{FY}E}$ we have*

$$2^{-H_2(\mathbf{FY}|E)_{\sigma_{\mathbf{FY}E}|\sigma_{\mathbf{X}E}}} \leq \sum_{(\boldsymbol{a},\boldsymbol{b})\in\mathfrak{S}_+} \lambda_{(\boldsymbol{a},\boldsymbol{b})} 2^{-H_2(\mathbf{X}|E)_{\sigma_{\mathbf{X}E}}} + (\max_{(\boldsymbol{a},\boldsymbol{b})\in\mathfrak{S}_-} \lambda_{(\boldsymbol{a},\boldsymbol{b})}) d^n, \tag{2.4}$$

where, in general, for a (not necessarily normalized) quantum state $\rho_{AB} \in \mathcal{P}(\mathcal{H}_A \otimes \mathcal{H}_B)$, $H_2(A|B)$ is the so-called collision entropy [74], given as

$$H_2(A|B)_{\rho_{AB}} = -\log\left(\mathrm{Tr}\left\{\left(\rho_B^{-1/4}\rho_{AB}\rho_B^{-1/4}\right)^2\right\}\right).$$

If we further condition on a general quantum state $\sigma_B \in \mathcal{P}(\mathcal{H}_B)$, we have

$$H_2(A|B)_{\rho_{AB}|\sigma_B} = -\log\left(\mathrm{Tr}\left\{\left(\sigma_B^{-1/4}\rho_{AB}\sigma_B^{-1/4}\right)^2\right\}\right).$$

It is interesting to note that when $\mathcal{M}$ is trace preserving, we have,

$$2^{-\mathrm{H}_2(\mathbf{FY}|E)_{\sigma_{\mathbf{FY}E}|\sigma_{\mathbf{X}E}}} = 2^{-\mathrm{H}_2(\mathbf{FY}|E)_{\sigma_{\mathbf{FY}E}}}.$$

This follows from the definition of the collision entropy and the fact that $\mathrm{Tr}_{\boldsymbol{FY}}\left[\mathcal{M}(\sigma_{\boldsymbol{X}E})\right] = \sigma_E$ [73].

Now, we need a way to relate min-entropy and collision entropy to have useful bounds for min-entropy. This is done through the following two Lemmas.

Lemma 7 (Lemma 17, [73])**.** *Let $\rho_{AB'} \in \mathcal{P}(\mathcal{H}_A \otimes \mathcal{H}_{B'})$ and $d_A = \dim \mathcal{H}_A$. Then*

$$H_{\min}(A|B')_\rho \leq H_2(A|B')_\rho \leq 2H_{\min}(A|B')_\rho + \log d_A.$$

Lemma 8 (Lemma 18, [73])**.** *Let $\rho_{XB'} \in \mathcal{P}(\mathcal{H}_X \otimes \mathcal{H}_{B'})$ be a cq-state. Then*

$$H_{\min}(X|B')_\rho \leq H_2(X|B')_\rho \leq 2H_{\min}(X|B')_\rho.$$

2.3.3 Two-universal functions

We start by defining a particular set of functions that are usually used to amplify the privacy of the parties' input and output elements.

Definition 4 ($\delta-$almost two-universal hash family; two-universal hash family)**.** *A family, $\mathfrak{G}$, of functions, g, with domain D and range R is called a $\delta-$almost two-universal hash family if for any two distinct elements $w, w' \in D$ and for g chosen at random from $\mathfrak{G}$, the probability of a collision $g(w) = g(w')$ is at most δ. In the special case that $\delta = 1/|R|$, where $|R|$ is the size of the range R, the family is called two-universal.*

Now, we present a particular two-universal hash family that preserves the structure of the OLE input and output while keeping its privacy amplification guarantees. This family is called Multi-linear Modular Hashing as it is based on the modular inner product of vectors [75].

Definition 5 (Definition 2, [75]). *Let d be a prime and let n be an integer $n > 0$. Define a family MMH* (Multi-linear Modular Hashing) of functions from $\mathbb{Z}_d^n$ to $\mathbb{Z}_d$ as follows*

$$MMH^* := \{g_x : \mathbb{Z}_d^n \to \mathbb{Z}_d \,|\, x \in \mathbb{Z}_d^n\}$$

where the functions g_x are defined for any $x = (x_1, \ldots, x_n)$, $m = (m_1, \ldots, m_n) \in \mathbb{Z}_d^n$

$$g_x(m) = x \cdot m \mod d = \sum x_i\, m_i \mod d$$

Theorem 1 (Theorem 3, [75]). *The family MMH* is two-universal.*

In fact, Halevi and Krawczyk [75] prove a stronger result, namely that the MMH* family is $\Delta-$*universal*, which is more general than two-universal. For the sake of simplicity, we just present here this particular version of the theorem.

The following *Generalized Leftover Hash Lemma* is a relevant ingredient in the security proof of Chapter 5, as it ensures that, after applying a known function g from a two-universal family to a random variable X, the resulting random variable $Z = g(X)$ is close to uniform, conditioned on some (possibly quantum) side information E. This is a high-dimensional version of the Lemma, that can be easily deduced by using Lemma 4 from [76] with $d_A = d^l$. Note also that this is a special version, since Tomamichel et al. in [76] prove it in the more general case for $\delta-$almost two-universal hash families.

Lemma 9 (Generalized Leftover Hash Lemma [76]). *Let X be a random variable, E a quantum system, and $\mathfrak{G}$ a two-universal family of hash functions from X to $\mathbb{Z}_d^l$. Then, on average over the choices of g from $\mathfrak{G}$, the output $Z := g(X)$ is ξ-close to uniform conditioned on E, where*

$$\xi = \frac{1}{2}\sqrt{2^{l \log d - H_{min}(X|E)}}. \tag{2.5}$$

2.4 Universal Composability

The Universal Composability (UC) framework was introduced by Canetti [77] in the classical setting and extended to the quantum setting by Unruh, and Ben-Or and Mayers

[78, 79](see also [80, 81]). It renders strong composability guarantees as it ensures that the security of the protocol is independent of any external execution of the same or other protocols. Both classical and quantum frameworks follow the same ideal-real world comparison structure and consider the same interactions between machines. The difference lies on the operations allowed by the quantum-UC framework, where we can also store, transmit, and process quantum states.

More specifically, the quantum-UC security of some protocol π is drawn by comparing two scenarios. A real scenario where π is executed and an ideal scenario where an ideal functionality, $\mathcal{F}$, that carries out the same task and is defined *a priori*, is executed. The comparison is done by a special machine, called the *environment*, $\mathcal{Z}$, that supervises the execution of both scenarios and has access to any external information (e.g. concurrent executions of the same or any other protocol). In the two-party case, the structure of the machines in both scenarios is as follows: The real scenario has the environment, $\mathcal{Z}$, the adversary, Adv, and the two parties Alice and Bob, while the ideal scenario has the environment $\mathcal{Z}$, the simulator $\mathcal{S}$, the two parties Alice and Bob and the ideal functionality $\mathcal{F}$. Informally, we say that the protocol π is quantum-UC secure if no environment $\mathcal{Z}$ can distinguish between the execution of π in the real scenario and the execution of the functionality $\mathcal{F}$ in the ideal scenario. Any possible attack of the adversary Adv in the execution of π can be simulated by the simulator $\mathcal{S}$ in the ideal-world execution of $\mathcal{F}$, without any noticeable difference form the point-of-view of the environment $\mathcal{Z}$. Since the ideal functionality $\mathcal{F}$ is secure by definition, the real-world adversary is not able to extract any more information than what is allowed by the functionality $\mathcal{F}$.

Let us now see the formal definition. Let π and ρ be the real and ideal two-party protocols. We denote by $\text{EXEC}_{\pi^C,Adv,\mathcal{Z}}$ (analogously $\text{EXEC}_{\rho^C,\mathcal{S},\mathcal{Z}}$) the output of the environment $\mathcal{Z}$ at the end of the real (ideal) execution, and by C the corrupted party.

Definition 6 (Statistical quantum-UC security, Computational quantum-UC security [80])**.** *Let protocols π and ρ be given. We say that π statistically quantum-UC emulates ρ if and only if for every party, C, and for every adversary, Adv, there exists a simulator, $\mathcal{S}$, such that for every environment $\mathcal{Z}$, and every $z \in \{0,1\}^*$, $n \in \mathbb{N}$*

$$\left|Pr[EXEC_{\pi^C,Adv,\mathcal{Z}}(n,z)=1] - Pr[EXEC_{\rho^C,\mathcal{S},\mathcal{Z}}(n,z)=1]\right| \leq \mu(n),$$

where $\mu(n)$ is a negligible function and n is the security parameter. We furthermore require that if Adv is quantum-polynomial-time, so is $\mathcal{S}$. Finally, if we consider quantum-polynomial-time Adv and $\mathcal{Z}$ we have computational quantum-UC security.

The role of the simulator, $\mathcal{S}$, is to simulate the execution of the protocol π in such a way that the environment $\mathcal{Z}$ is not able to distinguish between the two executions. In particular, since $\mathcal{S}$ does not have access to the inputs/outputs of the actual honest party, it runs a simulated honest party interacting with the environment which is acting as the adversary. Moreover, $\mathcal{S}$ controls the dishonest party, therefore controlling their inputs to the ideal functionality $\mathcal{F}$. It then relies on its ability to extract the inputs that the environment provides to the dishonest party. Then, it uses them along with the ideal functionality outputs in order to successfully generate a simulated execution that cannot be distinguished by the environment.

Finally, note that, in case the real execution of the protocol π makes use of some external functionality $\mathcal{F}_{\mathbf{ext}}$, the simulator $\mathcal{S}$ can control and reprogram $\mathcal{F}_{\mathbf{ext}}$ in whatever way suits best in the ideal world to produce an indistinguishable simulation of the real world.

Ideal functionalities

Whenever a protocol π makes use of an external functionality $\mathcal{F}_{\mathbf{ext}}$, we say that π is in the $\mathcal{F}_{\mathbf{ext}}-$hybrid model. The quantum OLE protocol $\pi_{\mathbf{QOLE}}$ presented in Chapter 5 uses the ideal commitment functionality, $\mathcal{F}_{\mathbf{COM}}$, defined in Figure 2.5. Note that the protocol makes several calls to $\mathcal{F}_{\mathbf{COM}}$ and only opens a subset of the committed elements. Therefore, we use an index element i to specify different instance calls. In the commitment phase, Bob sends $(\texttt{commit}, i, M)$ to the functionality and the functionality sends $(\texttt{commit}, i)$ to Alice. In the opening phase, Bob sends $(\texttt{open}, i)$ and the functionality sends $(\texttt{open}, i, M)$ to Alice.

The $\mathcal{F}_{\mathbf{COM}}$ functionality can be replaced by the commitment protocol $\pi_{\mathbf{COM}}$ presented in [**?**] which is computationally UC-secure in the Common Reference String (CRS) model. As analyzed in [82] (Theorem 3.), the protocol $\pi_{\mathbf{COM}}$ computationally quantum-UC realizes $\mathcal{F}_{\mathbf{COM}}$ in the CRS model. As a result, since $\pi_{\mathbf{QOLE}}$ is proved to be quantum-UC secure, the resulting protocol $\pi_{\mathbf{QOLE}}^{\pi_{\mathbf{COM}}}$ is quantum-UC secure by the composition theorem [80].

$\mathcal{F}_{\mathbf{COM}}$ functionality

- **Commitment phase**. Upon receiving $(\texttt{commit}, M)$ from Bob, the functionality sends $\texttt{commit}$ to Alice.
- **Opening phase**. Upon receiving $\texttt{open}$ from Bob, the functionality sends $(\texttt{open}, M)$ to Alice.

Figure 2.5: Commitment functionality.

Chapter 3

Quantum oblivious transfer

In a recent survey on classical oblivious transfer (OT) [83], all the analysed protocols require some form of asymmetric cryptography. Indeed, in the classical setting, it is impossible to develop information-theoretic secure OT or even reduce it to one-way functions, requiring some public-key computational assumptions. As shown by Impaggliazzo and Rudich [57], one-way functions (symmetric cryptography) alone do not imply key agreement (asymmetric cryptography). Also, Gertner et al. [84] pointed out that since it is known that OT implies key agreement, this sets a separation between symmetric cryptography and OT, leading to the conclusion that OT cannot be generated alone by symmetric cryptography. Otherwise, one could use one-way functions to implement key agreement through the OT construction. This poses a threat to all classical OT protocols [85–87] that are based on mathematical assumptions provably broken by a quantum computer [21]. Besides the security problem, asymmetric cryptography tends to be computationally more complex than symmetric cryptography, creating a problem in terms of speed when a large number of OTs are required. The classical post-quantum approach, thrives to find protocols resistant against quantum computer attacks. However, these are still based on complexity problems and are not necessarily less computationally expensive, than the previously mentioned ones.

In parallel to the classical post-quantum approach, the quantum cryptography community tackled this security issue by presenting some OT protocols based on quantum technologies. Intriguingly enough, more than a decade before the first classical OT by Rabin (1981, [48]) was published, Wiesner proposed a similar concept. However, at the time, it was rejected for publication due to the lack of acceptance in the research community. The first published quantum OT (QOT) protocol, known as the BBCS (Bennett-Brassard-Crépeau-Skubiszewska) protocol [30] was only presented in 1992. Remarkably, there is a distinctive difference between classical and quantum OT from a security standpoint, as the latter is proved to be possible assuming only the existence of quantum-hard one-way functions [31, 32]. This means quantum OT requires weaker security assumptions than classical OT.

In this chapter, we review the particular topic of quantum OT. We mainly comment on several important OT protocols, their underlying security models and assumptions. To the best of our knowledge, there is no prior survey dedicated to quantum OT protocols alone. Usually, its analysis is integrated into more general surveys under the topic of "quantum cryptography", leading to a less in-depth exposition of the topic. For reference, we provide some distinctive reviews on the general topic of quantum cryptography [22, 88–94].

This chapter is divided as follows. We start by giving a brief overview of the impossibility results related to quantum OT. Then, we provide an exposition about some of the most well-known quantum OT protocols based on assumptions. Finally, we give a brief overview of OT protocols not covered throughout this thesis.

3.1 Impossibility results

The beginning of the development of quantum OT (QOT) came hand in hand with the development of quantum bit commitment (QBC). In fact, the first proposed QOT protocol (BBCS [30]) reduces QOT to QBC . This sets a distinctive difference between classical and quantum protocols. Although bit commitment (BC) can be reduced to oblivious transfer (OT) [18], the reverse is not true using only classical communication [95]. Therefore, Yao's proof [96] of BBCS protocol [30] gives quantum communications the enhanced quality of having an equivalence between QOT and QBC - they can be reduced to each other - a relation that is not known in the classical realm.

At the time of the BBCS protocol, the quest for unconditionally secure QOT was based on the possibility of unconditional secure QBC. A year later, Brassard et al. presented a QBC protocol [97] named after the authors, BCJL (Brassard-Crépeau-Jozsa-Langlois). However, this work presented a flawed proof of its unconditional security which was generally accepted for some time, until Mayers spotted an issue on it [98]. Just one year after, Lo and Chau [99], and Mayers [100] independently proved unconditional QBC to be impossible. Nevertheless, the existence of unconditionally secure QOT not based on QBC was still put as an open question [88] even after the so-called no-go theorems [99, 100]. However, Lo was able to prove directly that unconditionally secure QOT is also impossible [101]. He concluded this as a corollary of a more general result that states that secure two-party computations which allow only one of the parties to learn the result (one-side secure two-party computation) cannot be unconditionally secure. Lo's results triggered a line of research on the possibility of two-sided secure two-party computation (both parties are allowed to learn the result without having access to the other party's inputs), which was also proved by Colbeck to be impossible [102] and extended in subsequent works [103–105]. For a more in-depth review of the impossibility results presented by Lo, Chau and Mayers, we refer the interested reader to the following works [95, 106].

Although the impossibility results have been well accepted in the quantum cryptography community, there was some criticism regarding the generality of the results [107–110]. This line of research reflects the view put forward by Yuen [107] in the first of these papers: "Since there is no known characterization of all possible QBC protocols, logically there can really be no general impossibility proof, strong or not, even if it were indeed impossible to have an unconditionally secure QBC protocol." In parallel, subsequent analyses were carried out, reaffirming the general belief of impossibility [111–113]. However, most of the discord has ended with Ariano et al. proof [114] in 2007, giving an impossibility proof covering all conceivable protocols based on classical and quantum information theory. Subsequent work digested Ariano et al. [114] work, trying to present more succinct proofs [115–117] and to translate it into categorical quantum mechanics language [118–120].

Facing these impossibility results, the quantum cryptography community followed two main paths:

1. Develop OT protocols under some assumptions. These could be based on limiting the technological power of the adversary (e.g. noisy-storage model, relativistic protocols, isolated-qubit model) or assuming the security of additional functionalities (e.g. bit commitment).

2. Develop OT protocols with a relaxed security definition. These allow the adversary to extract, with a given probability, some information (partial or total) about the honest party input/output. This approach leads to the concepts of weak OT and weak private database query.

In the next section, we explore protocols that produce a special primitive called *oblivious keys* as an intermediate step.

3.2 BBCS-based protocols

In this section, we explore protocols that circumvent the no-go theorems [99, 100] through assumptions. Some of the presented solutions are based on one-way functions, which are believed to be quantum-hard [31, 32, 121], and others rely on technological or physical limitations of the adversaries [122–127]. The latter are qualitatively different from complexity-based assumptions on which post-quantum protocols rely. Also, all these assumptions have the important property that they only have to hold during the execution of the protocol for its security to be preserved. In other words, even if the assumptions lose their validity at some later point in time, the security of the protocol is not compromised. This property is commonly known as *everlasting* security [128]. Everlasting security is also a major distinctive feature of quantum protocols when compared with classical cryptographic approaches.

We start by presenting the first QOT protocol. Then, we see how this protocol led to the development of two assumption models: $\mathcal{F}_{\text{COM}}-$hybrid model and the limited-quantum-storage model.

3.2.1 BBCS protocol

Notation conflict: $\mathcal{F}$ to denote universal functions

In 1983, Wiesner came up with the idea of *quantum conjugate coding* [129]. This technique is the main building block of many important quantum cryptographic protocols [28, 130, 131], including quantum oblivious transfer [30]. It also goes under the name of *quantum multiplexing* [131], *quantum coding* [132] or *BB84 coding* [95]. In quantum conjugate coding we encode classical information in two conjugate (non-orthogonal) bases. This allows us to have the distinctive property that measuring on one basis destroys the encoded information on the corresponding conjugate basis. So, when bit 0 and 1 are encoded by these two bases, no measurement is able to perfectly distinguish the states. We will be using the following bases in the two-dimensional Hilbert space $\mathcal{H}_2$:

- Computational basis: $+ := \left\{ |0\rangle_+ , |1\rangle_+ \right\}$;
- Hadamard basis: $\times := \left\{ |0\rangle_\times , |1\rangle_\times \right\} = \left\{ \frac{1}{\sqrt{2}}\big(|0\rangle_+ + |1\rangle_+ \big), \frac{1}{\sqrt{2}}\big(|0\rangle_+ - |1\rangle_+ \big) \right\}$.

Throughout this chapter we abuse the notation and consider that the set of bases $\{+, \times\}$ can be associated with the binary set $\{0, 1\}$. $+$ is associated with 0 and $\times$ with 1. This is specially useful to compare strings of bases from different parties, i.e. the XOR operation ($\oplus$) between two vectors $\boldsymbol{\theta}^{\mathsf{A}}, \boldsymbol{\theta}^{\mathsf{B}} \in \{+, \times\}^n$ is defined as the XOR between the corresponding binary vectors $\boldsymbol{\theta}^{\mathsf{A}}, \boldsymbol{\theta}^{\mathsf{B}} \in \{0, 1\}^n$.

Protocol [30]. The first proposal of a quantum oblivious transfer protocol is presented in Figure 3.1 and it is called after its creators, Bennett-Brassard-Crépeau-Skubiszewska (BBCS). It builds on top of the quantum conjugate coding technique. Alice starts by using this encoding to generate a set of qubits that are subsequently randomly measured by Bob. These two steps make up the first phase of the BB84 QKD protocol. For this reason, this is called the *BB84 phase*. Next, both parties use the output bits obtained from Bob and the random elements generated by Alice to share a special type of key, known as *oblivious key*. This is achieved when Alice reveals her bases $\boldsymbol{\theta}^{\mathsf{A}}$ to Bob. Using the oblivious key as a resource, Alice can then obliviously send one of the messages m_0, m_1 to Bob, ensuring that he is only able to know one of the messages. This is achieved using a two-universal family of hash functions $\mathcal{F}$ from $\{0,1\}^{n/2}$ to $\{0,1\}^l$. Recall, we use the notation $s \leftarrow_\$ S$ to describe a situation where an element s is drawn uniformly at random from the set S.

$\Pi^{\mathbf{BBCS}}$ protocol

Parameters: n, security parameter; $\mathcal{F}$ two-universal family of hash functions.
Alice's input: $(m_0, m_1) \in \{0,1\}^l$ (two messages).
Bob's input: $b \in \{0,1\}$ (bit choice).

BB84 phase:

1. Alice generates random bits $\boldsymbol{x}^{\mathsf{A}} \leftarrow_\$ \{0,1\}^n$ and random bases $\boldsymbol{\theta}^{\mathsf{A}} \leftarrow_\$ \{+,\times\}^n$. Sends the state $\left|\boldsymbol{x}^{\mathsf{A}}\right\rangle_{\boldsymbol{\theta}^{\mathsf{A}}}$ to Bob.
2. Bob randomly chooses bases $\boldsymbol{\theta}^{\mathsf{B}} \leftarrow_\$ \{+,\times\}^n$ to measure the received qubits. We denote by $\boldsymbol{x}^{\mathsf{B}}$ his output bits.

Oblivious key phase:

3. Alice reveals to Bob the bases $\boldsymbol{\theta}^{\mathsf{A}}$ used during the *BB84 phase* and sets his oblivious key to $\mathsf{ok}^{\mathsf{A}} := \boldsymbol{x}^{\mathsf{A}}$.
4. Bob computes $\mathsf{e}^{\mathsf{B}} = \boldsymbol{\theta}^{\mathsf{B}} \oplus \boldsymbol{\theta}^{\mathsf{A}}$ and sets $\mathsf{ok}^{\mathsf{B}} := \boldsymbol{x}^{\mathsf{B}}$.

Transfer phase:

5. Bob defines $I_0 = \{i : \mathsf{e}^{\mathsf{B}}_i = 0\}$ and $I_1 = \{i : \mathsf{e}^{\mathsf{B}}_i = 1\}$ and sends the $(I_b, I_{b\oplus 1})$ to Alice.
6. Alice picks two uniformly random hash functions $f_0, f_1 \in \mathcal{F}$, computes the pair of strings (s_0, s_1) as $s_i = m_i \oplus f_i(\mathsf{ok}^{\mathsf{A}}_{I_{b\oplus i}})$ and sends the pairs (f_0, f_1) and (s_0, s_1) to Bob.
7. Bob computes $m_b = s_b \oplus f_i(\mathsf{ok}^{\mathsf{B}}_{I_0})$.

Alice's output: $\perp$.
Bob's output: m_b.

Figure 3.1: BBCS OT protocol.

Oblivious keys. As we saw in the BBCS protocol, oblivious keys can be used as a resource to produce OT instances. In fact, we can draw a comparison between standard encryption keys and oblivious keys. In the same way as standard keys are the resource that allows the encryption of a specific message, oblivious keys are the resource that enables the performance of OT with messages. In other words, encryption methods consume standard keys, while OT methods consume oblivious keys. The term, oblivious key, was used for the first time by Fehr and Schaffner [81] referring to random OT. However, under a subtle different concept, it was put forth by Jakobi et al. [133] and used to implement private database queries (PDQ). Also, in a recent work, Lemus et al. [134] presented the concept of oblivious key applied to OT protocols. We can define it as follows.

Definition 7 (Oblivious key)**.** *An oblivious key shared between two parties, Alice and Bob, is a tuple* $\mathsf{ok} := \big(\mathsf{ok}^{\mathsf{A}}, (\mathsf{ok}^{\mathsf{B}}, \mathsf{e}^{\mathsf{B}})\big)$ *where* ok^{A} *is Alice's key,* ok^{B} *is Bob's key and* e^{B} *is Bob's signal string.* e^{B} *indicates which indexes of* ok^{A} *and* ok^{B} *are correlated and which indexes are uncorrelated, i.e.* $\mathsf{e}^{\mathsf{B}}_i = 0$ *when the corresponding indexes are correlated and* $\mathsf{e}^{\mathsf{B}}_i = 1$ *when they are not.*

Note that, for some index i, when two index elements $\mathsf{ok}^{\mathsf{A}}_i$ and $\mathsf{ok}^{\mathsf{B}}_i$ are correlated, $\mathsf{ok}^{\mathsf{A}}_i = \mathsf{ok}^{\mathsf{B}}_i$. However, when they are uncorrelated, they are drawn independently. This means that both index elements may either be equal or different. Consider the following oblivious key $\mathsf{ok} = (001101101101, (000101001100, 101000110001))$ as an example. We can check it is a well strucured oblivious key:

$$\left.\begin{array}{rcccccccccccc} \mathsf{ok}^{\mathsf{A}}: & 0&0&1&1&0&1&1&0&1&1&0&1\\ \mathsf{ok}^{\mathsf{B}}: & 0&0&0&1&0&1&0&0&1&1&0&0\\ \mathsf{e}^{\mathsf{B}}: & 1&0&1&0&0&0&1&1&0&0&0&1 \end{array}\right\}\mathsf{ok}$$

It is worth stressing that oblivious keys are independent of the sender's messages m_0, m_1 and are not the same as random OT. In fact, as Alice does not know the groups of indexes I_0 and I_1 computed by Bob after the basis revelation, Alice does not have her messages fully defined. A similar concept was defined by König et al. [124] under the name of *weak string erasure.*

Security. Regarding security, the BBCS protocol is unconditionally secure against dishonest Alice. Intuitively, this comes from the fact that Alice does not receive any information from Bob other than some set of indexes I_0. However, the BBCS protocol is insecure against dishonest Bob. In its original paper [30], the authors describe a memory attack that provides Bob complete knowledge on both messages m_0 and m_1 without being detected. This can be achieved by having the receiver delay his measurements in step 2 to some moment after step 3. This procedure is commonly called the memory attack as it requires quantum memory to hold the states until step 3. The authors suggest that, for the protocol to be secure, the receiver has to be forced to measure the received states at step 2. In the following sections, we present two common approaches to tackle this issue. We may assume the existence of commitments or set physical assumptions that constrain Bob from delaying his measurement.

3.2.2 BBCS in the $\mathcal{F}_{\mathbf{COM}}-$hybrid model

As mentioned in the previous section, a secure BBCS protocol requires Bob to measure his qubits in step 2. In this section, we follow the suggestion from the original BBCS paper [30] and fix this loophole using a commitment scheme. Since we assume we have access to some commitment scheme, we call it $\mathcal{F}_{\mathbf{COM}}-$hybrid model[1].

$\Pi^{\mathbf{BBCS}}_{\mathcal{F}_{\mathbf{COM}}}$ protocol

Parameters: n, security parameter; $\mathcal{F}$ two-universal family of hash functions.
Alice's input: $(m_0, m_1) \in \{0,1\}^l$ (two messages).
Bob's input: $b \in \{0,1\}$ (bit choice).

BB84 phase: Same as in $\Pi^{\mathbf{BBCS}}$ (Figure 3.1).

Cut and choose phase:

3. Bob commits to the bases used and the measured bits, i.e. $\mathbf{COM}\big(\boldsymbol{\theta}^{\mathsf{B}}, \boldsymbol{x}^{\mathsf{B}}\big)$, and sends to Alice.
4. Alice asks Bob to open a subset T of commitments (e.g. $n/2$ elements) and receives $\{\theta_i^{\mathsf{B}}, x_i^{\mathsf{B}}\}_{i\in T}$.
5. In case any opening is not correct or $x_i^{\mathsf{B}} \neq x_i^{\mathsf{A}}$ for $\theta_i^{\mathsf{B}} = \theta_i^{\mathsf{A}}$, abort. Otherwise, proceed.

Oblivious key phase: Same as in $\Pi^{\mathbf{BBCS}}$ (Figure 3.1).

Transfer phase: Same as in $\Pi^{\mathbf{BBCS}}$ (Figure 3.1).

Alice's output: $\perp$.
Bob's output: m_b.

Figure 3.2: BBCS OT protocol in the $\mathcal{F}_{\mathbf{COM}}-$hybrid model.

[1]The notation $\mathcal{F}_{\mathbf{COM}}$ is commonly used for ideal functionalities. However, here we abuse the notation by using $\mathcal{F}_{\mathbf{COM}}$ to refer to any commitment scheme (including the ideal commitment functionality).

Protocol. The modified BBCS (Figure 3.2) adds a *cut and choose* phase that makes use of a commitment scheme **COM** to check whether Bob measured his qubits in step 2 or not. It goes as follows. Bob commits to the bases used to measure the qubits in the *BB84 phase* and the resulting output bits. Then,Alice chooses a subset of qubits to be tested and asks Bob to open the corresponding commitments of the bases and output elements. If no inconsistency is found, both parties can proceed with the protocol. Note that the size of the testing subset has to be proportional to n (security parameter), as this guarantees that the rest of the qubits were measured by Bob with overwhelming probability in n.

Security. Formally proving the security of this protocol led to a long line of research [29–32, 81, 96, 135–140]. Earlier proofs from the 90's started by analyzing the security of the protocol against limited adversaries that were only able to do individual measurements [136]. Then, Yao [96] was able to prove its security against more general adversaries capable of doing fully coherent measurements. Although these initial works [96, 136, 137] were important to start developing a QOT security proof, they were based on unsatisfactory security definitions. At the time of these initial works, there was no composability framework [81, 139] under which the security of the protocol could be considered. In modern quantum cryptography, these protocols are commonly proved in some quantum simulation-paradigm frameworks [29, 81, 124, 139]. In these paradigms, the security is proved by showing that an adversary in a real execution of the protocol cannot cheat more than what he is allowed in an ideal execution, which is secure by definition. This is commonly proved by utilizing an entity, called simulator, whose role is to guarantee that a real execution of the protocol is indistinguishable from an ideal execution. Moreover, they measured the adversary's information through average-case measures (e.g. Collision Entropy, Mutual Information) which are proven to be weak security measures when applied to cryptography [141, 142].

More desirable worst-case measures started to be applied to quantum oblivious transfer around a decade later [74, 143]. These were based on the concept of *min-entropy* [141, 142], $H_{\min}$, which, intuitively, reflects the maximum probability of an event to happen. More precisely, in order to prove security against dishonest Bob, one is interested in measuring Bob's min-entropy on Alice's oblivious key ok^{A} conditioned on some quantum side information E he may has, i.e. $H_{\min}(\mathsf{ok}^{\mathsf{A}}|E)$. Informally, for a bipartite classical-quantum state ρ_{XE} the conditional min-entropy $H_{\min}(X|E)$ is given by

$$H_{\min}(X|E)_{\rho_{XE}} := -\log P_{guess}(X|E),$$

where $P_{guess}(X|E)$ is the probability the adversary guesses the value x maximized over all possible measurements. Damgård et al. [29] were able to prove the stand-alone QOT security when equipped with this min-entropy measure and with the quantum simulation-paradigm framework developed by Fehr and Schaffner [81]. Their argument to prove the security of the protocol against dishonest Bob can be summarized as follows. The cut and choose phase ensures that Bob's conditional min-entropy on the elements of ok^{A} belonging to I_1 (indexes with uncorrelated elements between Alice's and Bob's oblivious keys) is lower-bounded by some value that is proportional to the security parameter, i.e. $H_{\min}(\mathsf{ok}^{\mathsf{A}}_{I_1}|E) \geq n\lambda$ for some $\lambda > 0$. Note that this is equivalent to derive an upper bound on the guessing probability $P_{guess}(\mathsf{ok}^{\mathsf{A}}_{I_1}|E) \leq 2^{-n\lambda}$. Having deduced an expression for λ, they proceed by applying a random hash function f from a two-universal family $\mathcal{F}$, $f \leftarrow_{\$} \mathcal{F}$. This final step ensures that $f(\mathsf{ok}^{\mathsf{A}}_{I_1})$ is statistically indistinguishable from uniform (privacy amplification theorem [143–145]). The proof provided by Damgård et al. [29] was extended by Unruh [139] to the quantum Universal Composable (UC) model, making use of ideal commitments. Now, a natural question arises:

Which commitment schemes can be used to render simulation-based security?

Commitment scheme. The work by Aaronson [121] presented a non-constructive proof that "indicates that collision-resistant hashing might still be possible in a quantum setting", giving confidence in the use of commitment schemes based on quantum-hard one-way functions in the $\Pi^{\mathbf{BBCS}}_{\mathcal{F}_{\mathbf{COM}}}$ protocol. Hopefully, it was shown that commitment schemes can be built from any one-way function [146–148], including quantum-hard one-way functions. Although it is intuitive to plug in into $\Pi^{\mathbf{BBCS}}_{\mathcal{F}_{\mathbf{COM}}}$ a commitment scheme derived from a quantum-hard one-way function, this does not necessarily render a simulation-based secure protocol. This happens because the nature of the commitment scheme can make the simulation-based proof difficult or even impossible. For a detailed discussion see [31].

Indeed, the commitment scheme must be quantum secure. Also, the simulator must have access to two intriguing properties: *extractability* and *equivocality*. Extractability means the simulator can extract the committed value from a malicious committer. Equivocal means the simulator can change the value of a committed value at a later time. Although it seems counter-intuitive to use a commitment scheme where we can violate both security properties (hiding and biding properties), it is fundamental to prove its security. Extractability is used by the simulator to prove security against the dishonest sender and equivocality is used by the simulator to prove security against the dishonest receiver. In the literature, there have been some proposals of the commitment schemes COM with these properties based on:

- Quantum-hard one-way functions [31, 32];

- Common Reference String (CRS) model [139, 149];
- Bounded-quantum-storage model [150];
- Quantum hardness of the Learning With Errors assumption [29].

Composability. The integration of secure OT executions in secure multiparty protocols [14] should not lead to security breaches. Although it seems intuitive to assume that a secure OT protocol can be integrated within more complex protocols, proving this is highly non-trivial as it is not clear *a priori* under which circumstances protocols can be composed [151].

The first step towards composability properties is the development of simulation based-security. However, this does not necessarily imply composability (see Section 4.2 of [151] for more details), as a composability framework is also required. In the literature, there have been some proposals for such a framework. In summary, Fehr and Schaffner [81] developed a composability framework that allows sequential composition of quantum protocols in a classical environment. The works developed by Ben-Or and Mayers [152] and Unruh [139, 153] extended the classical Universal Composability model [77] to a quantum setting (quantum-UC model), allowing concurrent composability. Maurer and Renner [154] developed a more general composability framework that does not depend on the models of computation, communication, and adversary behaviour. More recently, Broadbent and Karvonen [120] created an abstract model of composable security in terms of category theory. Up until now, and to the best of our knowledge, the composable security of the protocol $\Pi^{\mathbf{BBCS}}_{\mathcal{F}_{\mathbf{COM}}}$ was only proven in the Fehr and Schaffner model [81] by Damgård et al. [29] and in the quantum-UC by Unruh [139].

3.2.3 BBCS in the limited-quantum-storage model

In this section, we review protocols based on the limited-quantum-storage model. The protocols developed under this model avoid the no-go theorems because they rely their security on reasonable assumptions regarding the storage capabilities of both parties. Under this model, there are mainly two research lines. One was started by Damgård, Fehr, Salvail and Schaffner [122], who developed the bounded-storage model. In this model, the parties can only store a limited number of qubits. The other research line was initiated by Wehner, Schaffner and Terhal [123], who developed the noisy-storage model. In this model the parties can store *all* qubits. However, they are assumed to be unstable, i.e. they only have imperfect noisy storage of qubits that forces some decoherence. In both models, the adversaries are forced to use their quantum memories as both parties have to wait a predetermined time (Δt) during the protocol.

3.2.4 Bounded-quantum-storage model

In the bounded-quantum-storage model or BQS model for short, we assume that, during the waiting time Δt, the adversaries are only able to store a fraction $0 < \gamma < 1$ of the transmitted qubits, i.e. the adversary is only able to keep $q = n\gamma$ qubits. The parameter γ is commonly called the storage rate.

Protocol. The protocol in the BQS model, $\Pi^{\mathbf{BBCS}}_{\mathbf{bqs}}$, is very similar to the BBCS protocol $\Pi^{\mathbf{BBCS}}$ presented in Figure 3.1. The difference is that both parties have to wait a predetermined time (Δt) after step 2. This protocol is presented in Figure 3.3.

$\Pi^{\mathbf{BBCS}}_{\mathbf{bqs}}$ protocol

Parameters: n, security parameter; $\mathcal{F}$ two-universal family of hash functions.
Alice's input: $(m_0, m_1) \in \{0,1\}^l$ (two messages).
Bob's input: $b \in \{0,1\}$ (bit choice).

BB84 phase: Same as in $\Pi^{\mathbf{BBCS}}$ (Figure 3.1).

Waiting time phase:

3. Both parties wait time Δt.

Oblivious key phase: Same as in $\Pi^{\mathbf{BBCS}}$ (Figure 3.1).

Transfer phase: Same as in $\Pi^{\mathbf{BBCS}}$ (Figure 3.1).

Alice's output: $\perp$.
Bob's output: m_b.

Figure 3.3: BBCS OT protocol in the bounded-quantum-storage model.

Security. We just comment on the security against dishonest Bob because the justification for the security against dishonest Alice is the same as in the original BBCS protocol, $\Pi^{\mathbf{BBCS}}$ (see Section 3.2.1).

Under the BQS assumption, the waiting time (Δt) effectively prevents Bob from holding *a large fraction* of qubits until Alice reveals the bases choices $\boldsymbol{\theta}^{\mathsf{A}}$ used during the *BB84 phase*. This comes from the fact that a dishonest Bob is forced to measure a fraction of the qubits, leading him to lose information about Alice's bases $\boldsymbol{\theta}^{\mathsf{A}}$.

More specifically, Damgård et al. [143] showed that, with overwhelming probability, the loss of information about Alice's oblivious key ($\mathsf{ok}^{\mathsf{A}}_{I_1}$) is described by a lower bound on the min-entropy [91]

$$H_{\min}(\mathsf{ok}^{\mathsf{A}}_{I_1}|E) \geq \frac{1}{4}n - \gamma n - l - 1.$$

Similarly to the $\mathcal{F}_{\mathbf{COM}}-$hybrid model, the min-entropy value has to be bounded by a factor proportional to the security parameter n. To render a positive bound, we derive an upper bound on the fraction of qubits that can be saved in the receiver's quantum memory, while preserving the security of the protocol, i.e. $\gamma < \frac{1}{4}$.

The above upper bound was later improved by König et al. [124] to $\gamma < \frac{1}{2}$. The authors also showed that the BQS model is a special case of the noisy-quantum-storage model. Subsequently, based on higher-dimensional mutually unbiased bases, Mandayam and Wehner [155] presented a protocol that is still secure when an adversary cannot store even a small fraction of the transmitted pulses. In this latter work, the storage rate γ approaches 1 for increasing dimension.

Composability. The initial proofs given by Damgård et al. [122, 143] were only developed under the stand-alone security model [156]. In this model the composability of the protocol is not guaranteed to be secure. These proofs were extended by Wehner and Wullschleger [156] to a simulation-based framework that guarantees sequential composition. Also, in a parallel work, Fehr and Schaffner developed a sequential composability framework under which $\Pi^{\mathbf{BBCS}}_{\mathbf{bqs}}$ is secure considering the BQS model.

The more desirable quantum-UC framework was extended by Unruh and combined with the BQS model [150]. In Unruh's work, he developed the concept of BQS-UC security which, as in UC security, implies a very similar composition theorem. The only difference is that in the BQS-UC framework we have to keep track of the quantum memory-bound used by the machines activated during the protocol. Under this framework, Unruh follows a different approach as he does not use the protocol $\Pi^{\mathbf{BBCS}}_{\mathbf{bqs}}$ (Figure 3.3). He presents a BQS-UC secure commitment protocol and composes it with the $\Pi^{\mathbf{BBCS}}_{\mathcal{F}_{\mathbf{COM}}}$ protocol (Figure 3.2) in order to get a constant-round protocol that BQS-UC-emulates any two-party functionality.

3.2.5 Noisy-quantum-storage model

The noisy-quantum-storage model, or NQS model for short, is a generalization of the BQS model. In the NQS model, the adversaries are allowed to keep any fraction ν of the transmitted qubits (including the case $\nu = 1$) but their quantum memory is assumed to be noisy [124], i.e. it is impossible to store qubits for some amount of time (Δt) without undergoing decoherence.

More formally, the decoherence process of the qubits in the noisy storage is described by a completely positive trace preserving (CPTP) map (also called channel) $\mathcal{C} : \mathcal{P}(\mathcal{H}_{\text{in}}) \to \mathcal{P}(\mathcal{H}_{\text{out}})$, where $\mathcal{H}_{\text{in/out}}$ is the Hilbert space of the stored qubits before (in) and after (out) the storing period Δt and $\mathcal{P}(\mathcal{H})$ is the set of positive semi-definite operators with unitary trace acting on an Hilbert space $\mathcal{H}$. $\mathcal{C}$ receives a quantum state $\rho \in \mathcal{H}_{\text{in}}$ at time t and outputs a quantum state $\rho' \in \mathcal{H}_{\text{out}}$ at a later time $t + \Delta t$.

With this formulation, we can easily see that the BQS model is a particular case of the NQS. In BQS, the channel is of the form $\mathcal{C} = \mathbb{1}^{\otimes \nu n}$, where the storage rate ν is the fraction of transmitted qubits stored in the quantum memory. The most studied scenario is restricted to $n-$fold quantum channels, i.e. $\mathcal{C} = \mathcal{N}^{\otimes \nu n}$ [123, 124, 157], where the channel $\mathcal{N}$ is applied independently to each individual stored qubit. In this particular case, it is possible to derive specific security parameters.

Protocols. The protocol from BQS model $\Pi^{\mathbf{BBCS}}_{\mathbf{bqs}}$ is also considered to be secure in the NQS model [157]. However, the first proposed protocol analysed in this general NQS model was developed by König et al. [124]. This protocol draws inspiration from the research line initiated by Cachin, Crépeau and Marcil [158] about classical OT in the bounded-classical-storage model [159, 160]. Similar to these works [158–160], the protocol presented by König et al. [124] uses the following two important techniques in its classical post-processing phase: encoding of sets and interactive hashing. The former is defined as an injective function $\mathsf{Enc} : \{0,1\}^t \to T$, where T is a set of all subsets of $[n]$ with size $n/4$. The latter is a two-party protocol between Alice and Bob with the following specifications. Bob inputs some message W^t and both parties receive two messages W^t_0 and W^t_1 such that there exists some $b \in \{0,1\}$ with $W^t_b = W^t$. The index b is unknown to Alice, and Bob has little control over the choice of the other message W^t, i.e. it is randomly chosen by the functionality.

In this section, we only present the naïve protocol presented in the original paper [124] as it is enough to give an intuition on the protocol. Although both $\Pi^{\mathbf{BBCS}}_{\mathbf{bqs}}$ and $\Pi^{\mathbf{BBCS}}_{\mathbf{nqs}}$ protocols are different, we keep a similar notation for a comparison purpose. The protocol $\Pi^{\mathbf{BBCS}}_{\mathbf{nqs}}$ (Figure 3.4) goes as follows. The first two phases (*BB84* and *Waiting time*) are the same as in $\Pi^{\mathbf{BBCS}}_{\mathbf{bqs}}$ (Figure 3.3). Then, both parties generate a very similar resource to oblivious keys, named *weak string erasure* (WSE). After the WSE generation, Alice also holds the totality of the key ok^{A}, while Bob holds a fourth of this key, i.e. the tuple $(I, \mathsf{ok}^{\mathsf{B}} := \mathsf{ok}^{\mathsf{A}}_I)$ where I is the set of indexes they measured in the same basis and its size is given by $|I| = \frac{n}{4}$. Then, along with a method of encoding sets into binary strings, both parties use interactive hashing to generate two index subsets, I_0 and I_1. The two subsets (I_0 and I_1) together with two 2−universal hash functions are enough for Alice to generate her output messages (m_0, m_1) and for Bob to get his bit choice along with the corresponding message (b, m_b). For more details on the protocols for encodings of sets and interactive hashing, we refer to Ding et al. [159] and Savvides [160].

Security. Based on the original BQS protocol (Figure 3.3), the first proofs in the NQS model were developed by Schaffner, Wehner and Terhal [123, 161]. However, in these initial works, the authors only considered individual-storage attacks, where the adversary treats all incoming qubits equally. Subsequently, Schaffner [157] was able to prove the security of $\Pi^{\mathbf{BBCS}}_{\mathbf{bqs}}$ against arbitrary attacks in the more general NQS model defined by König et al. [124].

In this more general NQS model, the security of both protocols $\Pi^{\mathbf{BBCS}}_{\mathbf{bqs}}$ and $\Pi^{\mathbf{BBCS}}_{\mathbf{nqs}}$ (Figures 3.3 and 3.4) against a dishonest receiver depends on the ability to set a lower-bound on the min-entropy of the "unknown" key $\mathsf{ok}^{\mathsf{A}}_{I_{1-b}}$ given the receiver's quantum side information. His quantum side information is given by the output of the quantum channel $\mathcal{C}$ when applied to the received states. More formally, one has to lower-bound the expression $H_{\min}\left(\mathsf{ok}^{\mathsf{A}}_{I_{1-b}}|\mathcal{C}\left(Q_{\text{in}}\right)\right)$, where Q_{in} denotes the subsystem of the received states before undergoing decoherence. It is proven [124] that this lower-bound depends on the receiver's maximal success probability of correctly decoding a randomly chosen n-bit string $x \in \{0,1\}^n$ sent over the quantum channel $\mathcal{C}$, i.e. $P^{\mathcal{C}}_{\text{succ}}(n)$.

For particular channels $\mathcal{C} = \mathcal{N}^{\otimes\nu}$, König et al. [124] concluded that security in the NQS model can be obtained in case

$$c_{\mathcal{N}} \cdot \nu < \frac{1}{2},$$

where $c_{\mathcal{N}}$ is the classical capacity of quantum channels $\mathcal{N}$ satisfying a particular property (strong-converse property).

3.2.6 Experimental attacks

Although QKD and QOT protocols are proved to be theoretically secure, experimental implementations may come with loopholes that allow to break their security. This mismatch between theory and practice comes from the fact that theoretical proofs usually assume that the physical apparatus of honest parties cannot be hacked. However, imperfections in both the generation and measurement the qubits can be exploited in multiple ways to perform quantum attacks. We refer the interested reader to proper review articles [162, 163] on QKD attacks and possible mitigation measures. Here, we briefly discuss the

Naïve $\Pi^{\mathbf{BBCS}}_{\mathbf{nqs}}$ **protocol**

Parameters: n, security parameter; $\mathcal{F}$ two-universal family of hash functions.
Alice's input: $\perp$.
Bob's input: $\perp$.

BB84 phase: Same as in $\Pi^{\mathbf{BBCS}}$ (Figure 3.1).

Waiting time phase: Same as in $\Pi^{\mathbf{BBCS}}_{\mathbf{bqs}}$ (Figure 3.3).

Weak String Erasure phase: Similar to *Oblivious key phase* of $\Pi^{\mathbf{BBCS}}$ (Figure 3.1).

4. Alice reveals to Bob the bases $\boldsymbol{\theta}^{\mathsf{A}}$ used during the *BB84 phase* and sets her oblivious key to $\mathsf{ok}^{\mathsf{A}} := \boldsymbol{x}^{\mathsf{A}}$.

5. Bob computes $\mathsf{e}^{\mathsf{B}} = \boldsymbol{\theta}^{\mathsf{B}} \oplus \boldsymbol{\theta}^{\mathsf{A}}$. Then, he defines $I = \{i : \mathsf{e}^{\mathsf{B}}_i = 0\}$ and sets $\mathsf{ok}^{\mathsf{B}} := \boldsymbol{x}^{\mathsf{B}}_I$.

6. If $|I| < n/4$, Bob randomly adds elements to I and pads the corresponding positions in ok^{B} with 0s. Otherwise, he randomly truncates I to size $n/4$, and deletes the corresponding values in ok^{B}.

Interactive hashing phase:

7. Alice and Bob execute interactive hashing with Bob's input W to be equal to a description of $I = \mathsf{Enc}(W)$. They interpret the outputs W_0 and W_1 as descriptions of subsets I_0 and I_1 of $[n]$.

Transfer phase:

5. Alice generates random $f_0, f_1 \leftarrow_{\$} \mathcal{F}$ and sends them to Bob.

6. Alice computes the pair of messages (m_0, m_1) as $m_i = f_i(\mathsf{ok}^{\mathsf{A}}_{I_i})$.

7. Bob computes $b \in \{0, 1\}$ by comparing $I = I_b$ and computes $m_b = f_b(\mathsf{ok}^{\mathsf{B}}_I)$.

S **output:** $(m_0, m_1) \in \{0, 1\}^l$ (two messages).
R **output:** (b, m_b) where $b \in \{0, 1\}$ (bit choice).

Figure 3.4: BBCS OT protocol in the noisy-quantum-storage model.

impact of these attacks on BBCS-based QOT protocols.

QOT attacks

It is important to stress that there is a fundamental difference between QKD and QOT protocols. In QKD, both parties can cooperate to detect an external attack, whereas, in QOT, both parties are distrustful of each other. Moreover, QKD external attacks pressupose that the adversary has physical access to the quantum channel and is able to play some sort of man-in-the-middle attack. Regarding QOT protocols, both parties are already linked by a quantum channel. Therefore, in principle, QOT attacks require less engineering effort to succeed as the adversary is already using the quantum channel.

According to the security properties of QOT, Alice must not know Bob's bit b and Bob must not know m_{1-b}. Regarding BBCS-based QOT protocols, its security depends on the security requirements of oblivious keys. Informally, this means that Alice must not be able to know which set of indexes is known by Bob (i.e. e^{B}) and Bob must have limited knowledge on Alice's key (i.e. ok^{A}). These two pieces of information (e^{B} and ok^{A}) can be easily deduced if the adversary has access to the quantum bases used by the other party ($\boldsymbol{\theta}^{\mathsf{A}}$ or $\boldsymbol{\theta}^{\mathsf{B}}$). Indeed, Alice gets e^{B} by computing $\boldsymbol{\theta}^{\mathsf{B}} \oplus \boldsymbol{\theta}^{\mathsf{A}}$ and Bob gets ok^{A} by measuring all the qubits with Alice's bases $\boldsymbol{\theta}^{\mathsf{A}}$. Therefore, the main aim of the adversary is to use his quantum channel to gain some information (or control) about the set of bases used by the other. Two of the most common attacks on quantum systems are faked-state attacks [164] (FSA) and trojan-horses attacks [165] (THA). The former targets measurement apparatus only and the latter can target both preparation and measurement apparatus. In a prepare-and-measure setting, FSA can only be used by Alice (sender) while THA can be used by both. For the sake of exposition, let us see how these two approaches can be used to attack both $\Pi^{\mathbf{BBCS}}_{\mathbf{bqs}}$ and $\Pi^{\mathbf{BBCS}}_{\mathcal{F}_{\mathbf{COM}}}$ protocols. The attacks on $\Pi^{\mathbf{BBCS}}_{\mathbf{nqs}}$ follow the same reasoning but the notation vary slightly.

We denote by $\tilde{\boldsymbol{\theta}}^{\mathsf{B}}_{J} \leftarrow \mathcal{A}_{\mathbf{qok}}(J)$ Alice's quantum hacking procedure ($\mathcal{A}_{\mathbf{qok}}(J)$) that breaks the security requirements of oblivious keys and provides her with Bob's bases ($\tilde{\boldsymbol{\theta}}^{\mathsf{B}}_{J}$) from index set J. Similarly for Bob, i.e. $\tilde{\boldsymbol{\theta}}^{\mathsf{A}}_{J} \leftarrow \mathcal{B}_{\mathbf{qok}}(J)$.

FSA attacks. These attacks can be performed with well crafted optical signals that allow Alice to take control over Bob's measurement outcomes. In summary, as described by Jain et al. [166], when both parties' bases coincide, Bob's detector clicks; when these are orthogonal, he gets no detection event ($\perp$). In other words, Alice forces Bob to only use the measurements where their bases coincide. So, the indexes corresponding to no detection events will be discarded by both parties whereas the others will be used in the rest of the protocol. This way, Alice gains full knowledge about Bob's bases and can easily distinguish I_0 from I_1. Note that Alice does not have to attack all measurement turns. She only needs one successful FSA to guess one basis. This happens with high

$\Pi^{\mathsf{A}}_{\mathbf{FSA}}$ attack

Alice's input: set of indexes J of size q.

1. Alice performs some faked-state attack $\left\{\tilde{\theta}^{\mathsf{B}}_j\right\}_{j\in J} \leftarrow \mathcal{A}_{\mathbf{qok}}(J)$ where $\tilde{\theta}^{\mathsf{B}}_j \in \{+,\times\}$ or $\tilde{\theta}^{\mathsf{B}}_j = \perp$.
2. If $\exists j \in J$ such that $\tilde{\theta}^{\mathsf{B}}_j \neq \perp$:
   (a) $b = 0$ if $j \in I_b$, ;
   (b) $b = 1$ if $j \notin I_b$.
3. Otherwise, sets $b = \perp$.

Alice's output: b.

Figure 3.5: Alice faked-state attack to $\Pi^{\mathbf{BBCS}}_{\mathbf{bqs}}$ and $\Pi^{\mathbf{BBCS}}_{\mathcal{F}_{\mathbf{COM}}}$ protocols.

probability in the number of attacks q,

$$Pr[\text{Success Alice attack in } q \text{ rounds}] = 1 - \left(\frac{1}{2}\right)^q.$$

From this basis, Alice can deduce to which set (I_0 or I_1) the corresponding index (j) belongs. As Bob computes his message m_b with the set where their basis coincide, and since Alice computes both messsages m_0 and m_1 out of both sets, she can determine Bob's message m_b. Indeed, m_b will be the message that comes from the set where j belongs. The attack $\Pi^{\mathsf{A}}_{\mathbf{FSA}}$ against both $\Pi^{\mathbf{BBCS}}_{\mathbf{bqs}}$ and $\Pi^{\mathbf{BBCS}}_{\mathcal{F}_{\mathbf{COM}}}$ is summarized in Figure 3.5.

THA attacks. These types of attacks are performed by sending bright pulses into the equipment under attack and scanning through the different reflections to obtain the bases used. Likewise the FSA, Alice only needs to successfully find one basis used by Bob. By comparing her basis and Bob's basis to that particular turn, she can find Bob's bit b. This attack $\Pi^{\mathsf{A}}_{\mathbf{THA}}$ is summarized in Figure 3.6.

Bob's attack through THA is more challenging. Not only he has to successfully guess *all* Alice's bases, he also has to be able to correctly measure the corresponding qubits after leaking the sender's bases. Without the help of quantum memories, this procedure is much more difficult to succeed. Bob's attack $\Pi^{\mathsf{B}}_{\mathbf{THA}}$ is summarized in Figure 3.7.

$\Pi^{\mathsf{A}}_{\mathbf{THA}}$ attack

Alice's input: one index element, j.

1. Alice performs some trojan-horse attack $\left\{\tilde{\theta}^{\mathsf{B}}_j\right\} \leftarrow \mathcal{A}_{\mathbf{qok}}(i)$ where $\tilde{\theta}^{\mathsf{B}}_j \in \{+, \times\}$.
2. Alice compares the received basis $\tilde{\theta}^{\mathsf{B}}_j$ with her corresponding base θ^{A}_j. Denote by $\tilde{\mathsf{e}}^{\mathsf{B}}_j := \tilde{\theta}^{\mathsf{B}}_j \oplus \theta^{\mathsf{A}}_j$.
3. Upon receiving I_b from R:
   (a) $b = \tilde{\mathsf{e}}^{\mathsf{B}}_j$ if $j \in I_b$;
   (b) $b = 1 - \tilde{\mathsf{e}}^{\mathsf{B}}_j$ if $j \notin I_b$.

Alice's output: b.

Figure 3.6: Alice trojan-horse attack to $\Pi^{\mathbf{BBCS}}_{\mathbf{bqs}}$ and $\Pi^{\mathbf{BBCS}}_{\mathcal{F}_{\mathbf{COM}}}$ protocols.

$\Pi^{\mathsf{B}}_{\mathbf{THA}}$ attack

Parameters: n, security parameter..

1. Bob performs some trojan-horse attack to all qubits sent by Alice, i.e. $\left\{\tilde{\theta}^{\mathsf{A}}_i\right\}_{i\in[n]} \leftarrow \mathcal{B}_{\mathbf{qok}}([n])$ where $\tilde{\theta}^{\mathsf{A}}_i \in \{+, \times\}$.
2. Bob measures the received states $\left|\boldsymbol{x}^{\mathrm{A}}\right\rangle_{\boldsymbol{\theta}^{\mathrm{A}}}$ with the correct bases, $\left\{\tilde{\theta}^{\mathsf{A}}_i\right\}_{i\in[n]}$.

Bob's output: ok^{A}.

Figure 3.7: Bob trojan-horse attack to $\Pi^{\mathbf{BBCS}}_{\mathbf{bqs}}$ and $\Pi^{\mathbf{BBCS}}_{\mathcal{F}_{\mathbf{COM}}}$ protocols.

Countermeasures

We have seen how two well-known quantum hacking techniques can undermine the security of oblivious keys and, consequently, the security of oblivious transfer. Fortunately, there are some countermeasures that can be applied that prevent such attacks from breaking the system's security. These countermeasures can be divided into two categories: security patches that tackle specific vulnerabilities and novel schemes that allow faulty devices.

Regarding the two presented possible attacks, it is commonly possible to implement security patches that prevent them. FSA can be prevented by placing an additional detector (usually called watchdog) at the entrance of the receiver's measurement device. This detector monitors possible malicious radiation that blinds his detector. Also, THA can be blocked by an isolator placed at both parties entrance devices. However, as mentioned by Jain et al. [166] these two countermeasures only prevent these attacks perfectly in case the isolators and watchdogs work at all desired frequencies, which is not the case in practice.

There is a research line focused on the study of security patches for each technological loophole [167]. However, this approach pursues the difficult task of approximating the experimental implementations to the ideal protocols. It would be more desirable to develop protocols that already consider faulty devices and are robust against any kind of quantum hacking attack. This is the main goal of device-independent (DI) cryptography, where we drop the assumption that quantum devices cannot be controlled by the adversary and we treat them simply as black-boxes [168, 169]. Here, we give a general overview of the state-of-the-art of DI protocols. For a more in-depth description, we refer to the corresponding original works.

Kaniewski-Wehner DI protocol [170]. The first DI protocol of QOT was presented in a joint work by Kaniewski and Wehner [170] and its security proof was improved by Ribeiro et al. [171]. The protocol was proved to be secure in the noisy-quantum-storage (NQS) model as it uses the original NQS protocol $\Pi_{\mathbf{nqs}}^{\mathbf{BBCS}}$ (Figure 4) for trusted devices. It analyzes two cases leading to slightly different protocols.

First, they assume that the devices have the same behaviour every time they are used (memoryless assumption). This assumption allows for testing the devices independently from the actual protocol, leading to a DI protocol in two phases: device-testing phase and protocol phase . Under this memoryless assumption, one can prove that the protocol is secure against general attacks using proof techniques borrowed from [124]. Then, they analyse the case *without* the memoryless assumption. In that case, it is useless to test the devices in advance as they can change their behaviour later. Consequently, the structure of the initial DI protocol (with two well-separated phases) has to be changed to accommodate this more realistic scenario. That is, the rounds for the device-testing phase have to be intertwined with the rounds for the protocol phase.

As a common practice in DI protocols, the DI property comes from some violation of Bell inequalities [172], which ensures a certain level of entanglement. This means that, in the protocol phase, the entanglement-based variant of $\Pi_{\mathbf{nqs}}^{\mathbf{BBCS}}$ must be used. Here, the difference lies in the initial states prepared by Alice, which, for this case, are maximally entangled states $|\Phi^+\rangle\langle\Phi^+|$, where $|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. The Bell inequality used in this case comes from the Clauser-Holt-Shimony-Horne (CHSH) inequality [173].

Broadbent-Yuen DI protocol [174]. More recently, Broadbent and Yuen [174] used the $\Pi_{\mathbf{bqs}}^{\mathbf{BBCS}}$ (Figure 3) to develop a DI protocol in the BQS model. Similar to Kaniewski and Wehner's work, the protocol is secure under the memoryless assumption. However, they do not require non-communication assumptions that ensure security from Bell inequality violations. Instead of using the CSHS inequality, their work uses a recent self-testing protocol [175, 176] based on a post-quantum computational assumption (hardness of Learning with Errors (LWE) problem [177]).

Ribeiro-Wehner MDI protocol [178]. Ribeiro and Wehner [178] developed an OT protocol in the measurement-device-independent (MDI) regime [179] to avoid the technological challenges in the implementation of DI protocols [180]. In this regime, two parties perform QOT with untrusted measurement devices while trusting their sources. In addition, this work was motivated by the fact that, so far, there is no security proof in the DI setting. Furthermore, many attacks on the non device-independent protocols affect the measurement devices rather than the sources [181]. The presented protocol follows the research line of König et al. [124] and start by executing a weak string erasure in the MDI setting (MDI-WSE phase). For this reason, it is also proved to be secure in the NQS model.

The initial MDI-WSE phase goes as follows. Both Alice and Bob send random states $\left|\boldsymbol{x}^{\mathsf{A}}\right\rangle_{\boldsymbol{\theta}^{\mathsf{A}}}$ and $\left|\boldsymbol{x}^{\mathsf{B}}\right\rangle_{\boldsymbol{\theta}^{\mathsf{B}}}$, respectively, to an external agent that can be controlled by the dishonest party. The external agent performs a Bell measurement on both received states and announces the result. Bob flips his bit according to the announced result to match Alice's bits. Then, both parties follow the $\Pi_{\mathbf{nqs}}^{\mathbf{BBCS}}$ protocol (Figure 4) from the waiting time phase onward. A similar protocol was presented by Zhou et al. [182] which additionally takes into account error estimation to improve the security of the protocol.

Chapter 4

Classical and quantum oblivious transfer

Secure multiparty computation (SMC) has the potential to be a disruptive technology in the realm of data analysis and computation. It enables several parties to compute virtually any function while preserving the privacy of their inputs. However, most of its protocols' security and efficiency relies on the security and efficiency of oblivious transfer (OT). For this reason, it is fundamental to understand the pros and cons of classical and quantum approaches. In this chapter, we start by analysing both the security and efficiency of classical OT protocols. Then, we compare these classical protocols with their quantum analog. However, we note that classical and quantum approaches use different information medium. Also, classical technology is indeed much more mature than quantum technology. These two observations make it dubious how to perform such a comparison.

In Chapter 3, we reviewed several quantum OT protocols and, in particular, we explored BBCS-based QOT protocols. Beyond being resistant to quantum computer attacks, these protocols provide a practical way to perform OT within SMC. These are divided into two independent phases: oblivious key phase and transfer phase. The first phase corresponds to a precomputation phase that uses quantum technologies and is independent of the parties input elements (m_0, m_1 and b). The second phase only uses classical communication and is based on the precomputed elements from the first phase (oblivious keys). It can be argued that the precomputation phase is not so hungry-efficient as the transfer phase. This comes from the fact that the precomputation is independent of the parties' inputs and, thus, can be performed way before starting an SMC execution. Since the classical OT protocols can also be divided into these two phases, we can compare the transfer phase of both quantum and classical approaches. Furthermore, we do not need quantum equipment to be run concurrently with the SMC execution.

4.1 Classical oblivious transfer

Let us start by presenting the Bellare-Micali (BM) OT protocol [2] based on public key Diffie-Hellman. This exposition aims to shed some light on the issues related to classical OT implementations. The security and efficiency issues explored in this section also apply to most of the major classical protocols [85–87].

We consider $\mathbb{G}_q$ to be a subgroup of $\mathbb{Z}_p^*$ with generator g and order q, where p is prime and $p = 2q+1$. Also, we assume public knowledge on the value of some constant $C \in \mathbb{G}_q$. This constant guarantees that Bob follows the protocol. Also, for simplicity, we assume the protocol uses a random oracle described as a function H. For comparison purposes with quantum OT version presented in Chapter 3, we split the BM OT protocol into two phases: precomputation phase and transfer phase. The first phase sets the necessary resources to execute the oblivious transfer in the second phase. The BM OT protocol Π_{BM} is shown in Fig. 4.1.

Π_{BM} protocol

Alice's input: $(m_0, m_1) \in \{0,1\}^l$ (two messages).
Bob's input: $b \in \{0,1\}$ (bit choice).

(Precomputation phase)

1. Bob randomly generates $k \in \mathbb{Z}_q$ and computes g^k.
2. Alice randomly generates $r_0, r_1 \in \mathbb{Z}_q$ and computes g^{r_0} and g^{r_1}.

(Transfer phase)

3. Bob sets $\mathsf{pk}_b := g^k$. Also, he computes $\mathsf{pk}_{b\oplus 1} = C \cdot \mathsf{pk}_b^{-1}$.
4. Bob sends both public keys $(\mathsf{pk}_0, \mathsf{pk}_1)$ to Alice.
5. Alice checks if $(\mathsf{pk}_0, \mathsf{pk}_1)$ were correctly generated by computing their product: $C = \mathsf{pk}_0 \times \mathsf{pk}_1$.
6. Alice computes and sends to Bob the two tuples: $E_0 = (g^{r_0}, H(\mathsf{pk}_0^{r_0}) \oplus m_0)$ and $E_1 = (g^{r_1}, H(\mathsf{pk}_1^{r_1}) \oplus m_1)$ for some hash function H.
7. Bob is now able to compute $H(\mathsf{pk}_b^{r_b})$ and recover m_b.

Alice's output: $\perp$.
Bob's output: m_b.

Figure 4.1: Bellare-Micali classical OT protocol divided into two phases [2].

4.1.1 Security issues

The Bellare-Micali OT protocol is secure if it complies with both the concealing and obliviousness property. The former is achieved because Bob does not send any information that reveals his input bit choice b to Alice. The latter relies on Alice's ability to keep her randomly generated elements r_0 and r_1 private. Thus, the obliviousness property is compromised if Bob is able to compute the discrete logarithm of g^{r_i} for $i = 0, 1$ (discrete logarithm problem).

The hardness of the discrete logarithm problem on cyclic groups is the basis of several other important protocols. Thus, it is crucial to understand its security limits. Nevertheless, it remains to be proven whether, given a general cyclic group $\mathbb{G}_q$ with generator g and order q, there exists a polynomial-time algorithm that computes r from g^r, where $r \in \mathbb{Z}_q$. Indeed, the BM OT protocol's security relies on the assumption that Bob has limited computational power and is not able to compute the discrete logarithm of a general number.

Although the general discrete logarithm problem is not known to be tractable in polynomial-time, there are specific cases where it is possible to compute it efficiently. This leads to some classical attacks where the structure of the cyclic group considered is not robust enough. As an example, if a prime p is randomly generated without ensuring that $p - 1$ contains a big prime p_b in its decomposition, it is possible to use a divide-and-conquer technique [183] along with some other methods (Shank's method [184], Pollard's rho [185], Pollard's lambda [185]) to solve the discrete logarithm problem. In this case, the computation time will only depend on the size of p_b. So, the smaller the prime p_b, the faster the algorithm can be. In order to avoid these types of attacks, it is recommended to use safe primes, i.e. $p = 2q + 1$ prime where q is also prime. However, it is computationally more expensive to find safe primes because they are less frequent when compared with prime numbers. Beyond the cyclic group structure, it is also important to find big enough prime numbers p. Otherwise, it is possible to compute the discrete logarithm in an acceptable time. As reported in [186], after one week of precomputation, it is possible to compute the discrete logarithm in a 512-bit group in one minute by using the number field sieve algorithm. So, by following this method, after a week-long computation, Bob would be able to find both messages m_0 and m_1 of the BM OT protocol in one minute. In an SMC scenario based on the Yao approach [13], where each OT performed corresponds to one input bit of Alice and the chosen group parameters are fixed, Bob would be able to get the keys corresponding to both 0 and 1 bit and, consequently, he would be able to discover all Alice's inputs. Therefore, at the expense of efficiency, it is necessary to use big enough prime numbers (2048-bit or larger), for which these classical attacks could not be feasibly implemented.

We have just seen specific examples where it is possible to break the security of OT protocol using classical techniques. However, it is known that it is possible to break the general discrete logarithm problem with a quantum computer. In 1995, Peter Shor published a quantum algorithm that is able to solve both prime factorization and discrete logarithm problems in polynomial-time [21]. This remarkable finding poses a threat to most of our currently deployed asymmetric cryptographic protocols (Rivest-Shamir-Adleman, elliptic-curve cryptography and Diffie-Hellman key exchange) as they have their security based on these computational assumptions. Therefore, in the BM OT protocol Bob would be able to perform two attacks with the help of a quantum computer:

Quantum attack 1:

1. Bob computes the discrete logarithm of $g^{r_{b\oplus 1}}$ received from Alice using Shor's algorithm, i.e. $r_{b\oplus 1} = \log_g g^{r_{b\oplus 1}}$.
2. Bob is then able to compute $H\big((g^{r_b})^k\big) = H(\mathsf{pk}_b^{r_b})$ and $H(\mathsf{pk}_{b\oplus 1}^{r_{b\oplus 1}})$ and get both messages m_b and m_{b-1}.

Quantum attack 2:

1. Bob computes the discrete logarithm of $\mathsf{pk}_{b\oplus 1}$ with the Shor's algorithm, i.e. $s = \log_g \mathsf{pk}_{b\oplus 1}$.
2. Bob is then able to compute $H\big((g^{r_b})^k\big) = H(\mathsf{pk}_b^{r_b})$ and $H\big((g^{r_{b\oplus 1}})^s\big) = H(\mathsf{pk}_{b\oplus 1}^{r_{b\oplus 1}})$ and get both messages m_b and $m_{b\oplus 1}$.

In the research literature, there are mainly two approaches to tackle this issue: the development of protocols with assumptions on the computational power of quantum computers or the development of protocols that make use of quantum technology. The former is known as post-quantum cryptography [187] and its public-key cryptography protocols are generally more demanding due to the nature of the computational assumptions used. It is also worth stressing that these computational assumptions are still unproven and have survived just a few years of scrutiny, rendering it likely to be attacked in the near future. The latter is known as quantum cryptography [188]. It provides solutions without relying on asymmetric cryptography but it drastically increases the cost of technological equipment required. Finally, it is important to note that, in general, quantum protocols do not suffer from *intercept now - decipher later* attack (everlasting security) because they base their security on quantum theory. On the contrary, this possible threat is always present in protocols based on computational assumptions.

4.1.2 Efficiency issues

In the previous section, we noted that every mitigation process used to increase security would bring a downside in efficiency: generating safe primes is more demanding, computing bigger exponents and module primes is heavier in general, and using post-quantum solutions require stronger computational assumptions and thus tends to increase the computational complexity.

Now, let us understand the efficiency limitations of the BM OT protocol. We start by looking at the operations used in the protocol (random number generation, modular multiplication, modular inversion, modular exponentiation, hash function evaluation, XOR operation) from which the most demanding operation is modular exponentiation. For this reason, the complexity of BM OT heavily depends on the complexity of modular exponentiation. The number of modular exponentiations executed in each phase is summarised in the Table 4.1.

	Alice	Bob
Precomputation phase	2	1
Transfer phase	2	1

Table 4.1: Number of modular exponentiations in the BM protocol for each phase.

One of the most efficient methods to compute general modular exponentiation with $n-$bit numbers is through a square-and-multiply algorithm along with Karatsuba multiplication. The former method takes $\mathcal{O}(n)$ multiplications and the latter has complexity $\mathcal{O}(n^{1.58})$. Thus, the overall method takes $\mathcal{O}(n^{2.58})$ $n-$bit operations [189]. To set an overestimation on the OT generation rate, let us only consider the time (in CPU cycles) required to compute all modular exponentiation operations. We can use the following expression:

$$\left(\frac{C_{mexp}}{C_{cycles}} \times N_{mexp}\right)^{-1} \tag{4.1}$$

where C_{mexp} is the number of CPU cycles required to compute one modular exponentiation, C_{cycles} is the CPU frequency (number of cycles per second) and N_{mexp} is the number of modular exponentiations performed in the OT implementation. This expression only renders an overestimation because it depends on both the implementation of the modular exponentiation operation and the CPU frequency used.

Considering a standard CPU operating around 2.5 GHz ($C_{cycles} = 2.5 \times 10^9$ cycles per second) and a very efficient implementation of modular exponentiation [190] ($C_{mexp} \sim$

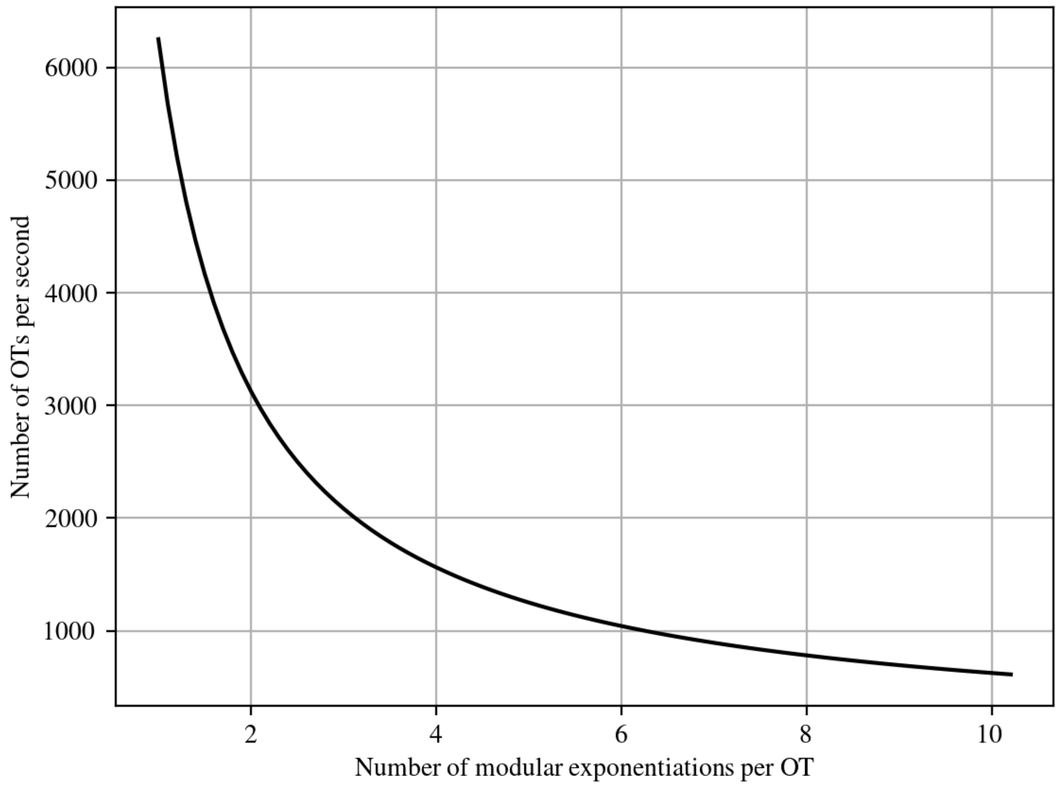

Figure 4.2: Plot of expression (4.1) on the overestimation of OT rate against the number of modular exponentiation operations required per OT.

400 000 CPU cycles), the BM OT protocol would be able to perform at most ~ 1041 BM OTs in one second as represented in Fig. 4.2. Note that this is a very loose overestimation of the number of OT per second. Here, we just took into consideration the computational complexity of modular exponentiation, and we assumed that all the other operations do not have a big impact on the computation time. So, we can conclude that the real OT rate must be well below this threshold. As reported in [3], it takes around 18 ms to generate a similar OT protocol: Naor-Pinkas OT [86], which requires 5 modular exponentiations. This corresponds to a rate of just 56 OT per second.

The OT rates presented above lead to serious constraints on the execution of SMC protocols that rely on OT. The Yao SMC protocol [13] uses boolean circuits to privately compute the desired functionality and requires as many OT as half the number of input wires. Thus, the execution time of the OT phase of the Yao protocol with a 32 000 input boolean circuit would take at least 16 s using our rough OT rate estimation and around 2 min 23 s using Naor-Pinkas OT rate. In a deployment environment where several rounds of the same circuit are evaluated, this approach becomes impractical and higher rates must be achieved.

4.1.3 OT extension protocols

Because most of the required computation to achieve OT comes from asymmetric cryptographic primitives that use modular exponentiation, it would be desirable to substitute it by more efficient methods. Symmetric cryptography has the advantage to be more efficient than asymmetric cryptography. In addition, all known quantum attacks to symmetric cryptography based on the Grover's algorithm only provide a quadratic advantage over classical approaches, which can be mitigated by doubling the size of the symmetric keys [187]. Unfortunately, as we saw in the beginning of Chapter 3, Impagliazzo and Rudich's result [20] implies that OT protocols require asymmetric cryptographic assumptions. This means OT cannot be performed by symmetric cryptographic tools alone.

Nonetheless, researchers developed some OT schemes to circumvent Impagliazzo and Rudich's result using hybrid protocols mixing symmetric and asymmetric cryptography. This idea was introduced by Beaver [191], where he showed that it is possible to extend the number of OT using symmetric cryptography when a small number of base OT is created using asymmetric cryptography. Although Beaver's protocol was very inefficient, it paved the way to more efficient implementations [3, 192–195]. Currently, one of the most efficient protocols is able to generate around 10 million OTs in 2.62 s [3]. Because these protocols use a small number of base OTs and quantum secure symmetric tools, the security of the extended protocol mainly depends on the security of the base OT protocol. Moreover, the protocol that we analyse in Section 4.2.3 [3] is not secure against malicious parties and must only be deployed in a semi-honest environment. Protocols that are secure against malicious parties need an extra consistency check phase which increases their complexity [4, 195] as we see in Section 4.2.3.

4.2 Oblivious transfer complexity analysis

In this section, we compare the complexity of the transfer phase of an optimized version of the BBCS-based QOT protocols ($\Pi^{\mathbf{BBCS}}_{\mathcal{F}_{\mathbf{COM}}}$ and $\Pi^{\mathbf{BBCS}}_{\mathbf{bqs}}$) presented before and several well known classical protocols. We start by explaining the optimization.

4.2.1 Optimization

Recall that both $\Pi^{\mathbf{BBCS}}_{\mathcal{F}_{\mathbf{COM}}}$ and $\Pi^{\mathbf{BBCS}}_{\mathbf{bqs}}$ can be divided into two phases: the oblivious key distribution phase (we also call it a *precomputation* phase) and the transfer phase. It is interesting to note that both protocols follow the same steps in the transfer phase. We present the transfer phase of both protocols in Figure 4.3. We slightly rewrite the protocol by using only one hash function (H describes a random oracle) instead of two random hash functions f_0 and f_1. This is done for comparison purposes and because, in practice, H is implemented as a specific hash function, such as SHA.

$\Pi^{\mathbf{BBCS}}$ protocol

Alice's input: $(m_0, m_1) \in \{0,1\}^l$ (two messages).
Bob's input: $b \in \{0,1\}$ (bit choice).

Precomputation phase: Alice and Bob generate an oblivious key $(\mathsf{ok}^{\mathsf{A}}, (\mathsf{ok}^{\mathsf{B}}, \mathsf{e}^{\mathsf{B}}))$ according to the corresponding procedure. $\Pi^{\mathbf{BBCS}}_{\mathcal{F}_{\mathbf{COM}}}$ as in Figure 3.2 and $\Pi^{\mathbf{BBCS}}_{\mathbf{bqs}}$ as in Figure 3.3.

Transfer phase:

5. Bob defines $I_0 = \{i : \mathsf{e}^{\mathsf{B}}_i = 0\}$ and $I_1 = \{i : \mathsf{e}^{\mathsf{B}}_i = 1\}$ and sends the pair $(I_b, I_{b\oplus 1})$ to Alice.

6. Alice computes the pair of strings (s_0, s_1) as $s_i = m_i \oplus H(\mathsf{ok}^{\mathsf{A}}_{I_{b\oplus i}})$ and sends to Bob.

7. Bob computes $m_b = s_b \oplus H(\mathsf{ok}^{\mathsf{B}}_{I_0})$.

Alice's output: $\perp$.
Bob's output: m_b.

Figure 4.3: Transfer phase of BBCS-based QOT protocols in the $\mathcal{F}_{\mathbf{COM}}-$hybrid model and bounded-quantum-storage model.

Now, observe that Bob sends two sets $(I_b, I_{b\oplus 1})$ to Alice during the first communication round (Figure 4.3, Step 5). This can be optimized as it is redundant to send both sets of indexes. In fact, with only one set (I_b), Alice is able to know its complement $(\overline{I_b} = I_{b\oplus 1})$). Thus, we end up with the optimized protocol $(\Pi^{\mathbf{BBCS}}_{\mathbf{O}})$ presented in Figure 4.4. This optimized version requires less bandwidth when compared with the initially proposed transfer phase. Now, note that the size of the sets can be identified with a symmetric security parameter κ, as the sets define the keys $(\mathsf{ok}_{I_i},\ i = 0, 1)$ to be used in the hash scheme H. For comparison purposes, we consider that $\kappa = 128$. Furthermore, we can consider the messages, m_0 and m_1, to be garbled circuit's keys. As their size can be $l = 128, 192$ or 256, we consider that $l \sim \kappa$ have the same order of magnitude, meaning they represent the same cost of bits. Therefore, Bob only needs to send l bits to Alice in step 5, leading to an overall reduction of one fourth in the number of bits sent during the transfer phase.

In order to fairly compare the transfer phase of $\Pi^{\mathbf{BBCS}}_{\mathbf{O}}$ protocol with the corresponding

$\Pi_{\mathbf{O}}^{\mathbf{BBCS}}$ protocol

Alice's input: $(m_0, m_1) \in \{0,1\}^l$ (two messages).
Bob's input: $b \in \{0,1\}$ (bit choice).

Precomputation phase: Alice and Bob generate an oblivious key $(\mathsf{ok}^{\mathsf{A}}, (\mathsf{ok}^{\mathsf{B}}, \mathsf{e}^{\mathsf{B}}))$ according to the corresponding procedure. $\Pi_{\mathcal{F}_{\mathbf{COM}}}^{\mathbf{BBCS}}$ as in Figure 3.2 and $\Pi_{\mathbf{bqs}}^{\mathbf{BBCS}}$ as in Figure 3.3.

Transfer phase:

5. Bob defines $I_0 = \{i : \mathsf{e}_i^{\mathsf{B}} = 0\}$ and $I_1 = \{i : \mathsf{e}_i^{\mathsf{B}} = 1\}$ and **sends only** I_b to Alice.
6. Alice computes the pair of strings (s_0, s_1) as $s_i = m_i \oplus H(\mathsf{ok}^{\mathsf{A}}_{I_{b\oplus i}})$ and sends to Bob.
7. Bob computes $m_b = s_b \oplus H(\mathsf{ok}^{\mathsf{B}}_{I_0})$.

Alice's output: $\perp$.
Bob's output: m_b.

Figure 4.4: Transfer phase of BBCS-based QOT protocols in the $\mathcal{F}_{\mathbf{COM}}-$hybrid model and bounded-quantum-storage model.

phase of other classical protocols, we have to divide the classical protocols in these two phases. We apply the following rule: all the steps used in the protocol that are independent of the messages (m_0 and m_1) and of the bit choice (b) are considered to be part of the precomputation phase, otherwise they are included in the transfer phase. Since the precomputation phase can be executed before the execution of the Yao GC protocol, it is more important to guarantee that the transfer phase has small complexity. Furthermore, we stress that here we will only compare the complexity among the different protocols' transfer phase because their precomputation phase rely on different technologies. Since quantum technologies are still in their infancy and constantly evolving, it is difficult to compare the efficiency with classical approaches. Nevertheless, it is worth noting that the oblivious key phase of $\Pi_{\mathbf{O}}^{\mathbf{BBCS}}$ protocol is linear in all its security parameters. In fact, as presented by Lemus et al. [134], the time complexity of $\Pi_{\mathcal{F}_{\mathbf{COM}}}^{\mathbf{BBCS}}$ is of the order $\mathcal{O}(\kappa(2l+t))$, where κ is the security parameter of the hash-based commitments, $2l$ is the number of qubits sent used to directly generate the oblivious keys and t is the number of testing qubits.

4.2.2 Classical OT

In section 4.1, we divided the well known Bellare-Micali protocol in these two phases and we observed that it uses three exponentiations during the transfer phase. In Table 4.2 we present the number of required modular exponentiations and communication rounds during the transfer phase of four well known classical protocols that have their security based on the computational hardness of the Discrete Logarithm problem.

Protocol	Exponentiation	Comm. rounds
EGL [85]	3	2
BM [2]	3	2
NP [86]	2	2
SimpleOT [87]	1	2

Table 4.2: Number of modular exponentiation operations and communication rounds executed during the transfer phase of four other classical protocols.

From Table 4.2, we see that the most efficient protocol (SimpleOT [87]) still requires one exponentiation operation and 2 communication rounds. From the above formula (4.1) and setting $C_{mcyles} = 2.5 \times 10^9$, $C_{mexp} = 400\,000$ and $N_{mexp} = 1$, we get an overestimation of around 6000 OT per second. Comparing with the rate achieved by OT extension protocols (10 million OT in 2.62 s), it is still very inefficient.

This means that the current classical OT protocols have a computational complexity limited by $\mathcal{O}(n^{2.58})$ bit operations due to modular exponentiation. The $\Pi_{\mathbf{O}}^{\mathbf{BBCS}}$ protocol only depends on simple bit operations (XOR, truncation and comparison), meaning its computational complexity is linear in the length of the messages $\mathcal{O}(n)$.

In addition, it is important to stress that none of the above protocols are secure against quantum computer attacks. In order to have classical OT protocols with this level of security, we need to follow post-quantum approaches which may lead to more demanding operations [196]. As reported in [197], using Kyber key encapsulation based on the module learning with errors (M-LWE) problem [198], it takes 24 ms to generate one OT in a LAN network. This leads to a rate of just 41 OT per second, which is even lower than the rate reported by [3] for the Naor-Pinkas [86]: 56 OT per second. In [199, 200], the authors present a 1-out-of-n OT based on the NTRU post-quantum encryption system [201] and compare it with the SimpleOT [87] version. In this case, although Bob and Alice sides are more efficient individually, the overall NTRU OT protocol is still less efficient. For the highest security level, it takes around 1.372 ms to generate one OT with the post-quantum approach, whereas it takes 0.727 ms using the original SimpleOT protocol. These timings lead to the rates of 728 and 1375 OT per second, respectively. It is important to note that these protocols are still prone to *intercept now - decipher later* attacks since they are based on computational assumptions that are only *believed* (and not proved) to be secure against quantum computer attacks.

4.2.3 OT extension

As we explained in section 4.1.3, several techniques based on an hybrid symmetric-asymmetric approach were developed as a way to increase the OT execution rate. These techniques use a small number κ (= 128) of base OT protocols (e.g. EGL, BM, NP, SimpleOT) and extend this resource to m (= 10 000 000) OT executions, where $m >> \kappa$.

Again, to compare OT extension protocols with $\Pi_{\mathbf{O}}^{\mathbf{BBCS}}$, we have to decompose them into the same two phases: precomputation phase and transfer phase. In this section, we make a bit-wise comparison of the communication and computational complexity of m executions of $\Pi_{\mathbf{O}}^{\mathbf{BBCS}}$ and one OT extension execution because OT extension protocols generate a predetermined number (m) of OTs at once. First, we compare $\Pi_{\mathbf{O}}^{\mathbf{BBCS}}$ with a semi-honest protocol which we call for short ALSZ13, and then, we compare the quantum version with a maliciously secure protocol denoted by KOS15.

ALSZ13 comparison

Let us consider the OT extension protocol (ALSZ13) proposed in [3]. It is shown in Figure 4.5. At the time of writing, it reports the fastest implementation: 10 million OT in 2.68 seconds. This protocol is originally divided into two phases: initial OT phase and OT extension phase. Note that these two phases correspond exactly to our division of precomputation and transfer phases. Thus, for comparative purposes, we are only interested in the second phase.

In Tables 4.3 and 4.4, we show the computational and communication complexity of both protocols, respectively. In Table 4.3, PRG stands for pseudorandom generator, κ represents the number of base OTs executed in the OT extension precomputation phase, m is the number of final OTs and l is the length of the OT strings. We consider that $l \sim \kappa$ have the same order of magnitude, meaning they represent the same cost of bits. This is so, because $\kappa = 128$ in [3] and the key length used in the garbled circuits are $l = 128, 192$ or 256. Now, we justify the analysis presented in Tables 4.3 and 4.4.

Regarding the ALSZ13 protocol, for every $i \in [\kappa]$, Bob computes two PRGs in step 4 and Alice computes one PRG in step 5. This accounts for 3κ PRG executions. For every $j \in [m]$, Alice computes two hash functions in step 6 and Bob computes one hash

ALSZ13 OT extensions protocol [3]

Alice's input: m pairs (x_j^0, x_j^1), $\forall j \in [m]$ of l-bit strings.
Bob's input: m selection bits $\boldsymbol{r} = (r_1, ..., r_m)$.

Initial OT phase (Precomputation phase)

1. Alice randomly generates a string $\boldsymbol{s} = (s_1, ..., s_\kappa)$.
2. Bob randomly chooses κ pairs of κ-bit strings $\{(\boldsymbol{k}_i^0, \boldsymbol{k}_i^1)\}_{i=1}^{\kappa}$.
3. Bob and Alice execute κ base OTs, where Alice plays the role of the receiver with input $\boldsymbol{k}$ and Bob plays the role of the sender with messages $(\boldsymbol{k}_i^0, \boldsymbol{k}_i^1)$, $\forall i \in [\kappa]$.

OT extension phase (Transfer phase)

4. Bob applies a pseudorandom number generator G to $\boldsymbol{k}_i^0$, i.e. $\boldsymbol{t}^i = G(\boldsymbol{k}_i^0)$. Computes $\boldsymbol{u}^i = \boldsymbol{t}^i \oplus G(\boldsymbol{k}_i^1) \oplus \boldsymbol{r}$ and sends $\boldsymbol{u}^i$ to Alice for every $i \in [\kappa]$.
5. Alice computes $\boldsymbol{q}^i = (s_i \cdot \boldsymbol{u}^i) \oplus G(\boldsymbol{k}_i^{s_i})$.
6. Alice sends (y_j^0, y_j^1) for every $j \in [m]$, where $y_j^0 = x_j^0 \oplus H(j, \boldsymbol{q}_j)$, $y_j^1 = x_j^1 \oplus H(j, \boldsymbol{q}_j \oplus \boldsymbol{s})$ and $\boldsymbol{q}_j$ is the j-th row of the matrix $Q = [\boldsymbol{q}^1|...|\boldsymbol{q}^\kappa]$. Note that, in practice, it is required to transpose Q to access its j-th row.
7. Bob computes $x_j^{r_j} = y_j^{r_j} \oplus H(j, \boldsymbol{t}_j)$.

Alice's output: $\perp$.
Bob's output: $(x_1^{r_1}, ..., x_m^{r_m})$.

Figure 4.5: Precomputation and transfer phases of OT extensions protocol presented in [3].

function. This accounts for $3m$ hash functions. For every $i \in [\kappa]$, Bob computes two $m-$bit XOR operations in step 4 and Alice computes one $m-$bit XOR operation. For every $j \in [m]$, Alice computes two $l-$bit XOR operations in step 6 and Bob computes one $l-$bit XOR operation in step 7. Also, for every $j \in [m]$, Alice computes one $\kappa-$bit XOR operation in step 6. This accounts for $3m\kappa + 3ml + m\kappa$ bitwise XOR operations. For every $i \in [\kappa]$, Alice computes one $m-$bit AND operation in step 5. Finally, Alice has to perform a matrix inversion which accounts for around $m \log m$ bit operations. The communication complexity is given by the following elements: Bob sends an $m-$bit vector for every $i \in [\kappa]$ and Alice sends two $l-$bit messages for every $j \in [m]$. This accounts for $2ml + m\kappa$ bits sent.

Regarding the $\Pi_{\mathbf{O}}^{\mathbf{BBCS}}$ protocol, for every execution of the protocol, Alice computes two hash functions in step 6 and Bob computes one hash function in step 7. This accounts for $3m$ hash functions. Also, Alice computes two $l-$bit XOR operations in step 6 and Bob computes one $l-$bit XOR operation in step 7. This accounts for $3ml$ bitwise XOR operations. For every execution of the protocol, Alice performs 2κ bitwise comparisons in step 5. Also, Alice computes two $\kappa-$bit truncation in step 6 and Bob computes one $\kappa-$bit truncation in step 7. The communication complexity is given by the following elements: Bob sends a $\kappa-$bit vector and Alice sends two $l-$bit messages, for every execution of the protocol. This accounts for $2ml + m\kappa$ bits sent.

Operation	ALSZ13	$\Pi_{\mathbf{O}}^{\mathbf{BBCS}}$
PRG (AES)	3κ	-
Hash (SHA-1)	$3m$	$3m$
Bitwise XOR	$3m\kappa + 3ml + m\kappa$	$3ml$
Bitwise AND	$m\kappa$	-
Matrix transposition	$m \log m$	-
Bitwise comparison	-	$2m\kappa$
Bitwise truncation	-	$3m\kappa$

Table 4.3: Computational complexity comparison between ALSZ13 [3] OT extension protocol and $\Pi_{\mathbf{O}}^{\mathbf{BBCS}}$ protocol from section 4.2.1.

	ALSZ13	$\Pi_{\mathbf{O}}^{\mathbf{BBCS}}$
Bits sent	$2ml + m\kappa$	$2ml + m\kappa$

Table 4.4: Communication complexity comparison between ALSZ13 [3] OT extension protocol and $\Pi_{\mathbf{O}}^{\mathbf{BBCS}}$ protocol from section 4.2.1.

We have that the communication complexity is exactly the same in both protocols: $\sim 3ml$. So, the OT extension does not have any advantage over $\Pi_{\mathbf{O}}^{\mathbf{BBCS}}$ during the communication phase. Regarding their computational complexity, we have to compare binary operations executed betwen each protocol.

Firstly, we can see that $\Pi_{\mathbf{O}}^{\mathbf{BBCS}}$ transfer phase is asymptotically more efficient than ALSZ13 OT extension transfer phase. The computational complexity of OT extension is not linear in the number of OT executions, $\mathcal{O}(m \log m)$, whereas it is linear in the case of $\Pi_{\mathbf{O}}^{\mathbf{BBCS}}$, $\mathcal{O}(m)$. Now, let us compare the binary operations between each protocol. Denote by $B_{\text{op}}^{\text{ALSZ13}}$ and $B_{\text{op}}^{\text{BBCS}}$ the number of binary operations executed by ALSZ13 and $\Pi_{\mathbf{O}}^{\mathbf{BBCS}}$, respectively. As both protocols execute $3m$ hash functions,we do not take into account their execution. Also, assuming that $\kappa \sim l$, $B_{\text{op}}^{\text{ALSZ13}}$ is roughly given by,

$$\begin{aligned} B_{\text{op}}^{\text{ALSZ13}} &= 3\kappa + 3m\kappa + 3ml + m\kappa + m\kappa + m\log m \\ &= 8m\kappa + 3\kappa + m\log m \end{aligned}$$

and $B_{\text{op}}^{\text{BBCS}} = 8m\kappa$. Here, we simplify and assume that 3κ PRGs executions consume only 3κ bit operations. Therefore, ALSZ13 has more $B_{\text{op}}^{\text{ALSZ13}} - B_{\text{op}}^{\text{BBCS}} \geq m\log m$ binary operations than the transfer phase of $\Pi_{\mathbf{O}}^{\mathbf{BBCS}}$ protocol.

From this, we conclude that $\Pi_{\mathbf{O}}^{\mathbf{BBCS}}$ transfer phase competes with the ALSZ13 corresponding phase and has the potential to be more efficient. It is important to stress that $\Pi_{\mathbf{O}}^{\mathbf{BBCS}}$ efficiency performance of the transfer phase comes along with a drastic increase in the security of the protocol. While ALSZ13 protocol relies on the computational assumptions of the base OT, $\Pi_{\mathbf{O}}^{\mathbf{BBCS}}$ protocol is proved to be secure against quantum computers. Moreover, while ALSZ13 is a semi-honest protocol (assumes well-behaved parties that follow the protocol), $\Pi_{\mathbf{O}}^{\mathbf{BBCS}}$ protocol is secure against any corrupted party. Indeed, in order to get a fair comparison, we should consider OT extension protocols that are secure against malicious parties. The work developed in [192] presented the first protocol in the malicious scenario, which was latter optimized by KOS15 [4] and ALSZ15 [195]. Both optimizations carry out one run of the semi-honest OT extension presented in ALSZ13 plus some consistency checks. The protocol presented in [4] adds to ALSZ13 a *check correlation* phase after the transfer phase and the protocol presented in [195] adds a *consistency check* phase during the transfer phase. This means that both malicious protocols' transfer phases have greater computational and communication complexity when compared with ALSZ13. Therefore, we can easily deduce that $\Pi_{\mathbf{O}}^{\mathbf{BBCS}}$ transfer phase has less computational and communication complexity than its classical equivalents with respect to the adversary model used. Next, we compare the KOS15 protocol [4] and $\Pi_{\mathbf{O}}^{\mathbf{BBCS}}$ protocol.

KOS15 comparison

KOS15 protocol is very similar to ALSZ13 with the addition of a *check correlation* phase. This phase ensures that the receiver is well behaved and does not cheat. The KOS15 protocol that generates m l-bit string OT out of κ base OT with computational security given by κ and statistical security given by w is shown in Figure 4.6. Note that, in Figure 4.6, we join all the subprotocols presented in the original paper: $\prod_{\text{COTE}}^{\kappa,m'}$, $\prod_{\text{ROT}}^{\kappa,m}$ and $\prod_{\text{DEROT}}^{\kappa,m}$. Also, they identify $\mathbb{Z}_2^\kappa$ with the finite field $\mathbb{Z}_{2^\kappa}$ and use "$\cdot$" for multiplication in $\mathbb{Z}_{2^\kappa}$. For example, the element $\boldsymbol{t}_j$ in $\sum_{j=1}^{m'} \boldsymbol{t}_j \cdot \chi_j$ (Figure 4.6, step 10) should be considered in $\mathbb{Z}_{2^\kappa}$.

Similarly to the $\Pi_{\mathbf{O}}^{\mathbf{BBCS}}$ and ALSZ13 protocols, KOS15 starts with a precomputation

KOS15 OT extensions protocol [4]

Alice's input: m pairs (x_j^0, x_j^1), $\forall j \in [m]$ of l-bit strings.
Bob's input: m selection bits $\boldsymbol{r} = (r_1, ..., r_m)$.

Initial OT phase (Precomputation phase)

1. Alice randomly generates a string $\boldsymbol{s} = (s_1, ..., s_\kappa)$ and Bob randomly chooses κ pairs of κ-bit strings $\{(\boldsymbol{k}_i^0, \boldsymbol{k}_i^1)\}_{i=1}^{\kappa}$.
2. Bob and Alice execute κ base OTs. Alice plays the role of the receiver with input $\boldsymbol{s}$ and Bob plays the role of the sender with messages $(\boldsymbol{k}_i^0, \boldsymbol{k}_i^1)$, $i \in [\kappa]$.
3. Bob applies a pseudorandom number generator G to $\boldsymbol{k}_i^0$ and $\boldsymbol{k}_i^1$: $\boldsymbol{t}^i = G(\boldsymbol{k}_i^0)$ and $\boldsymbol{t}_1^i = G(\boldsymbol{k}_i^1)$. Also, set $\boldsymbol{T}^i = \boldsymbol{t}^i \oplus \boldsymbol{t}_1^i$.
4. Alice applies G to $\boldsymbol{k}_i^{s_i}$ and sets $\boldsymbol{g}_i^{s_i} = G(\boldsymbol{k}_i^{s_i})$.

OT extension phase (Transfer phase)
Extend

5. Bob generates random elements r_j, for $r \in [m+1, m']$ and resize $\boldsymbol{r} = (r_1, ..., r_m, r_{m+1}, ..., r_{m'})$, where $m' = m + (\kappa + w)$.
6. Bob computes $\boldsymbol{u}^i = \boldsymbol{T}^i \oplus \boldsymbol{r}$ and sends $\boldsymbol{u}^i$ to Alice for every $i \in [\kappa]$.
7. Alice computes $\boldsymbol{q}^i = (s_i \times \boldsymbol{u}^i) \oplus \boldsymbol{g}_i^{s_i}$ for every $i \in [\kappa]$.

Check correlation

8. Sample $(\chi_1, ..., \chi_{m'}) \leftarrow \mathcal{F}_{\mathrm{Rand}}(\mathbb{F}_{2^\kappa}^{m'})$.
9. Bob computes $x = \sum_{j=1}^{m'} r_j \cdot \chi_j$ and $t = \sum_{j=1}^{m'} \boldsymbol{t}_j \cdot \chi_j$, where $\boldsymbol{t}_j$ is the j-th row of the matrix $[\boldsymbol{t}^1|...|\boldsymbol{t}^\kappa]$ and sends these to Alice.
10. Alice computes $q = \sum_{j=1}^{m'} \boldsymbol{q}_j \cdot \chi_j$, where $\boldsymbol{q}_j$ is the j-th row of the matrix $Q = [\boldsymbol{q}^1|...|\boldsymbol{q}^\kappa]$, and checks that $t = q + r \cdot \boldsymbol{s}$. If the check fails, output Abort, otherwise continue.

Randomize and encrypt

11. Alice sends (y_j^0, y_j^1) for every $j \in [m]$, where $y_j^0 = x_j^0 \oplus H(j, \boldsymbol{q}_j)$, $y_j^1 = x_j^1 \oplus H(j, \boldsymbol{q}_j \oplus \boldsymbol{s})$.
12. Bob computes $x_j^{r_j} = y_j^{r_j} \oplus H(j, \boldsymbol{t}_j)$.

Alice's output: $\perp$.
Bob's output: $(x_1^{r_1}, ..., x_m^{r_m})$.

Figure 4.6: Precomputation and transfer phases of OT extensions protocol presented in [4].

Operation	KOS15	$\Pi_{\mathbf{O}}^{\mathbf{BBCS}}$
Hash (SHA-1)	$3m$	$3m$
Bitwise XOR	$3m\kappa + 3ml + m\kappa$	$3ml$
Bitwise AND	$m\kappa$	-
Matrix transposition	$m \log m$	-
Bitwise comparison	-	$2ml$
Bitwise truncation	-	$3ml$
κ-bit additon	$3(m + (\kappa + w))\kappa$	-
κ-bit mult	$2(m + (\kappa + w))\kappa^{1.58}$	-

Table 4.5: Computational complexity comparison between KOS15 [4] OT extension protocol and $\Pi_{\mathbf{O}}^{\mathbf{BBCS}}$ protocol from section 4.2.1.

	KOS15	$\Pi_{\mathbf{O}}^{\mathbf{BBCS}}$
Bits sent	$2ml + m\kappa + \kappa$	$2ml + m\kappa$

Table 4.6: Communication complexity comparison between KOS15 [4] OT extension protocol and $\Pi_{\mathbf{O}}^{\mathbf{BBCS}}$ protocol from section 4.2.1.

phase that can be carried out before the actual computation of the OT protocols. However, it is interesting to note that in the original KOS15 paper [4] the computation of the PRGs G is carried out in the OT extension phase. In fact, these 3κ G computations can be executed during the precomputation phase because they do not depend on the input elements. As mentioned before, the additional steps that KOS15 added to the ALSZ13 protocol are steps $9 - 11$ (check correlation phase). Here, both parties start by calling a random oracle functionality $\mathcal{F}_{\text{Rand}}(\mathbb{F}_{2^\kappa}^{m'})$ that provides them with equal random values. Bob has to compute twice m' κ-bit sums, m' κ-bit multiplication and sends 2κ bit (x and t) to Alice. Finally, Alice has to compute m' κ-bit sums and m' κ-bit multiplications. We consider karatsuba method for multiplication with complexity $O(\kappa^{1.585})$ and schoolbook addition with complexity $O(\kappa)$. For simplicity, we consider that the sum of two κ takes κ bit operations and the multiplication takes $\kappa^{1.585}$.

Similarly to what with did with ALSZ13 protocol, let us compare the binary operations between each KOS15 and $\Pi_{\mathbf{O}}^{\mathbf{BBCS}}$. Denote by $B_{\text{op}}^{\text{KOS15}}$ and $B_{\text{op}}^{\text{BBCS}}$ the number of binary operations executed by KOS15 and $\Pi_{\mathbf{O}}^{\mathbf{BBCS}}$, respectively. Again, without taking into account the execution of $3m$ hash functions and assuming that $\kappa \sim l$, $B_{\text{op}}^{\text{KOS15}}$ is roughly given by,

$$
\begin{aligned}
B_{\mathrm{op}}^{\mathrm{KOS15}} &= 3m\kappa + 3ml + m\kappa \\
&+ m\kappa + m\log m \\
&+ 3(m + (\kappa + w))\kappa \\
&+ 2(m + (\kappa + w))\kappa^{1.58} \\
&= 11m\kappa + m\log m \\
&+ 3\kappa^2 + 3w\kappa \\
&+ 2m\kappa^{1.58} + 2\kappa^{2.58} + 2w\kappa^{1.58}
\end{aligned}
$$

and $B_{\mathrm{op}}^{\mathrm{HQOT}} = 8m\kappa$. Therefore, KOS15 has more $B_{\mathrm{op}}^{\mathrm{KOS15}} - B_{\mathrm{op}}^{\mathrm{HQOT}} \geq 5m\kappa + m\log m$ binary operations than $\Pi_{\mathbf{O}}^{\mathbf{BBCS}}$ transfer phase. For this estimation, note that we are considering the lower bound $2m\kappa$ instead of $2m\kappa^{1.58}$ and we are not taking into account the implementation of the random oracle $\mathcal{F}_{\mathrm{Rand}}(\mathbb{F}_{2^\kappa}^{m'})$, which would add an extra cost linear in the number of OT executions.

Regarding the communication complexity, the number of bits sent during both KOS15 and $\Pi_{\mathbf{O}}^{\mathbf{BBCS}}$ is almost the same. KOS15 only adds κ bits to the communication in KOS15 during the check correlation phase. However, since this overhead is independent of m (number of OTs executed) its effect is amortized for big m.

4.3 Conclusion

The importance of secure and efficient implementations of OT is crucial for several secure computations. In particular, Yao garbled circuit protocol's security and efficiency is deeply connected with OT's security and efficiency. However, Impagliazzo and Rudich result enforces classical OT protocols to be founded on asymmetric cryptographic primitives, that are known to be insecure against quantum attacks or their security is based on some conjectures. On the other hand, several works [31, 32, 134, 139] used the laws of physics to prove the BBCS-based QOT protocols to be secure against malicious adversaries with access to a quantum computer. Moreover, we have that oblivious keys allow to separate the quantum technological burden from the execution of the OT and can be used to efficiently implement OT.

In this chapter, we compared an optimized version ($\Pi_{\mathbf{O}}^{\mathbf{BBCS}}$) of the BBCS-based QOT protocols' transfer phase with the transfer phase of the protocol that, to the best of our knowledge, is the fastest implementation of OT[3]. We concluded that $\Pi_{\mathbf{O}}^{\mathbf{BBCS}}$ transfer phase has the potential to be faster than ALSZ13 OT extension transfer phase while preserving a much higher security. In fact, the ALSZ13 protocol is only proved to be secure in the semi-honest model while $\Pi_{\mathbf{O}}^{\mathbf{BBCS}}$ is secure in the malicious setting. Also, we concluded that the transfer phase of current maliciously secure implementations [4, 195] have greater computation and communication complexity than the $\Pi_{\mathbf{O}}^{\mathbf{BBCS}}$ transfer phase.

Chapter 5

Quantum oblivious linear evaluation

The topic of this Chapter are the results of [AM16], that have been obtained in collaboration with Gorjan Alagic, and large parts of this article are used here unaltered.

This introduction has already been used in the introduction of the thesis.

Oblivious Linear Evaluation (OLE) is a cryptographic task that permits two distrustful parties, say Alice and Bob, to jointly compute the output of a linear function $f(x) = ax+b$ in some finite field, $\mathbb{F}$. Alice provides inputs $a, b \in \mathbb{F}$ and Bob provides $x \in \mathbb{F}$, while the output, $f(x)$, becomes available only to Bob. As the parties are distrustful, a secure OLE protocol should not permit Alice to learn anything about Bob's input, while also Alice's inputs should remain unknown to Bob. OLE can be seen as a generalization of Oblivious Transfer (OT) [48], a basic primitive for secure two-party computation, which is a special case of secure multi-party computation [46**? ?**]. OT has been shown to be complete for secure multi-party computation [**?**], i.e., any such task, including OLE, can be achieved given an OT implementation.

Impagliazzo and Rudich proved that OT protocols require public-key cryptography and cannot just rely on symmetric cryptography [57]. Consequently, OLE cannot rely on symmetric cryptography either, and we need to resort to public-key cryptography. However, Shor's quantum algorithm [**?**] poses a threat to the currently deployed public-key systems, motivating the search for protocols secure against quantum attacks. Bennet et al. [30] and Crépeau [**?**] proposed the first protocols for quantum OT (QOT). As far as quantum OLE (QOLE) is concerned, to the best of our knowledge, no protocol has been proposed as of now. Analogously to the classical case, it is expected that one can implement QOLE based on QOT protocols. That said, in this work we propose a protocol for QOLE that, additionally, does not rely on any QOT implementation.

OLE is commonly generalized to vector OLE (VOLE). In this setting, Alice defines a set of k linear functions $(\boldsymbol{a}, \boldsymbol{b}) \in \mathbb{F}^k \times \mathbb{F}^k$ and Bob receives the evaluation of all these functions on a specified element $x \in \mathbb{F}$, i.e. $\boldsymbol{f} := \boldsymbol{a}x + \boldsymbol{b}$. One can think of VOLE as the arithmetic analog of string OT and show how it can be used in certain Secure Arithmetic Computation and Non-Interactive Zero Knowledge proofs [65]. Ghosh et. al put further in evidence the usefulness of VOLE by showing that it serves as the building block of Oblivious Polynomial Evaluation [60], a primitive which allows more sophisticated applications, such as password authentication, secure list intersection, anonymous complaint boxes [**?**], anonymous initialization for secure metering of client visits in servers [**?**], secure Taylor approximation of relevant functions (e.g. logarithm) [**?**], secure set intersection [**?**] and distributed generation of RSA keys [**?**]. We also show how our QOLE protocol can be adapted to achieve secure VOLE.

5.1 Contributions overview

We present a quantum protocol for OLE with universally composable security (quantum-UC security, see Definition 6) in the $\mathcal{F}_{\mathbf{COM}}-$hybrid model, i.e. when assuming the existence of a commitment functionality, $\mathcal{F}_{\mathbf{COM}}$ (see Figure 2.5). To obtain a secure protocol, we take advantage of the properties of mutually unbiased bases (MUBs) in high-dimensional Hilbert spaces with prime and prime-power dimension. Such a choice is motivated by recent theoretical and experimental advances that pave the way for the development and realization of new solutions for quantum cryptography [34**? ? ? ? ? ?** –38]. To the best of our knowledge, our protocol is the first proposal of a QOLE protocol proved to be quantum-UC secure. Moreover, it is not based on any QOT implementation which would be the standard approach. To prove its security, the only assumption we make is the existence of a commitment functionality. We consider the static corruption adversarial model with both semi-honest and dishonest adversaries. Finally, we modify the proposed protocol to generate quantum-UC secure VOLE.

Main tool. The proposed protocol $\pi_{\mathbf{QOLE}}$ (see Figure 5.5) is based on the fact that in a Hilbert space of dimension d (isomorphic to $\mathbb{Z}_d$) there exists a set of MUBs $\{|e_r^x\rangle\}_{x,r\in\mathbb{Z}_d}$, such that, upon the action of a certain operator V_a^b, each basis element r is shifted by some linear factor $ax-b$ inside the same basis x:

$$V_a^b\,|e_r^x\rangle = c_{a,b,x,r}\,\left|e_{ax-b+r}^x\right\rangle, \tag{5.1}$$

where $a,b,x,r\in\mathbb{Z}_d=\{0,1,\ldots,d-1\}$. If Alice controls the operator V_a^b and Bob controls the quantum state $|e_r^x\rangle$, they are able to compute a linear function $f(x)=ax-b$ where effectively Alice controls the function $f=(a,b)$ and Bob controls its input x. Moreover, since Bob controls x and r, he can receive $f(x)$ by measuring the output element.

Protocol overview. In a nutshell, the QOLE protocol (see Figure 5.5) with inputs $f = (a, b)$ from Alice and x from Bob is divided into two main phases. In the first *quantum phase*, Alice and Bob use high-dimensional quantum states to generate n random weak OLE (RWOLE) instances, where n is the security parameter. In this phase, Alice outputs n random elements $f_i^0 = (a_i^0, b_i^0)$, and Bob outputs n elements $(x_i^0, y^0 = f_i^0(x_i^0))$. These instances are considered to be weaker because Bob is allowed to have some amount of information about the n outputs of Alice (a_i^0, b_i^0). In the second *post-processing phase*, Alice and Bob use classical tools to extract one secure OLE from the aforementioned n instances.

More specifically, in the quantum phase, Bob randomly generates $m = (1+t)n$ quantum states $\left|e_{r_i}^{x_i^0}\right\rangle$ and sends them to Alice. Then, Bob commits to his choice (x_i^0, r_i), $\forall i \in [m]$, where for any $l \in \mathbb{N}$, $[l]$ denotes the set $\{1, \dots, l\}$, using an ideal commitment functionality, $\mathcal{F}_{\mathbf{COM}}$, and Alice asks to verify a subset T of size tn of these commitments. This intermediate *commit-and-open* step allows Alice to test Bob's behaviour and ensure that he does not deviate *too much* from the protocol, and it is a common method used in security proofs of QOT protocols [29, 80]. If Bob passes all the tests, Alice randomly generates (a_i^0, b_i^0) and applies $V_{a_i^0}^{b_i^0}$ to the remaining n received states $\left|e_{r_i}^{x_i^0}\right\rangle$, for $i \in [m] \setminus T$. For the rest of this section we relabel and denote $[n] = [m] \setminus T$. According to the expression (5.1), the output states are given by $\left|e_{a_i^0 x_i^0 - b_i^0 + r_i}^{x_i^0}\right\rangle$ and she sends them to Bob, who outputs $y_i^0 = a_i^0 x_i^0 - b_i^0$ by measuring the received states in the corresponding basis x_i^0 and subtracting r_i, $\forall i \in [n]$.

The post-processing phase uses two subprotocols: a derandomization step (see Figure 5.3) and an extraction step (see Figure 5.4). The derandomization step is based on the protocol π_{OLE}^n from [**?**] and transforms the n RWOLE instances into n weak OLE (WOLE) instances with inputs $(a_i, b_i)_{i \in [n]}$ chosen by Alice and inputs x_i for $i \in [n]$ chosen by Bob. The extraction protocol uses the so-called *Multi-linear Modular Hashing* family, MMH*, of two-universal hash functions [75] to render Bob's information on Alice's system useless and to extract one secure OLE out of n instances of WOLE. In the extraction phase, Alice samples a two-universal hash function $g_{\boldsymbol{\kappa}}$ from MMH* and sends it to Bob. Then, with adequately-crafted vectors $(\boldsymbol{a}, \boldsymbol{b}) = \big((a_1, \dots, a_n), (b_1, \dots, b_n)\big)$, Alice has $a = g_{\boldsymbol{\kappa}}(\boldsymbol{a})$ and $b = g_{\boldsymbol{\kappa}}(\boldsymbol{b})$, and Bob outputs $y = g_{\boldsymbol{\kappa}}(\boldsymbol{y})$, where $\boldsymbol{y} = \boldsymbol{a}\boldsymbol{x} + \boldsymbol{b}$ after point-wise vector multiplication with the constant vector $\boldsymbol{x} = (x, \dots, x)$.

quantum-UC security. Due to the quantum nature of the states $\left|e_{r_i}^{x_i^0}\right\rangle_{i \in [n]}$, a dishonest Alice is not able to distinguish which bases $x_i^0, i \in [n]$ are used by Bob. From her point of view, Bob's states are maximally mixed and therefore completely hide x_i^0. This is enough to ensure that, in the derandomization step, Alice does not receive any information about

Bob's final input x. For a dishonest Bob, to correctly pass all Alice's tests, it means he did not cheat at all rounds with overwhelming probability. This ensures that he has some *bounded* information on Alice's random elements $(a_i^0, b_i^0)_{i\in[n]}$, and using privacy amplification techniques in the extraction step, Alice can guarantee that Bob's information about her final input (a, b) is the same as in the case of an ideal OLE functionality, i.e. the probability distribution of a is close to uniform.

Turning this intuition into a quantum-UC security proof requires some additional insights. First, we need a way to quantify Bob's information on Alice's elements (a_i^0, b_i^0) after the testing phase and the application of the corresponding $V_{a_i^0}^{b_i^0}$ operators, for $i \in [n]$; for this purpose we use the quantum *min-entropy* (see Definition **??**). We follow the approach of [29] to guarantee that Bob does not significantly deviate from the protocol in all the rounds, and we use Theorem 1 from [73] to compute a concrete lower bound of Bob's min-entropy on Alice elements $(a_i^0, b_i^0)_{i\in[n]}$. Along with Lemma 9, we have that $a = g_{\boldsymbol{\kappa}}(\boldsymbol{a})$ is close to uniform, which is sufficient to prove that Bob does not know more about (a, b) than what the output $y = ax + b$ reveals.

In order to show that the protocol $\pi_{\mathbf{QOLE}}$ is quantum-UC secure, we need to show that an ideal execution of $\pi_{\mathbf{QOLE}}$ with access to $\mathcal{F}_{\mathbf{OLE}}$ (Figure **??**) is indistinguishable from a real execution of the protocol from the point of view of an external entity called the *environment*. To prove this indistinguishability, we have to build a simulator that simulates the execution of the protocol in the ideal setting and generates messages on behalf of the honest simulated parties, while trying to extract the dishonest party's inputs and feed them in $\mathcal{F}_{\mathbf{OLE}}$. In particular, for a dishonest Alice, we have to demonstrate the existence of a simulator, $\mathcal{S}_A$, that generates messages on behalf of honest Bob and extracts Alice's input (a, b) which, in turn, feeds into $\mathcal{F}_{\mathbf{OLE}}$. To this end, we consider that $\mathcal{S}_A$ simulates an attack by Bob at all rounds, i, of the protocol which allows to extract the m values of Alice (a_i^0, b_i^0). However, the commit-and-open scheme described above is designed to catch such an attack, and to work around this issue we substitute the ideal commitment functionality, $\mathcal{F}_{\mathbf{COM}}$, with a fake commitment functionality, $\mathcal{F}_{\mathbf{FakeCOM}}$, that allows $\mathcal{S}_A$ to open the commitments later [80]. From the remaining n values (a_i^0, b_i^0), $\mathcal{S}_A$ computes Alice's input (a, b) and feeds it to $\mathcal{F}_{\mathbf{OLE}}$.

For a dishonest Bob, we have to show the existence of a simulator, $\mathcal{S}_B$, that generates messages on behalf of honest Alice and extracts Bob's input x. We assume that $\mathcal{S}_B$ has full control over $\mathcal{F}_{\mathbf{COM}}$, which means that it has access to Bob's m committed values (x_i^0, r_i); the input x can be easily extracted from these values.

Protocol generalization. We start by generalizing the main relation (5.4) to Galois Fields of prime-power dimension, $GF(d^M)$ for $M > 1$. Then, we show how we can obtain a protocol for quantum VOLE. In particular, from n WOLE instances, we are able to generate a VOLE with size proportional to n, and we bound this proportion by the min-entropy value on the WOLE instances.

5.1.1 Organization

Check in the end! In Section **??**, we introduce notation and state results and definitions, that we will use in the rest of the paper. In Section 5.3, in order to build some intuition, we present a QOLE protocol that is secure only if we consider Bob to be semi-honest; in case Bob is dishonest, its security is compromised. In Section 5.4, we construct a secure protocol that comprises the first part of our main QOLE protocol presented in Section 5.4.2. Next, in Section 5.5, we prove the security of the QOLE protocol in the quantum-UC framework. Finally, in the last Section 5.6, we show how to generalize the presented QOLE protocol to Galois Fields of prime-power dimensions and we also present a quantum-UC secure protocol achieving VOLE.

5.2 Mutually unbiased bases

In this section, we present the basics and some properties of mutually unbiased bases (MUBs) in some high-dimensional Hilbert space $\mathcal{H}^d$. This is the main tool that is used in our protocol. For more details about MUBs see [35].

Definition 8. *Let $\mathcal{B}_0 = \{|\psi_1\rangle, \dots, |\psi_d\rangle\}$ and $\mathcal{B}_1 = \{|\phi_1\rangle, \dots, |\phi_d\rangle\}$ be orthonormal bases in the d-dimensional Hilbert space $\mathcal{H}^d$. They are said to be mutually unbiased if $|\langle\psi_i|\phi_j\rangle| = \frac{1}{\sqrt{d}}$ for all $i, j \in \{1, \dots, d\}$. Furthermore, a set $\{\mathcal{B}_0, \dots, \mathcal{B}_m\}$ of orthonormal bases on $\mathcal{H}^d$ is said to be a set of MUBs if, for every $i \neq j$, $\mathcal{B}_i$ is mutually unbiased with $\mathcal{B}_j$.*

MUBs are extensively used in quantum cryptography because, in some sense, these bases are as far as possible from each other and the overlap between two elements from different bases is constant. Let $\{\,|0\rangle, \dots, |d-1\rangle\,\}$ be the computational basis of $\mathcal{H}^d$, where d is a prime number, and $\{\,|\tilde{0}\rangle, \dots, \left|\widetilde{d-1}\right\rangle\,\}$ be the dual basis which is given by the Fourier transform on the computational basis:

$$\left|\tilde{j}\right\rangle = \frac{1}{\sqrt{d}} \sum_{i=0}^{d-1} \omega^{-ij} |i\rangle ,$$

where $\omega = e^{\frac{2\pi i}{d}}$. We can easily verify that the computational basis and its dual basis are mutually unbiased, and we will make use of the following two operators, V_a^0 and V_0^b, to encode Alice's functions during the first (quantum) phase of the protocol.

Definition 9 (Shift operators)**.** *The shift operator V_a^0 shifts the computational basis by a elements, i.e.*

$$V_a^0 |i\rangle = |i+a\rangle .$$

Similarly, the dual shift operator V_0^b shifts the dual basis by b elements, i.e.

$$V_0^b |\tilde{j}\rangle = \left|\widetilde{j-b}\right\rangle .$$

The operators V_a^0 and V_0^b are diagonal in the dual and computational basis, respectively[1], i.e.

$$V_a^0 = \sum_{j=0}^{d-1} \omega^{aj} |\tilde{j}\rangle\langle\tilde{j}| \text{ and } V_0^b = \sum_{i=0}^{d-1} \omega^{bi} |i\rangle\langle i| .$$

Furthermore, following the convention from [35], we can define

$$V_a^b \;:=\; V_0^b V_a^0 = \sum_{l=0}^{d-1} \omega^{(l+a)b} |l+a\rangle\langle l| ,$$

obtaining the so-called *Heisenberg-Weyl operators*. These operators form a group of unitary transformations with d^2 elements; the group has $d+1$ commuting abelian subgroups of d elements, and for each abelian subgroup, there exists a basis of joint eigenstates of all V_a^b in the subgroup. These $d+1$ bases are pairwise mutually unbiased. Let $x \in \mathbb{Z}_{d+1}$ label the abelian subgroups, let $l \in \mathbb{Z}_d$ label the elements of each subgroup, and let U_l^x denote the corresponding subgroup operators. Finally, let the $i-$th basis element associated with the $x-$th subgroup be denoted by $|e_i^x\rangle$. Then, it can be seen that [35],

$$U_l^x = \sum_{i=0}^{d-1} \omega^{il} |e_i^x\rangle\langle e_i^x| \text{ and } |e_i^x\rangle = \frac{1}{\sqrt{d}} \sum_{l=0}^{d-1} \omega^{-il+\frac{l(l-1)}{2}x} |l\rangle ,$$

where

$$U_l^x = \alpha_l^x V_l^{xl} \text{ with } \alpha_l^x = \omega^{-xl(l+1)/2}.$$

One can show that

$$V_a^b |e_0^x\rangle = c_{x,a,b} \left|e_{ax-b}^x\right\rangle , \; x \in \mathbb{Z}_d \text{ and } V_a^b \left|e_0^d\right\rangle = c_{d,a,b} \left|e_a^d\right\rangle \text{ for } x = d,$$

or more generally

$$V_a^b |e_r^x\rangle = c_{a,b,x,r} \left|e_{ax-b+r}^x\right\rangle , \text{ with } c_{a,b,x,r} = \omega^{ar+\frac{a(a+1)}{2}x}. \tag{5.2}$$

[1] Note that V_a^0 and V_0^b can be seen as a generalization of the Pauli X and Z operators, respectively.

Proof. By definition, we have that

$$\begin{aligned}
V_a^b |e_r^x\rangle &= \frac{1}{\sqrt{d}} \sum_{k,l=0}^{d-1} \omega^{(k+a)b} |k+a\rangle\langle k| \, \omega^{-rl+\frac{l(l-1)}{2}x} |l\rangle \\
&= \frac{1}{\sqrt{d}} \sum_{l=0}^{d-1} \omega^{(l+a)b} \omega^{-rl+\frac{l(l-1)}{2}x} |l+a\rangle \\
&= \frac{1}{\sqrt{d}} \sum_{l=0}^{d-1} \omega^{lb} \omega^{-r(l-a)+\frac{(l-a)(l-a-1)}{2}x} |l\rangle \\
&= \frac{\omega^{ar}}{\sqrt{d}} \sum_{l=0}^{d-1} \omega^{-l(-b+r)} \omega^{\frac{l(l-1)}{2}x+lax+\frac{a(a+1)}{2}x} |l\rangle \\
&= \frac{\omega^{ar}}{\sqrt{d}} \sum_{l=0}^{d-1} \omega^{-l(-b+r)} \omega^{\frac{l(l-1)}{2}x-lax+\frac{a(a+1)}{2}x} |l\rangle \\
&= \frac{\omega^{ar+\frac{a(a+1)}{2}x}}{\sqrt{d}} \sum_{l=0}^{d-1} \omega^{-l(ax-b+r)+\frac{l(l-1)}{2}x} \\
&= \omega^{ar+\frac{a(a+1)}{2}x} \left| e_{ax-b+r}^x \right\rangle
\end{aligned}$$

□

This last property is the main ingredient for the construction of our protocol as it encodes a linear evaluation based on values a, b and $x \in \mathbb{Z}_d$[2]. In our protocol, we take a, b – that determine the operators V_a^b – to be Alice's inputs and x to be Bob's input.

Finally, let us see how the operators V_a^b act on the so-called *generalized Bell states*, since Bob's attack to the protocol is based on that. We start with the definition of the *seed* Bell state

$$|B_{0,0}\rangle = \frac{1}{\sqrt{d}} \sum_i |i^*, i\rangle \,,$$

where the map $|\psi\rangle \to |\psi^*\rangle$ is defined by taking the complex conjugate of the coefficients:

$$|\psi\rangle = \sum_i \beta_i |i\rangle \to |\psi^*\rangle = \sum_i \beta_i^* |i\rangle \,.$$

Using the properties of the operators V_a^b, we can derive the rest of the generalized Bell states from the seed state, as

$$|B_{a,b}\rangle = (\mathbb{1} \otimes V_a^b) |B_{0,0}\rangle = \frac{1}{\sqrt{d}} \sum_{i=0}^{d-1} \omega^{(i+a)b} |i^*, i+a\rangle \,, \tag{5.3}$$

[2]While $x \in \mathbb{Z}_{d+1}$, henceforth we consider $x \in \mathbb{Z}_d$, since we only use d out of the $d+1$ MUBs.

and one can prove that the set $\{|B_{a,b}\rangle\}_{(a,b)\in\mathbb{Z}_d^2}$ constitutes an orthonormal maximally entangled basis in the Hilbert space of two-qudit states [35].

5.3 Semi-honest QOLE protocol

In order to build some intuition on the proposed protocol for QOLE, we start by presenting a simpler protocol that is only secure under the semi-honest adversarial model. This semi-honest version leverages the properties of MUBs explored in Section 5.2 and, in particular, the one presented in expression (5.2). As we saw, given the set of MUBs $\{|e_r^x\rangle\}_{r\in\mathbb{Z}_d}$, $\forall x \in \mathbb{Z}_d$, the operators V_a^b simply permute the elements inside the basis x, according to a linear combination of the elements a, b, x and r:

$$V_a^b\,|e_r^x\rangle = c_{a,b,x,r}\,\left|e_{ax-b+r}^x\right\rangle. \tag{5.4}$$

Alice and Bob can use the above property to compute together a linear function $f(x) = ax-b$, where Alice chooses the parameters a and b, and Bob chooses the input element x. The protocol summarized in Figure 5.1. Bob starts by choosing a basis x and an element r therein, and prepares the state $|e_r^x\rangle$: the basis choice x plays the role of the input element x, and the basis element r is used to enhance Bob's security against a potentially dishonest Alice. Then, he sends the state $|e_r^x\rangle$ to Alice, who, in turn, applies on it the operator V_a^b and sends back to Bob the resulting state. According to (5.4), Bob receives $\left|e_{ax-b+r}^x\right\rangle$, measures it in the x basis, and outputs the linear function evaluation $f(x) = ax-b$ by subtracting r. Thus, the correctness of the protocol is ensured by expression (5.4).

As far as the security of this protocol is concerned, we can easily see that it is secure against a dishonest Alice. From her point of view, all the density matrices describing the several possible cases for $x = 0,\dots,d-1$ are maximally mixed states. Therefore, she cannot know anything about the value of x.

If, moreover, Bob is semi-honest the protocol remains secure. On the other hand, if Bob is dishonest and deviates from the protocol, he is able to find out Alice's inputs a and b with certainty. In Section 5.2 equation (5.3), we saw that the generalized Bell basis is generated by Alice's operators, V_a^b, i.e. $|B_{a,b}\rangle = (1\otimes V_a^b)\,|B_{0,0}\rangle$, and Bob can make use of this property in order to extract her inputs a and b. His attack can be described as follows:

1. Bob prepares the state $|B_{0,0}\rangle$ and sends the second qudit to Alice.
2. Alice applies her chosen operator V_a^b.
3. Bob measures both qudits in the generalized Bell basis and outputs a, b.

Semi-honest QOLE

Alice's input: $(a, b) \in \mathbb{Z}_d^2$
Bob's input: $x \in \mathbb{Z}_d$

1. Bob randomly generates $r \in \mathbb{Z}_d$. He prepares and sends the state $|e_r^x\rangle$ to Alice.
2. Alice prepares the operator V_a^b according to her inputs a and b. She then applies V_a^b to Bob's state: $V_a^b |e_r^x\rangle = c_{x,a,b,r} \left|e_{ax-b+r}^x\right\rangle$. She sends the resulting state back to Bob.
3. Bob measures in the basis x, subtracts r, and outputs the desired result $ax - b =: f(x)$.

Alice's output: $\perp$
Bob's output: $f(x)$

Figure 5.1: Semi-honest QOLE protocol.

It becomes clear that the protocol is secure only as long as Bob does not deviate from it; a dishonest Bob can break its security by performing the above attack. Therefore, we have to make sure that Bob sticks to the protocol. To achieve this, we apply a *commit-and-open* scheme [29] that can be briefly described as follows: Bob runs step 1. of the Semi-honest QOLE protocol (see Figure 5.1) multiple times, say m in total, for multiple values of x_i, and r_i, for $i \in [m]$ and commits to these values by means of the functionality $\mathcal{F}_{\mathbf{COM}}$ (see Figure 2.5). Then, he sends these states to Alice, who, in turn, asks him to disclose his chosen x_i's and r_i's for some of the m instances that she chooses. The functionality $\mathcal{F}_{\mathbf{COM}}$ forwards these committed values to Alice and she measures the corresponding received states in the disclosed bases. She can, thus, verify whether she got the right basis element for all the instances she chose to check. If Bob had used the Bell state $|B_{0,0}\rangle$ in one out of the m instances, then the probability of Alice getting the correct result after measuring the state in the committed basis would be $\frac{1}{d}$. In other words, Bob would get caught with high probability $1 - \frac{1}{d}$. Furthermore, if he chooses to attack all the instances, the probability of Alice getting correctly all the results is negligible, i.e. exponentially small in the number of instances, m. We explore this in detail in the next section, where we present a QOLE protocol secure against dishonest adversaries.

5.4 QOLE protocol

Our QOLE protocol is divided into two main phases: a quantum phase and a classical post-processing phase. The first phase uses quantum communication to generate several instances of OLE with random inputs. These instances may leak some information to the parties, therefore we refer to them as random weak OLE (RWOLE). The second phase is purely classical. It uses the RWOLE instances and extracts one classical OLE instance. The post-processing phase has two phases. It implements a derandomization procedure followed by an extraction phase that serves as a privacy amplification method. The full protocol is presented in Figure 5.5. Before proceeding, it is worth mentioning that we consider that neither dishonest party maliciously aborts the protocol. Indeed, in our setting, such a behaviour does not provide an advantage for learning the other party's input. The only case to abort the protocol is when honest Alice catches Bob cheating during the *commit-and-open* stage.

In the next sections, we break down the protocol, show its correctness and retrieve some technical lemmas used for the security proof. In Section 5.5, we prove the protocol to be secure in the quantum-UC model against static dishonest adversaries.

Notation. During the RWOLE phase, $\mathbf{F}_0 = (F_1^0, F_2^0, \ldots, F_n^0)$ is the vector whose components are the random variables associated to Alice's functions. Each F_i^0 ranges over the set of affine functions in $\mathbb{Z}_d$ such that $\Pr(F_i^0(x) = a_i^0 x + b_i^0)$ is uniform for all $i \in [n]$. We do not distinguish the set of affine functions in $\mathbb{Z}_d$ from $\mathbb{Z}_d^2$. The classical values $\mathbf{F}_0$ are saved in the Hilbert space $\mathcal{H}_{\mathbf{F}_0}$. The same holds for the derandomization phase, where $\mathbf{F}$ denotes the random variable for Alice's functions in the protocol $\pi^n_{\mathbf{WOLE}}$. $\mathbf{X}_0$ and $\mathbf{Y}_0$ are the random variables for $\mathbf{x}_0, \mathbf{y}_0 \in \mathbb{Z}_d^n$ in the RWOLE phase. and $\mathbf{X}$ and $\mathbf{Y}$ the corresponding random variables for $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_d^n$ in the post-processing phase. Also, we use A' and B' to denote the system that a dishonest Alice and Bob, respectively, hold at the end of the execution of the protocol.

5.4.1 RWOLE phase

We now introduce the quantum phase of the proposed QOLE protocol, which we informally call the random weak OLE (RWOLE) phase. We denote by π^n_{RWOLE} the protocol that implements this RWOLE phase and we present it in Figure 5.2. The protocol π^n_{RWOLE} is divided into four phases: Initialization, Test, Computation and Measurement.

If both parties are honest the protocol is correct: if Alice is honest, her functions $\mathbf{F}_0$ are chosen uniformly at random, and if Bob is honest he will obtain $\left| e^{x_i^0}_{a_i^0 x_i^0 - b_i^0 + r_i} \right\rangle_{i \in [n]}$ according to Equation (5.4).

Protocol $\pi^n_{\mathbf{RWOLE}}$

Parameters: n, number of output qudits; t, proportion of receiver test qudits.

(Initialization Phase:)

1. Bob randomly generates $m = (1+t)n$ different pairs (x_i^0, r_i) and commits to them by sending $(\texttt{commit}, (i, x_i^0, r_i))$ to $\mathcal{F}_{\mathbf{COM}}$. He prepares the states $\left|e^{x_i^0}_{r_i}\right\rangle_{i\in[m]}$ and sends them to Alice.

(Test Phase:)

2. Alice randomly chooses a subset of indices $T \subset [m]$ of size tn and sends it to Bob.

3. Bob sends $(\texttt{open}, i)$, $i \in T$, to $\mathcal{F}_{\mathbf{COM}}$ and $\mathcal{F}_{\mathbf{COM}}$ sends to Alice $(\texttt{open}, (i, x_i^0, r_i))$, $i \in T$.

4. Alice measures the received qudits in the corresponding x_i^0 basis for $i \in T$, and checks whether the received commitments are compatible with her measurements. In case there is no error she proceeds, otherwise she aborts. After the Test Phase, we relabel and identify $[n] = [m] \setminus T$.

(Computation Phase:)

6. Alice randomly generates n pairs (a_i^0, b_i^0) and prepares $V^{b_i^0}_{a_i^0}$ for $i \in [n]$.

7. Alice applies these operators to the received states, i.e. $V^{b_i^0}_{a_i^0}\left|e^{x_i^0}_{r_i}\right\rangle = c_{x_i^0, a_i^0, b_i^0, r_i}\left|e^{x_i^0}_{a_i^0 x_i^0 - b_i^0 + r_i}\right\rangle$, for $i \in [n]$, and sends the resulting states to Bob.

(Measurement Phase:)

8. Bob measures the received states in the basis x_i^0 for $i \in [n]$ and gets the states $\left|e^{x_i^0}_{a_i^0 x_i^0 - b_i^0 + r_i}\right\rangle$, $i \in [n]$. Finally, he subtracts r_i, for $i \in [n]$ from his results.

Alice's output: (a_i^0, b_i^0), for $i \in [n]$.
Bob's output: (x_i^0, y_i^0), where $y_i^0 = g_i(x_i^0) = a_i^0 x_i^0 - b_i^0$ for $i \in [n]$.

Figure 5.2: RWOLE protocol.

Security. In the case of a dishonest Alice, it is straightforward to verify that the security property of the semi-honest protocol still holds; following the same reasoning, we can conclude that she cannot learn anything about Bob's input or output values (x_i^0, y_i^0). In the case of dishonest Bob, though, these random instances of OLE might leak some information on Alice's random functions $\mathbf{F}_0$ to him. To quantify this side information of Bob, we must bound the min-entropy $H_{\min}(\mathbf{F}_0|B')_{\rho_{\mathbf{F}_0 B'}}$, on the state $\rho_{\mathbf{F}_0 B'}$, which is the output state of the real execution of $\pi^n_{\mathbf{RWOLE}}$. The following lemma shows that $\rho_{\mathbf{F}_0 B'}$ is at least $\epsilon-$close to an ideal state $\sigma_{\mathbf{F}_0 B'}$ independently of the attack that the dishonest party may perform. This ideal state $\sigma_{\mathbf{F}_0 B'}$ has the important property of having a bound on $H_{\min}(\mathbf{F}_0|B')_{\sigma_{\mathbf{F}_0 B'}}$ that is proportional to the security parameter.

Lemma 10 (Security against dishonest Bob)**.** *Let $\rho_{\mathbf{F}_0 B'}$ be the state given by the real execution of the protocol $\pi^n_{\boldsymbol{RWOLE}}$, where $\mathbf{F}_0$ is the system saving Alice's functions, B' is Bob's (possibly quantum) system. Fix $\zeta \in]0, 1-\frac{1}{d}]$ and let*

$$\epsilon(\zeta, n) = \exp\left(-\frac{2\zeta^2 t^2 n^2}{(nt+1)(t+1)}\right).$$

Then, for any attack of a dishonest Bob, there exists an ideal classical-quantum state $\sigma_{\mathbf{F}_0 B'}$, such that

1. $\sigma_{\mathbf{F}_0 B'} \approx_\epsilon \rho_{\mathbf{F}_0 B'}$,
2. $H_{\min}(\mathbf{F}_0|B')_{\sigma_{\mathbf{F}_0 B'}} \geq \frac{n \log d}{2}(1 - h_d(\zeta))$,

where $h_d(\zeta)$ is given in Definition 1.

The proof comprises two parts corresponding to the two conditions of Lemma 10: first, we prove that the state just before the *Computation Phase* is close to the ideal state $\sigma_{\mathbf{F}_0 B'}$; and then, we prove that the operators applied by Alice to $\sigma_{\mathbf{F}_0 B'}$ increase the min-entropy by a specific amount that is proportional to the number of output qudits, n. We present the proof in A, where we follow the same reasoning as Damgård et al. in Section 4.3 of [29], and adapt it to our case. We also use certain results from [73] in order to establish the lower bound given by property 2.

5.4.2 Post-processing phase

The $\pi^n_{\mathbf{RWOLE}}$ protocol (see Figure 5.2) generates several instances of RWOLE, which leak information to Bob about Alice's inputs. In this section, we present the post-processing phase that allows to extract one secure QOLE out of several RWOLE instances. Combining these instances is sufficient to generate a secure QOLE protocol, because Bob has only a negligible probability of attacking *all* the weak instances without being caught; indeed, if he chooses to attack one of the instances the probability of Alice not aborting is $\frac{1}{t+1} + \frac{t}{d(1+t)}$, while if he chooses to attack all instances this probability becomes $\frac{1}{d^{tn}}$, which is negligible in n, thus ensuring the asymptotic security of our protocol. The post-processing comprises two subprotocols: the first is a derandomization protocol (Figure 5.3) that integrates the randomized outputs of RWOLE into a deterministic scheme where Alice and Bob choose their inputs))); the second is an extraction protocol (Figure 5.4) that generates a secure QOLE protocol from these deterministic weak instances by means of a two-universal family of hash functions. Note that the classical post-processing phase does not give any advantage to a potentially dishonest Alice, therefore we only need to prove security against dishonest Bob.

Derandomization

Our derandomization protocol, denoted as $\pi^n_{\mathbf{WOLE}}$ and summarized in Figure 5.3, reduces the randomized RWOLE instances into deterministic ones, which we informally call weak OLE (WOLE). The output of $\pi^n_{\mathbf{WOLE}}$ is still a weak version of OLE because Bob is allowed to have some knowledge on Alice's inputs. The difference between RWOLE and WOLE is that the parties now choose their inputs. Our derandomization protocol is an adaptation of the derandomization protocol in [**?**]. We denote by $*$ the product of two matrices of the same dimensions, such that the result is also a matrix of the same dimensions whose elements are the product of the respective elements of the operand matrices.

Protocol $\pi^n_{\mathbf{WOLE}}$

Alice's input: $(\boldsymbol{a}, \boldsymbol{b}) \in \mathbb{Z}_d^{2n}$
Bob's input: $\boldsymbol{x} \in \mathbb{Z}_d^n$

1. Alice and Bob run the $\pi^n_{\mathbf{RWOLE}}$ protocol and receive $(\boldsymbol{a}_0, \boldsymbol{b}_0)$ and $(\boldsymbol{x}_0, \boldsymbol{y}_0)$, respectively.
2. Bob computes and sends to Alice $\boldsymbol{c} = \boldsymbol{x} - \boldsymbol{x}_0$.
3. Alice computes and sends to Bob $\boldsymbol{d} = \boldsymbol{a} - \boldsymbol{a}_0$ and $\boldsymbol{s} = \boldsymbol{b}_0 + \boldsymbol{a} * \boldsymbol{c} + \boldsymbol{b}$.
4. Bob computes $\boldsymbol{y} = \boldsymbol{y}_0 + \boldsymbol{x} * \boldsymbol{d} - \boldsymbol{d} * \boldsymbol{c} + \boldsymbol{s}$.

Alice's output: $\perp$
Bob's output: $\boldsymbol{y} = \boldsymbol{a} * \boldsymbol{x} + \boldsymbol{b}$

Figure 5.3: WOLE protocol.

Security. The requirements to prove security against dishonest Bob are summarized in Lemma 11, which is very similar in structure to Lemma 10. We show that the real output state $\rho_{\mathbf{F}B'}$ of the protocol $\pi^n_{\mathbf{WOLE}}$ is $\epsilon-$close to an ideal state $\sigma_{\mathbf{F}B'}$, which has min-entropy lower-bounded by a fixed value proportional to the security parameter n. Intuitively, this means that Bob's state is indistinguishable from a state where his knowledge on Alice's inputs is limited.

Lemma 11. *Let $\rho_{\mathbf{F}B'}$ be the state given by the real execution of the protocol $\pi^n_{\boldsymbol{WOLE}}$, where $\mathbf{F}$ is the system saving Alice's inputs, B' is Bob's (possibly quantum) system. Fix $\zeta \in]0, 1-\frac{1}{d}]$ and let*

$$\epsilon(\zeta, n) = \exp\left(-\frac{2\zeta^2 t^2 n^2}{(nt+1)(t+1)}\right). \tag{5.5}$$

Then, for any attack of a dishonest Bob, there exists a classical-quantum state $\sigma_{\mathbf{F}B'}$ such that

1. $\sigma_{\mathbf{F}B'} \approx_\epsilon \rho_{\mathbf{F}B'}$,

2. $H_{\min}(\mathbf{F}|B')_{\sigma_{\mathbf{F}B'}} \geq \frac{n \log d}{2}(1 - h_d(\zeta))$,

where $h_d(\zeta)$ is given in Definition 1.

Proof. Alice holds the system $A = \mathbf{F}\mathbf{F}_0\mathbf{C}\mathbf{D}\mathbf{S}$, where $\mathbf{F} = (\mathbf{F}_{\boldsymbol{a}}, \mathbf{F}_{\boldsymbol{b}})$ refers to her inputs $(\boldsymbol{a}, \boldsymbol{b}) \in \mathbb{Z}_d^{2n}$, $\mathbf{F}_0 = (\mathbf{F}_{\boldsymbol{a}_0}, \mathbf{F}_{\boldsymbol{b}_0})$ is the subsystem obtained from the RWOLE phase, and $\mathbf{C}, \mathbf{D}$ and $\mathbf{S}$ are classical subsystems used to save the values of $\boldsymbol{c}$, $\boldsymbol{d}$, and $\boldsymbol{s}$ from the protocol, respectively. Bob holds the system $B' = \mathbf{C}\mathbf{D}\mathbf{S}B'_0$ where $\mathbf{C}, \mathbf{D}$ and $\mathbf{S}$ are the subsystems on Bob's side where the values of $\boldsymbol{c}$, $\boldsymbol{d}$ and $\boldsymbol{s}$ are saved, respectively, and $B'_0 = \mathbf{Y}_0 E_0$ is his (possibly quantum) system generated from the RWOLE phase.

To prove property 1., we will use Lemma 10, namely that the state $\rho_{\mathbf{F}_0 B'_0}$ resulting from the RWOLE scheme is $\epsilon-$close to the ideal state $\sigma_{\mathbf{F}_0 B'_0}$. Then, we will show that the operations applied to $\rho_{\mathbf{F}_0 B'_0}$ during the derandomization process can only decrease the distance between the real and the ideal output states of the WOLE protocol, thus keeping them at least $\epsilon-$close. We start by specifying the operators corresponding to the classical operations executed in steps 2 and 3 of $\pi^n_{\mathbf{WOLE}}$. In step 2, a dishonest Bob can send to Alice some value $\boldsymbol{c}$ that depends on his system B'_0. So, he starts by applying a CPTP map $\mathcal{T}_{B'_0 \to \mathbf{C}B'_0} : \mathcal{P}\left(\mathcal{H}_{B'_0}\right) \to \mathcal{P}\left(\mathcal{H}_{B'_0} \otimes \mathcal{H}_{\mathbf{C}}\right)$ to his state and then projects it into the Hilbert space $\mathcal{H}_{\mathbf{C}}$. The operator for step 2 is a CPTP map

$$\mathcal{O}^{(2)} : \mathcal{P}\left(\mathcal{H}_{\mathbf{F}_0} \otimes \mathcal{H}_{B'_0}\right) \to \mathcal{P}\left(\mathcal{H}_{\mathbf{F}_0} \otimes \mathcal{H}_{\mathbf{C}} \otimes \mathcal{H}_{\mathbf{D}} \otimes \mathcal{H}_{\mathbf{S}} \otimes \mathcal{H}_{B'_0}\right)$$

described by his action on some general quantum state ρ, as

$$\mathcal{O}^{(2)}(\rho) = \mathbb{1} \otimes \sum_{\boldsymbol{d},\boldsymbol{s},\boldsymbol{c}} |\boldsymbol{c}\rangle\langle\boldsymbol{c}|_{\mathbf{C}}\, \mathcal{T}_{B'_0 \to \mathbf{C}B'_0}(\rho)\, |\boldsymbol{c}\rangle\langle\boldsymbol{c}|_{\mathbf{C}} \otimes |\boldsymbol{d}\rangle\langle\boldsymbol{d}|_{\mathbf{D}} \otimes |\boldsymbol{s}\rangle\langle\boldsymbol{s}|_{\mathbf{S}}\,.$$

In step 3, Bob takes no action. Since Alice is honest, the operator for this step simply describes her action on subsystems $\mathbf{D}$ and $\mathbf{S}$ according to her choice at subsystem $\mathbf{F}$. This operator is a CPTP map

$$\mathcal{O}^{(3)} : \mathcal{P}\left(\mathcal{H}_{\mathbf{F}_0} \otimes \mathcal{H}_{\mathbf{C}} \otimes \mathcal{H}_{\mathbf{D}} \otimes \mathcal{H}_{\mathbf{S}} \otimes \mathcal{H}_{B'_0}\right) \to \mathcal{P}\left(\mathcal{H}_{\mathbf{F}} \otimes \mathcal{H}_{\mathbf{F}_0} \otimes \mathcal{H}_{\mathbf{C}} \otimes \mathcal{H}_{\mathbf{D}} \otimes \mathcal{H}_{\mathbf{S}} \otimes \mathcal{H}_{B'_0}\right)$$

described by his action on some general quantum state ρ, as

$$\mathcal{O}^{(3)}(\rho) = \frac{1}{d^{2n}} \sum_{\boldsymbol{a},\boldsymbol{b}} \mathcal{P}^{\boldsymbol{a},\boldsymbol{b}} \, \rho \, (\mathcal{P}^{\boldsymbol{a},\boldsymbol{b}})^{\dagger},$$

where

$$\begin{aligned}\mathcal{P}^{\boldsymbol{a},\boldsymbol{b}} = |\boldsymbol{a},\boldsymbol{b}\rangle_{\mathbf{F}} \otimes \sum_{\boldsymbol{a}_0,\boldsymbol{b}_0,\boldsymbol{c}} |\boldsymbol{a}_0,\boldsymbol{b}_0\rangle\langle\boldsymbol{a}_0,\boldsymbol{b}_0|_{\mathbf{F}_0} \otimes |\boldsymbol{c}\rangle\langle\boldsymbol{c}|_{\mathbf{C}} \\ \otimes |\boldsymbol{a}-\boldsymbol{a}_0\rangle\langle\boldsymbol{a}-\boldsymbol{a}_0|_{\mathbf{D}} \otimes |\boldsymbol{b}_0+\boldsymbol{a}\cdot\boldsymbol{c}+\boldsymbol{b}\rangle\langle\boldsymbol{b}_0+\boldsymbol{a}\cdot\boldsymbol{c}+\boldsymbol{b}|_{\mathbf{S}} \,.\end{aligned}$$

Note that $\mathcal{O}^{(2)}$ adds subsystems **CDS** and distributes **C** according to Bob's action. The operator $\mathcal{O}^{(3)}$ adds subsystem **F** and projects **DS** according to the information at subsystem $\mathbf{F}\mathbf{F}_0$ and the expressions of $\boldsymbol{d}$ and $\boldsymbol{s}$. Regarding the trace distance between the real and ideal states, we have:

$$\begin{aligned}\delta(\rho_{\mathbf{F}_0B_0'},\sigma_{\mathbf{F}_0B_0'}) &\geq \delta\Big(\mathcal{O}^{(2)}(\rho_{\mathbf{F}_0B_0'}),\mathcal{O}^{(2)}(\sigma_{\mathbf{F}_0B_0'})\Big) \\ &\geq \delta\Big(\mathcal{O}^{(3)}\mathcal{O}^{(2)}(\rho_{\mathbf{F}_0B_0'}),\mathcal{O}^{(3)}\mathcal{O}^{(2)}(\sigma_{\mathbf{F}_0B_0'})\Big) \\ &= \delta(\rho_{\mathbf{F}B'},\sigma_{\mathbf{F}B'}).\end{aligned}$$

For the above inequalities, we took into account that $\mathcal{O}^{(2)}$ and $\mathcal{O}^{(3)}$ are CPTP maps, and as such they do not increase the trace distance (see Lemma 7 in [70]). For the last equality, recall that $B' = \boldsymbol{CDS}B_0'$. Now, from Lemma 10, we have that $\sigma_{\mathbf{F}_0B_0'} \approx_\epsilon \rho_{\mathbf{F}_0B_0'}$. Hence, we conclude that $\delta(\rho_{\mathbf{F}B'},\sigma_{\mathbf{F}B'}) \leq \epsilon(\zeta,n)$ for $\epsilon(\zeta,n)$ given as (5.5), i.e. $\sigma_{\mathbf{F}B'} \approx_\epsilon \rho_{\mathbf{F}B'}$.

We move on to prove property 2. Consider the bijective function $g^{\boldsymbol{c},\boldsymbol{d},\boldsymbol{s}} : \mathbb{Z}_d^{2n} \to \mathbb{Z}_d^{2n}$ given by

$$g^{\boldsymbol{c},\boldsymbol{d},\boldsymbol{s}}(\boldsymbol{x},\boldsymbol{y}) = (\boldsymbol{x}+\boldsymbol{d},\boldsymbol{s}-\boldsymbol{y}-(\boldsymbol{x}+\boldsymbol{d})*\boldsymbol{c})$$

for fixed $\boldsymbol{c},\boldsymbol{d}$ and $\boldsymbol{s}$. Essentially, $g^{\boldsymbol{c},\boldsymbol{d},\boldsymbol{s}}$ describes how the input vector $(\boldsymbol{a},\boldsymbol{b})$ is related to the RWOLE output vector $(\boldsymbol{a}_0,\boldsymbol{b}_0)$:

$$(\boldsymbol{a},\boldsymbol{b}) = g^{\boldsymbol{c},\boldsymbol{d},\boldsymbol{s}}(\boldsymbol{a}_0,\boldsymbol{b}_0) = (\boldsymbol{a}_0+\boldsymbol{d},\boldsymbol{s}-\boldsymbol{b}_0-(\boldsymbol{a}_0+\boldsymbol{d})*\boldsymbol{c}). \tag{5.6}$$

Intuitively, this means that the subsystem **F** is defined by the subsystems $\mathbf{F}_0\mathbf{CDS}$. We can rewrite the action of the operator $\mathcal{O}^{(3)}$ on some general quantum state ρ as follows:

$$\mathcal{O}^{(3)}(\rho) = \frac{1}{d^{2n}} \sum_{\boldsymbol{d},\boldsymbol{s}} \mathcal{P}^{\boldsymbol{d},\boldsymbol{s}} \, \rho \, (\mathcal{P}^{\boldsymbol{d},\boldsymbol{s}})^{\dagger},$$

where

$$\mathcal{P}^{\boldsymbol{d},\boldsymbol{s}} = \sum_{\boldsymbol{a}_0,\boldsymbol{b}_0,\boldsymbol{c}} \left| g^{\boldsymbol{c},\boldsymbol{d},\boldsymbol{s}}(\boldsymbol{a}_0,\boldsymbol{b}_0) \middle\rangle\!\middle\langle \boldsymbol{a}_0,\boldsymbol{b}_0 \right|_{\mathbf{F}} \otimes |\boldsymbol{a}_0,\boldsymbol{b}_0,\boldsymbol{c},\boldsymbol{d},\boldsymbol{s}\rangle\!\langle \boldsymbol{a}_0,\boldsymbol{b}_0,\boldsymbol{c},\boldsymbol{d},\boldsymbol{s}|_{\mathbf{F}_0\boldsymbol{CDS}}\,.$$

Hence, for the min-entropy bound, we have that:

$$\begin{aligned}
H_{\min}(\mathbf{F}\,|\,B')_{\sigma_{\mathbf{F}B'}} &= H_{\min}(\mathbf{F}_{\boldsymbol{a}},\mathbf{F}_{\boldsymbol{b}}\,|\,B')_{\sigma_{\mathbf{F}B'}} \\
&= H_{\min}(f^{\boldsymbol{C},\boldsymbol{D},\boldsymbol{S}}(\mathbf{F}_{\boldsymbol{a}_0},\mathbf{F}_{\boldsymbol{b}_0})\,|\,\mathbf{CDS}B'_0)_{\mathcal{O}^{(3)}\mathcal{O}^{(2)}\sigma_{\mathbf{F}_0B'_0}} \\
&\geq H_{\min}(\mathbf{F}_{\boldsymbol{a}_0},\mathbf{F}_{\boldsymbol{b}_0}\,|\,\mathbf{CDS}B'_0)_{\mathcal{O}^{(2)}\sigma_{\mathbf{F}_0B'_0}} \qquad (5.7)\\
&\geq H_{\min}(\mathbf{F}_{\boldsymbol{a}_0},\mathbf{F}_{\boldsymbol{b}_0}\,|\,B'_0)_{\sigma_{\mathbf{F}_0B'_0}} \qquad (5.8)\\
&\geq \frac{\log d}{2}(1-h_d(\zeta)). \qquad (5.9)
\end{aligned}$$

The inequality at step (5.7) comes from Lemma 4, as $g^{\boldsymbol{c},\boldsymbol{d},\boldsymbol{s}}$ is bijective. The inequality at step (5.8) comes from Theorem 6.19 in [**?**], as the operator $\mathcal{O}^{(2)}$ takes the form of $\mathcal{O}^{(2)} = \mathbb{1}\otimes\mathcal{M}$, where $\mathcal{M}$ is a CPTP map. The last inequality comes from Lemma 10, property 2.

□

Extraction

In this section, we present our extraction protocol, $\pi_{\mathbf{EXT}}$, that generates one OLE instance using the derandomization protocol $\pi^n_{\mathbf{WOLE}}$ and the two-universal family of hash functions, $\mathfrak{G}$ (see [75] and Definition 5). This family uses the inner product between two vectors in the $\mathbb{Z}_d^n$ space, and since OLE only involves linear operations, we can apply the inner product operation to all vectors $\boldsymbol{a}$, $\boldsymbol{b}$ and $\boldsymbol{y}$ without affecting the overall structure. The protocol $\pi_{\mathbf{EXT}}$ is summarized in Figure 5.4 and uses the n instances of WOLE in such a way that Bob's knowledge on Alice's inputs decreases exponentially with respect to n[3] For this reason, n is our security parameter.

The correctness of $\pi_{\mathbf{EXT}}$ is given by linearity:

$$\begin{aligned}
y &= g_{\boldsymbol{\kappa}}(\boldsymbol{y}) = \boldsymbol{\kappa}\cdot(a_1x+b_1,\dots a_nx+b_n)\\
&= \boldsymbol{\kappa}\cdot\left(\frac{a-\sum_{i=2}^n a_i\kappa_i}{\kappa_1}x+\frac{b-\sum_{i=2}^n b_i\kappa_i}{\kappa_1},\ a_2x+b_2,\dots,a_nx+b_n\right)\\
&= ax+b.
\end{aligned}$$

[3] This extraction step is similar to the privacy amplification step of QKD protocols.

Protocol $\pi_{\mathbf{EXT}}$

Alice's input: $(a, b) \in \mathbb{Z}_d^2$
Bob's input: $x \in \mathbb{Z}_d$

1. Alice chooses randomly some function $g_{\boldsymbol{\kappa}} \in \mathfrak{G}$ and sends it to Bob.
2. Alice randomly generates $a_2, \ldots, a_n, b_2, \ldots, b_n \leftarrow_{\$} \mathbb{Z}_d$. She computes $a_1 = \left(a - \sum_{i=2}^{n} a_i \kappa_i\right)/\kappa_1$ and $b_1 = \left(b - \sum_{i=2}^{n} b_i \kappa_i\right)/\kappa_1$. We write $\boldsymbol{a} = (a_1, \ldots, a_n)$ and $\boldsymbol{b} = (b_1, \ldots, b_n)$.
3. Alice and Bob run the derandomization protocol $\pi^n_{\mathbf{WOLE}}((\boldsymbol{a}, \boldsymbol{b}), \boldsymbol{x})$. Bob receives $\boldsymbol{y}$ as output.
4. Bob computes $y = g_{\boldsymbol{\kappa}}(\boldsymbol{y})$.

Alice's output: $\perp$
Bob's output: y

Figure 5.4: Extraction protocol.

Security. By definition, the derandomization protocol leaks some information to Bob about Alice's inputs $(\boldsymbol{a}, \boldsymbol{b})$. Since $\boldsymbol{y} = \boldsymbol{a} * \boldsymbol{x} + \boldsymbol{b}$, without loss of generality, any leakage of Alice's inputs can be seen as a leakage on just $\boldsymbol{a}$. In this case, the min-entropy of $\mathbf{F} = (\mathbf{F}_{\boldsymbol{a}}, \mathbf{F}_{\boldsymbol{b}})$ should be the same as the min-entropy of $\mathbf{F}_{\boldsymbol{a}}$. Now, recall the $\mathcal{F}_{\mathbf{OLE}}$ functionality definition (see Figure **??**), and note that Bob does not possess any knowledge about Alice's input (a, b) other than what can be deduced from his input and output (x, y). Similarly, since $y = ax + b$, Bob has some knowledge on the relation between a and b and – as b is completely determined by (a, x, y) – we only have to guarantee that a looks uniformly random to Bob. The role of the hash functions used in the above protocol $\pi_{\mathbf{EXT}}$ is precisely to extract a uniformly random a from the leaky vector $\boldsymbol{a}$, while preserving the structure of the OLE. This result is summarized in Lemma 12 and its proof is based on Lemma 9.

Lemma 12. *Let $\rho_{FB'}$ be the state given by the real execution of the protocol $\pi_{\boldsymbol{EXT}}$, where F is the system saving Alice's inputs (a, b), B' is Bob's (possibly quantum) system. Fix $\zeta \in]0, 1 - \frac{1}{d}]$ and let*

$$\epsilon(\zeta, n) = \exp\left(-\frac{2\zeta^2 t^2 n^2}{(nt+1)(t+1)}\right).$$

Then, for any attack of a dishonest Bob, there exists a classical-quantum state $\sigma_{FB'}$, where $F = (F_a, F_b)$, such that

1. *$\sigma_{FB'} \approx_\epsilon \rho_{FB'}$, and*

2. $\delta(\tau_{\mathbb{Z}_d} \otimes \sigma_{B'},\, \sigma_{F_a B'}) \leq K\, 2^{-n\, f_d(\zeta)}$, *where* $K = \frac{\sqrt{d}}{2}$, $f_d(\zeta) = \frac{\log d}{4}(1 - h_d(\zeta))$, n *is the security parameter, and* $h_d(\zeta)$ *is given in Definition 1.*

Proof. To prove property 1, we note that the extraction operation applied to the output of $\pi^n_{\mathbf{WOLE}}$ can be described by a projective operator on the space $F = (F_a, F_b)$. Therefore, as in the case of Lemma 11, property 1 follows from the fact that CPTP maps do not increase the trace distance (see Lemma 7 in [70]).

Regarding property 2, let us first consider Bob's subsystem E to integrate Bob's inputs $\boldsymbol{x}$, i.e. $E = \mathbf{X}E'$. Then, his full system B' is identified with $\mathbf{Y}E = \mathbf{YX}E'$. We have:

$$\begin{aligned} H_{\min}(\mathbf{F} \,|\, \mathbf{Y}E)_{\sigma_{\mathbf{FY}E}} &= H_{\min}(\mathbf{F}_{\boldsymbol{a}}, \mathbf{F}_{\boldsymbol{b}} \,|\, \mathbf{YX}E')_{\sigma_{\mathbf{FY}E}} \\ &= H_{\min}(\mathbf{F}_{\boldsymbol{a}}, \mathbf{Y} - \mathbf{F}_{\boldsymbol{a}}\mathbf{X} \,|\, \mathbf{YX}E')_{\sigma_{\mathbf{F}_a\mathbf{Y}E}} \\ &= H_{\min}(\mathbf{F}_{\boldsymbol{a}} \,|\, \mathbf{YX}E')_{\sigma_{\mathbf{F}_a\mathbf{Y}E}}. \end{aligned}$$

Therefore,

$$H_{\min}(\mathbf{F}_{\boldsymbol{a}} \,|\, \mathbf{Y}E)_{\sigma_{\mathbf{F}_a\mathbf{Y}E}} \geq \frac{n \log d}{2}(1 - h_d(\zeta)).$$

Now, since $\mathfrak{G}$ is a two-universal family of hash functions, we can directly apply Lemma 9 for $l = 1$. It follows that F_a is $\xi-$close to uniform conditioned on $\mathbf{Y}E$, i.e.

$$\delta(\tau_{\mathbb{Z}_d} \otimes \sigma_{\mathbf{Y}E},\, \sigma_{F_a\mathbf{Y}E}) \leq \frac{1}{2}\sqrt{2^{\log d - \frac{n \log d}{2}(1 - h_d(\zeta))}} = K\, 2^{-n\, f_d(\zeta)} =: \xi$$

where $K = \frac{\sqrt{d}}{2}$, $f_d(\zeta) = \frac{\log d}{4}(1 - h_d(\zeta))$ and n is the security parameter.

□

Now, we are in position to combine the above subprotocols ($\pi^n_{\mathbf{RWOLE}}$, $\pi^n_{\mathbf{WOLE}}$ and $\pi_{\mathbf{EXT}}$) and present the full protocol $\pi_{\mathbf{QOLE}}$ in Figure 5.5.

5.5 UC security

In this section, we will show that our protocol $\pi_{\mathbf{QOLE}}$ (see Figure 5.5) is quantum-UC secure. More formally, we will show that $\pi_{\mathbf{QOLE}}$ statistically quantum-UC realizes (see Definition 6) the functionality $\mathcal{F}_{\mathbf{OLE}}$ in the $\mathcal{F}_{\mathbf{COM}}-$hybrid model.

Theorem 2 (quantum-UC security of $\pi_{\mathbf{QOLE}}$). *The protocol $\pi_{\boldsymbol{QOLE}}$ from Figure 5.5 statistically quantum-UC realizes (see Definition 6) $\mathcal{F}_{\boldsymbol{OLE}}$ in the $\mathcal{F}_{\boldsymbol{COM}}-$hybrid model.*

Theorem 2 is proved by combining Lemma 13 and Lemma 14 that we present below. In the former we prove the protocol's security for the case where Alice is dishonest and

Protocol $\pi_{\mathbf{QOLE}}$

Parameters: n, security parameter; tn, number of test qudits.
Alice's input: $(a, b) \in \mathbb{Z}_d^2$
Bob's input: $x \in \mathbb{Z}_d$

(Quantum phase:)

1. Bob randomly generates $m = (1+t)n$ different pairs $(x_i^0, r_i) \in \mathbb{Z}_d^2$ and commits to them by sending $(\texttt{commit}, (i, x_i^0, r_i)_{i\in[m]})$ to $\mathcal{F}_{\mathbf{COM}}$. He also prepares the quantum states $\left|e_{r_i}^{x_i^0}\right\rangle_{i\in[m]}$ and sends them to Alice.
2. Alice randomly chooses a subset of indices $T \subset [m]$ of size tn and sends it to Bob.
3. Bob sends $(\texttt{open}, i)_{i\in T}$ to $\mathcal{F}_{\mathbf{COM}}$ and $\mathcal{F}_{\mathbf{COM}}$ sends to Alice $(\texttt{open}, (i, x_i^0, r_i))_{i\in T}$.
4. Alice measures the received quantum states in the corresponding x_i^0 basis for $i \in T$, and checks whether the received commitments are compatible with her measurements. She proceeds in case there is no error, otherwise she aborts.
5. Alice randomly generates n pairs $(a_i^0, b_i^0) \in \mathbb{Z}_d^2$ and prepares $V_{a_i^0}^{b_i^0}$ for $i \in [m] \setminus T$. We relabel $\boldsymbol{a}_0 = (a_1^0, \dots, a_n^0)$, $\boldsymbol{b}_0 = (b_1^0, \dots, b_n^0)$ and $\boldsymbol{x}_0 = (x_1^0, \dots, x_n^0)$, and from now on identify $[m]\backslash T \equiv [n]$.
6. Alice $\forall i \in [n]$ applies $V_{a_i^0}^{b_i^0}$ to the received state $\left|e_{r_i}^{x_i^0}\right\rangle$, i.e. $V_{a_i^0}^{b_i^0}\left|e_{r_i}^{x_i^0}\right\rangle = c_{x_i^0, a_i^0, b_i^0, r_i}\left|e_{a_i^0 x_i^0 - b_i^0 + r_i}^{x_i^0}\right\rangle$, and sends the resulting states to Bob.
7. Bob $\forall i \in [n]$ measures the received state in the corresponding basis x_i^0, and gets the state $\left|e_{a_i^0 x_i^0 - b_i^0 + r_i}^{x_i^0}\right\rangle$. Finally, $\forall i \in [n]$ he subtracts r_i from his result and gets $y_i^0 = a_i^0 x_i^0 - b_i^0$. We write $\boldsymbol{y}_0 = (y_1^0, \dots, y_n^0)$.

(Post-processing phase:)

8. Bob defines $\boldsymbol{x} = (x, \dots, x)$ as the constant vector according to his input x.
9. Alice chooses randomly some function $g_{\boldsymbol{\kappa}} \in \mathfrak{G}$, and she randomly generates $a_2, \dots, a_n, b_2, \dots, b_n \leftarrow_{\$} \mathbb{Z}_d$. She computes $a_1 = \big(a - \sum_{i=2}^n a_i\kappa_i\big)/\kappa_1$ and $b_1 = \big(b - \sum_{i=2}^n b_i\kappa_i\big)/\kappa_1$. We write $\boldsymbol{a} = (a_1, \dots, a_n)$ and $\boldsymbol{b} = (b_1, \dots, b_n)$.
10. Bob computes and sends to Alice $\boldsymbol{c} = \boldsymbol{x} - \boldsymbol{x}_0$.
11. Alice computes and sends to Bob $\boldsymbol{d} = \boldsymbol{a} - \boldsymbol{a}_0$ and $\boldsymbol{s} = \boldsymbol{b}_0 + \boldsymbol{a} * \boldsymbol{c} + \boldsymbol{b}$.
12. Bob computes $\boldsymbol{y} = \boldsymbol{y}_0 + \boldsymbol{x} * \boldsymbol{d} - \boldsymbol{d} * \boldsymbol{c} + \boldsymbol{s}$.
13. Finally, Alice sends $\boldsymbol{\kappa}$ to Bob and he computes $y = g_{\boldsymbol{\kappa}}(\boldsymbol{y})$.

Alice's output: $\perp$
Bob's output: y

Figure 5.5: QOLE protocol.

Bob is honest, while in the latter we prove security in the case where Alice is honest and Bob dishonest. In the first case, we have:

Lemma 13. *The protocol $\pi_{\boldsymbol{QOLE}}$ (Figure 5.5) statistically quantum-UC realizes (see Definition 6) $\mathcal{F}_{\boldsymbol{OLE}}$ in the $\mathcal{F}_{\boldsymbol{COM}}-$hybrid model in the case of dishonest Alice and honest Bob.*

Proof. We start by presenting the simulator $\mathcal{S}_A$ for the case where Alice is dishonest in Figure 5.6.

Simulator $\mathcal{S}_\mathbf{A}$

(Quantum phase:)

1. $\mathcal{S}_A$ sends `commit` to $\mathcal{F}_{\mathbf{FakeCOM}}$.
2. $\mathcal{S}_A$ generates $m = (1+t)n$ entangled states $|B_{0,0}\rangle_{Q_A Q_S}$ and sends subsystem Q_A to Alice.
3. Alice asks for a set of indices $T \subset [m]$ of size tn.
4. $\mathcal{S}_A$ measures the corresponding elements of subsystem Q_S using tn randomly chosen bases x_i^0 and provides $(\texttt{open}, (i, x_i^0, r_i))$ to $\mathcal{F}_{\mathbf{FakeCOM}}$, $\forall i \in T$.
5. Upon receiving the processed system $\hat{Q}_A$ from Alice, $\mathcal{S}_A$ measures the joint system $\hat{Q}_A Q_S$ and extracts the measurement outcomes $\mathbf{F} = (\boldsymbol{a}_0, \boldsymbol{b}_0) = \big((a_1^0, \ldots, a_n^0), (b_1^0, \ldots, b_n^0)\big)$.

(Post-processing phase:)

6. $\mathcal{S}_A$ randomly generates a vector $\boldsymbol{c}'$ and sends to Alice.
7. Upon receiving $\boldsymbol{d}$ and $\boldsymbol{s}$ from Alice, $\mathcal{S}_A$ extracts $\boldsymbol{a}$ and $\boldsymbol{b}$ based on its knowledge of $(\boldsymbol{a}_0, \boldsymbol{b}_0)$ as follows:
$$\begin{aligned} \boldsymbol{a} &= \boldsymbol{b} + \boldsymbol{a}_0 \\ \boldsymbol{b} &= \boldsymbol{s} - \boldsymbol{b}_0 - \boldsymbol{a} * \boldsymbol{c}'. \end{aligned} \tag{5.10}$$
8. Upon receiving $\boldsymbol{\kappa}$ from Alice, $\mathcal{S}_A$ extracts her inputs (a, b) as follows:
$$\begin{aligned} a &= \boldsymbol{a} \cdot \boldsymbol{\kappa} \\ b &= \boldsymbol{b} \cdot \boldsymbol{\kappa}. \end{aligned} \tag{5.11}$$
9. Finally, $\mathcal{S}_A$ sends (a, b) to the ideal functionality $\mathcal{F}_{\mathbf{OLE}}$.

Figure 5.6: Simulator $\mathcal{S}_A$ against dishonest Alice.

To prove statistical quantum-UC security according to Definition 6, we first consider a sequence of hybrid protocols from H_0 to H_4. The first hybrid protocol, H_0, in the sequence is the real execution of the protocol $\pi_{\mathbf{QOLE}}$, and we gradually change it until obtaining the hybrid H_4 which corresponds to the description of the simulator $\mathcal{S}_A$. By proving indistinguishaility of the hybrids throughout the sequence, we show statistical quantum-UC security for the protocol $\pi_{\mathbf{QOLE}}$ in the case of dishonest Alice.

Hybrid H_0: This is the real execution of the protocol $\pi_{\mathbf{QOLE}}$.

Hybrid H_1: This hybrid is identical to the previous one, H_0, except that we replace the functionality $\mathcal{F}_{\mathbf{COM}}$ with a fake commitment functionality, $\mathcal{F}_{\mathbf{FakeCOM}}$, in which Bob, i.e. the honest party, can commit no value. This fake functionality works as follows:

- Commitment phase: expects a `commit` message from Bob instead of (`commit,` x).
- Open phase: expects a message (`open`, x) (instead of open) and sends (`open`, x) to Alice.

Hybrids H_0 and H_1 are perfectly indistinguishable, as the simulator still opens the commitments in the same way.

Hybrid H_2: This hybrid is identical to the previous one, H_1, except that now $\mathcal{S}_A$ prepares entangled states $|B_{0,0}\rangle_{Q_AQ_S}$ instead of $\left|e_{r_i}^{x_i^0}\right\rangle_{i\in[m]}$, and sends the subsystem Q_A to Alice. Additionally, upon receiving the set of indices, T, from Alice, $\mathcal{S}_A$ measures the corresponding elements of subsystem Q_S using tn randomly chosen bases x_i^0 and provides $(\texttt{open}, (i, x_i^0, r_i))$ to $\mathcal{F}_{\mathbf{FakeCOM}}$, $\forall i \in T$.

Claim 3. The hybrids H_1 and H_2 are indistinguishable.

Proof. From Alice's point of view, the state received is exactly the same in both hybrids. In H_1, since the elements r are chosen randomly,

$$\frac{1}{d}\sum_{r=0}^{d-1}\left|e_r^{x^0}\middle\rangle\middle\langle e_r^{x^0}\right| = \frac{\mathbb{1}_A}{d},$$

for each $x^0 = 0, \ldots, d-1$. In H_2

$$\mathrm{Tr}_{Q_S}\,|B_{0,0}\rangle\langle B_{0,0}| = \frac{\mathbb{1}_A}{d}.$$

Thus, the environment is not able to distinguish the two scenarios. Furthermore, upon Alice's request of the test set, T, the simulator measures in random bases, x_i^0 for $i \in T$, the corresponding qudits of subsystem Q_S. Since both entangled qudits in Q_AQ_S get projected to the some random state, r_i for $i \in T$, $\mathcal{F}_{\mathbf{FakeCOM}}$ provides the correct pair $(x_i^0, r_i)_{i\in T}$ to Alice. Hence, the hybrids H_1 and H_2 are indistinguishable. □

Hybrid H_3: This hybrid is identical to the previous one, H_2, except that now $\mathcal{S}_A$ extracts Alice's elements $\mathbf{F}_0 = (\boldsymbol{a}_0, \boldsymbol{b}_0)$ by applying a joint measurement on the systems $\hat{Q}_A Q_S$ in the generalized Bell basis.

Hybrids H_2 and H_3 are perfectly indistinguishable, as the simulator only changes the measurement basis for the received state and does not communicate with Alice.

Hybrid H_4: This hybrid is identical to the previous one, H_3, except that now $\mathcal{S}_A$ generates $\boldsymbol{c}'$ uniformly at random. Additionally, upon receiving $\boldsymbol{d}$, $\boldsymbol{s}$ and $\boldsymbol{\kappa}$, the simulator extracts Alice's vectors $(\boldsymbol{a}, \boldsymbol{b})$ and inputs (a, b) by computing expressions (5.10) and (5.11). Finally, $\mathcal{S}_A$ sends (a, b) to the ideal functionality $\mathcal{F}_{\mathbf{OLE}}$. Hybrid H_4 corresponds to the description of the simulator $\mathcal{S}_A$.

Hybrids H_3 and H_4 are perfectly indistinguishable for the following reasons: first, from the proof of Claim 3, we have that the vector $\boldsymbol{x}_0$ looks uniformly random to Alice, and consequently, so does $\boldsymbol{c}$. Second, the extraction operations do not require any interaction with Alice. $\square$

We now proceed to the case where Alice is honest and Bob is dishonest. We have:

Lemma 14. *The protocol $\pi_{\boldsymbol{QOLE}}$ (Figure 5.5) statistically quantum-UC realizes (see Definition 6) $\mathcal{F}_{\boldsymbol{OLE}}$ in the $\mathcal{F}_{\boldsymbol{COM}}-$hybrid model in the case of honest Alice and dishonest Bob.*

Proof. We start by presenting the simulator $\mathcal{S}_B$ for the case where Bob is dishonest in Figure 5.7.

Then, we consider the following sequence of hybrid protocols, from H_0 corresponding to the execution of the real protocol to H_2 corresponding to the description of the simulator $\mathcal{S}_B$, and prove that they are indistinguishable in the case of dishonest Bob.

Hybrid H_0: This is the execution of the real protocol $\pi_{\mathbf{QOLE}}$. In this hybrid, $\mathcal{S}_B$ behaves just like honest Alice up to step 6 of $\pi_{\mathbf{QOLE}}$: tests the received qudits (steps 1-4), randomly generates n pairs $(a_i^0, b_i^0)_{i\in[n]}$ (step 5), and applies the respective operators $V_{a_i^0}^{b_i^0}$ for $i \in [n]$ to the received states (step 6).

Hybrid H_1: This hybrid is identical to the previous one, H_0, except that now $\mathcal{S}_B$ extracts Bob's random vector $\boldsymbol{x}_0$ from the commitment functionality $\mathcal{F}_{\mathbf{COM}}$. Additionally,

Simulator $\mathcal{S}_{\mathbf{B}}$

(Quantum phase:)

1. $\mathcal{S}_B$ receives the qudits from Bob and tests them as in the protocol $\pi_{\mathbf{QOLE}}$.
2. $\mathcal{S}_B$ randomly chooses vectors $\boldsymbol{a}_0$ and $\boldsymbol{b}_0$ and applies $V_{a_i^0}^{b_i^0}$, $i \in [n]$ to the received qudits.
3. $\mathcal{S}_B$ extracts the input element $\boldsymbol{x}_0$ from $\mathcal{F}_{\mathbf{COM}}$.

(Post-processing phase:)

4. Upon receiving $\boldsymbol{c}$ from Bob, $\mathcal{S}_B$ extracts his input x as $\boldsymbol{x} = \boldsymbol{c} + \boldsymbol{x}_0$.
5. $\mathcal{S}_B$ sends x to $\mathcal{F}_{\mathbf{OLE}}$ and receives y.
6. $\mathcal{S}_B$ randomly generates the elements $a' \leftarrow_\$ \mathbb{Z}_d$, $\boldsymbol{\kappa} \leftarrow_\$ \mathbb{Z}_d^n$ and $a_2, \ldots, a_n, b_2, \ldots, b_n \leftarrow_\$ \mathbb{Z}_d$.
7. $\mathcal{S}_B$ computes $b' = a'x - y$, $a_1 = \left(a' - \sum_{i=2}^n a_i \kappa_i\right)/\kappa_1$ and $b_1 = \left(b' - \sum_{i=2}^n b_i \kappa_i\right)/\kappa_1$.
8. $\mathcal{S}_B$ sends $\boldsymbol{d} = \boldsymbol{a} - \boldsymbol{a}_0$, $\boldsymbol{s} = \boldsymbol{b}_0 + \boldsymbol{a} * \boldsymbol{c} + \boldsymbol{b}$ and $\boldsymbol{\kappa}$ to Bob.

Figure 5.7: Simulator $\mathcal{S}_B$ against dishonest Bob.

upon receiving $\boldsymbol{c}$ from Bob, $\mathcal{S}_B$ extracts Bob's input x by computing $\boldsymbol{c} + \boldsymbol{x}_0$. Then, $\mathcal{S}_B$ sends the extracted element x to $\mathcal{F}_{\mathbf{OLE}}$ and receives y.

Hybrids H_0 and H_1 are perfectly indistinguishable, because $\mathcal{S}_B$ only interacts with Bob when receiving the element $\boldsymbol{c}$, and this does not change anything from Bob's point of view. The corresponding operations are either carried out locally by $\mathcal{S}_B$ or along with $\mathcal{F}_{\mathbf{COM}}$ which, by definition, is fully controlled by $\mathcal{S}_B$.

Hybrid H_2: This hybrid is identical to the previous one, H_1, except that now $\mathcal{S}_{\mathbf{B}}$ generates (a, b), $\boldsymbol{d}$ and $\boldsymbol{s}$ as follows: it starts by randomly generating $a' \leftarrow_\$ \mathbb{Z}_d$, $\boldsymbol{\kappa} \leftarrow_\$ \mathbb{Z}_d^n$ and $a_2, \ldots, a_n, b_2, \ldots, b_n \leftarrow_\$ \mathbb{Z}_d$. Then, it computes b' according to the generated a', the extracted element x and the output y of $\mathcal{F}_{\mathbf{OLE}}$, as $b' = a'x - y$. It then masks a' and b' as

$$a' = \boldsymbol{a} \cdot \boldsymbol{\kappa} \qquad \text{and} \qquad b' = \boldsymbol{b} \cdot \boldsymbol{\kappa},$$

by setting a_1 and b_1 accordingly, i.e. $a_1 = \left(a' - \sum_{i=2}^n a_i \kappa_i\right)/\kappa_1$ and $b_1 = \left(b' - \sum_{i=2}^n b_i \kappa_i\right)/\kappa_1$. Finally, $\mathcal{S}_B$ sends $\boldsymbol{d} = \boldsymbol{a} - \boldsymbol{a}_0$, $\boldsymbol{s} = \boldsymbol{b}_0 + \boldsymbol{a} * \boldsymbol{c} + \boldsymbol{b}$ and $\boldsymbol{\kappa}$ to Bob. This is the last hybrid of the sequence and corresponds to the description of the simulator $\mathcal{S}_B$.

Claim 4. The hybrids H_1 and H_2 are indistinguishable.

Proof. Since, in its first two steps, $\mathcal{S}_B$ executes a RWOLE scheme, according to Lemma 10 we have that $\mathcal{S}_B$ is $\epsilon-$close to a situation where Bob's knowledge on the vectors $(\boldsymbol{a}_0, \boldsymbol{b}_0)$ is lower-bounded by the value

$$\frac{1}{n}\lambda(\zeta) = \frac{\log d}{2}(1 - h_d(\zeta))$$

for $\zeta \in]0, 1 - \frac{1}{d}]$, n the security parameter and $\epsilon(\zeta, n) = \exp\left(-\frac{2\zeta^2 t^2 n^2}{(nt+1)(t+1)}\right)$. Also, as Bob receives $\boldsymbol{d}$ and $\boldsymbol{s}$, according to Lemma 11 his knowledge on $(\boldsymbol{a}, \boldsymbol{b})$ is also lower-bounded by the same $\lambda(\zeta)/n$. Furthermore, since $\mathcal{S}_B$ defines $\boldsymbol{a}$ such that $a' = \boldsymbol{a} \cdot \boldsymbol{\kappa}$, from Lemma 12 we can conclude that $\mathcal{S}_B$ is $(\xi + \epsilon)-$close to a scenario where a' is uniformly distributed. This comes from the properties in Lemma 12 and the triangle inequality:

$$\begin{aligned} \delta(\tau_{\mathbb{Z}_d} \otimes \sigma_{B'},\, \rho_{F_a B'}) &\leq \delta(\tau_{\mathbb{Z}_d} \otimes \sigma_{B'},\, \sigma_{F_a B'}) + \delta(\sigma_{F_a B'},\, \rho_{F_a B'}) \\ &\leq K\, 2^{-n\, f_d(\zeta)} + e^{-\frac{2\zeta^2 t^2 n^2}{(nt+1)(t+1)}} = \xi + \epsilon \end{aligned}$$

where $K = \frac{\sqrt{d}}{2}$, $f_d(\zeta) = \frac{\log d}{4}(1 - h_d(\zeta))$. This means that the triple $(\boldsymbol{d}, \boldsymbol{s}, \boldsymbol{\kappa})$ only gives to the environment a negligible advantage in distinguishing between the real and ideal world executions. $\square$

$\square$

5.6 Protocol Generalizations

5.6.1 QOLE in Galois fields of prime-power dimensions

So far, we have been working in Hilbert spaces of prime dimensions; this reflects the fact that, for prime d, $\mathbb{Z}_d$ is a field and, under a well-defined set of MUBs $\{|e_r^x\rangle\}_{r\in\mathbb{Z}_d}$, $\forall x \in \mathbb{Z}_d$, we have the affine relation (5.4):

$$V_a^b \, |e_r^x\rangle = c_{a,b,x,r} \left| e^x_{ax-b+r} \right\rangle.$$

In this section, we generalize our protocol, $\pi_{\mathbf{QOLE}}$, to Hilbert spaces of prime-power dimensions, $N = d^M$ (d prime and $M > 1$), taking advantage of the fact that in a Galois field of dimension d^M, $GF(d^M)$, we can build a complete set of $N + 1$ MUBs [35].

Succinctly, in $GF(d^M)$, we identify the integers $i \in \mathbb{Z}_N$ with their $d-$ary representation, i.e.

$$\mathbb{Z}_N \ni i = \sum_{n=0}^{M-1} i_n d^n \longleftrightarrow (i_0, \ldots, i_{M-1}) \in GF(d^M).$$

In these fields there are two operations, addition and multiplication, which we denote by $\oplus$ and $\odot$, respectively. Addition is straightforward, as it is given by the component-wise addition modulo d of elements, i.e. $i \oplus j = (i_0 + j_0 \mod d, \ldots, i_{M-1} + j_{M-1} \mod d)$. Considering $i = \sum_{n=0}^{M-1} i_n d^n$ as a polynomial of degree $M-1$ given by $i(p) = \sum_{n=0}^{M-1} i_n p^n$, multiplication between two elements i, j, is given by the multiplication between the corresponding polynomials $i(p)$ and $j(p)$ modulo some irreducible polynomial $m(p)$, i.e. $i \odot j = \big(i(p) \times j(p)\big) \mod m(p)$.

Analogously to prime-dimension fields, we can write the operators V_a^b in the computational basis, as

$$V_a^b = \sum_{k=0}^{N-1} |k \oplus a\rangle\, \omega^{(k\oplus a)\odot b} \langle k| \,,$$

and the eigenstates for the corresponding $N+1$ pairwise MUBs, as

$$|e_r^x\rangle = \frac{1}{\sqrt{N}} \sum_{l=0}^{N-1} |l\rangle\, \omega^{\ominus(r\odot l)} \alpha_{\ominus l}^{x*},$$

where $\alpha_{\ominus l}^{x}$ is a phase factor whose form depends on whether d is even or odd. For details, see Section 2.4.2 in [35].

Given the above, we can derive (see **??**) the following affine relation similar to (5.4):

$$V_a^b \left|e_r^i\right\rangle = \omega^{r\odot a} \alpha_a^{i*} \left|e^i_{i\odot a \ominus b \oplus r}\right\rangle . \tag{5.12}$$

Notice that all the steps in the $\pi_{\mathbf{QOLE}}$ depend on the properties of the field operations (addition and multiplication) and on the fact that expression (5.4) holds. Hence, we can use $\pi_{\mathbf{QOLE}}$ adapted for the operations $\oplus$ and $\odot$, in order to quantum-UC-realize $\mathcal{F}_{\mathbf{OLE}}$ in fields of prime-power dimension d^M.

5.6.2 Quantum Vector OLE

In the proposed protocol $\pi_{\mathbf{QOLE}}$, we extract one instance of OLE out of n instances of WOLE. As far as efficiency is concerned, it would be desirable to generate more instances of OLE out of those n instances of WOLE. Here, we show how to use WOLE as a resource to realize the VOLE functionality, $\mathcal{F}^k_{\mathbf{VOLE}}$, presented in Figure **??**. In this case, Alice fixes a k (which is specified later), defines a set of k linear functions $(\boldsymbol{a}, \boldsymbol{b}) \in \mathbb{F}_q^k \times \mathbb{F}_q^k$ and Bob outputs the evaluation of all these functions on a specified element $x \in \mathbb{F}_q$ that he chooses, i.e. $\boldsymbol{f} := \boldsymbol{a}x + \boldsymbol{b}$. Since $\pi_{\mathbf{QOLE}}$ can be extended to finite fields $\mathbb{F}_q$, where q is a prime or prime-power number (see Section 5.6.1), the $\mathcal{F}^k_{\mathbf{VOLE}}$ functionality can also be defined in $\mathbb{F}_q$.

In the extraction phase of $\pi_{\mathbf{QOLE}}$, Alice randomly chooses a function $g_{\boldsymbol{\kappa}}$ and applies it to the pair $(\boldsymbol{a}, \boldsymbol{b})$. This procedure suggests that, in order to generate different input elements (a', b'), Alice can randomly choose another function $g_{\boldsymbol{\kappa}'}$ and set $a' = g_{\boldsymbol{\kappa}'}(\boldsymbol{a})$ and $b' = g_{\boldsymbol{\kappa}'}(\boldsymbol{b})$. This is equivalent to generating a random $2 \times n$ matrix in $\mathbb{F}_q$, i.e.

$$\begin{bmatrix} — & \boldsymbol{\kappa} & — \\ — & \boldsymbol{\kappa}' & — \end{bmatrix} \begin{bmatrix} | & | \\ \boldsymbol{a} & \boldsymbol{b} \\ | & | \end{bmatrix} = \begin{bmatrix} a & b \\ a' & b' \end{bmatrix}.$$

However, in case $\boldsymbol{\kappa}$ and $\boldsymbol{\kappa}'$ are linearly dependent (i.e. $\boldsymbol{\kappa} = c\boldsymbol{\kappa}'$ for some $c \in \mathbb{Z}_d$), Bob would have some extra information about Alice's elements (a, b) and (a', b'), as $(a, b) = c(a', b')$. This leads to a situation beyond the $\mathcal{F}^k_{\mathbf{VOLE}}$ definition. To avoid this issue, let us consider the set of $k \times n$ matrices with rank k over $\mathbb{F}_q$ for $1 \leq k \leq n$, and denote it by $\mathcal{R}_{k\times n}(\mathbb{F}_q)$. For a binary finite field, $\mathcal{R}_{k\times n}(\mathbb{F}_2)$ is a two-universal hash family [**? ?**]. Similarly, one can prove that the more general set $\mathcal{R}_{k\times n}(\mathbb{F}_q)$ is also a two-universal hash family from $\mathbb{F}_q^n$ to $\mathbb{F}_q^k$. During the extraction phase of the original $\pi_{\mathbf{QOLE}}$, Alice chooses vectors $(\boldsymbol{a}, \boldsymbol{b})$ according to the random vector $\boldsymbol{\kappa}$ and the desired final elements (a, b) (see step 9 in Figure 5.5). In that case, since there is only one random vector $\boldsymbol{\kappa}$, there are $n - 1$ undefined variables for each vector $\boldsymbol{a}$ and $\boldsymbol{b}$, i.e. $a_2, \ldots, a_n$ and $b_2, \ldots, b_n$ that can be chosen freely. For the VOLE protocol, instead of choosing just one vector $\boldsymbol{\kappa}$, Alice randomly chooses a matrix $\mathcal{K} \in \mathcal{R}_{k\times n}(\mathbb{F}_q)$ of rank k. She then defines vectors $(\boldsymbol{a}', \boldsymbol{b}') \in \mathbb{F}_q^n \times \mathbb{F}_q^n$ consistent with the final elements $(\boldsymbol{a}, \boldsymbol{b}) \in \mathbb{F}_q^k \times \mathbb{F}_q^k$. That is, Alice has the following system:

$$\begin{bmatrix} — & \boldsymbol{\kappa}_1 & — \\ & \cdots & \\ — & \boldsymbol{\kappa}_k & — \end{bmatrix} \begin{bmatrix} | & | \\ \boldsymbol{a}' & \boldsymbol{b}' \\ | & | \end{bmatrix} = \begin{bmatrix} | & | \\ \boldsymbol{a} & \boldsymbol{b} \\ | & | \end{bmatrix},$$

that can be solved by means of the Gaussian elimination method. Since $\mathcal{K} \in \mathcal{R}_{k\times n}(\mathbb{F}_q)$, there will be $n - k$ undefined variables in both vectors $\boldsymbol{a}'$ and $\boldsymbol{b}'$. Let U denote the set of undefined indexes in $\boldsymbol{a}'$ and $\boldsymbol{b}'$. Alice randomly chooses a'_i and b'_i for $i \in U$ and solves the above equation system. Then, they proceed similarly to the original $\pi_{\mathbf{QOLE}}$ and execute the derandomization protocol $\pi^n_{\mathbf{WOLE}}((\boldsymbol{a}', \boldsymbol{b}'), \boldsymbol{x})$. Finally, Bob applies Alice's chosen matrix $\mathcal{K}$ to his output vector $\boldsymbol{y}'$ to get the final element $\boldsymbol{y}$. This vectorized extraction protocol $\pi_{\mathbf{VEXT}}$ is presented in Figure 5.8.

Protocol $\pi_{\mathbf{VEXT}}$

Alice's input: $(\boldsymbol{a}, \boldsymbol{b}) \in \mathbb{F}_q^k \times \mathbb{F}_q^k$
Bob's input: $x \in \mathbb{F}_q$

1. Alice chooses randomly a matrix $\mathcal{K} \in \mathcal{R}_{k \times n}(\mathbb{F}_q)$ and sends it to Bob.
2. Using the Gaussian elimination method, Alice finds one solution of the system:

$$\mathcal{K} \begin{bmatrix} | & | \\ \boldsymbol{a}' & \boldsymbol{b}' \\ | & | \end{bmatrix} = \begin{bmatrix} | & | \\ \boldsymbol{a} & \boldsymbol{b} \\ | & | \end{bmatrix}$$

   (a) Alice finds the set U of undefined indexes in $\boldsymbol{a}'$ and $\boldsymbol{b}'$.
   (b) Alice randomly generates $a_i', b_i' \leftarrow_\$ \mathbb{F}_q$ for $i \in U$.
   (c) Alice solves the system for indexes $i \notin U$.

3. Alice and Bob run $\pi^n_{\mathbf{WOLE}}((\boldsymbol{a}', \boldsymbol{b}'), \boldsymbol{x})$, where $\boldsymbol{x} = (x, \dots, x)$. Bob outputs $\boldsymbol{y}' \in \mathbb{F}_q^n$.
4. Bob computes $\boldsymbol{y} = \mathcal{K}\boldsymbol{y}'$.

Alice's output: $\perp$
Bob's output: $\boldsymbol{y} \in \mathbb{F}_q^k$

Figure 5.8: Extraction protocol for VOLE.

The correctness of the protocol is drawn immediately from linearity:

$$\boldsymbol{y} = \mathcal{K}\boldsymbol{y}' = \mathcal{K}(\boldsymbol{a}'x + \boldsymbol{b}') = \boldsymbol{a}x + \boldsymbol{b}.$$

The security of the protocol is constrained by the closeness parameter

$$\xi = \frac{1}{2}\sqrt{2^{k \log q - H_{\min}(X|E)}}$$

given by Lemma 9, where we consider l to be k and d to be q. As before, $\mathbf{F}_{\boldsymbol{a}'}$ denotes the distribution of the $\pi^n_{\mathbf{WOLE}}$ protocol's input $\boldsymbol{a}'$ from Bob's perspective. From Lemma 9, since $\mathcal{R}_{k\times n}(\mathbb{F}_q)$ is a two-universal family of hash functions, we know that $\mathcal{K} \in \mathcal{R}_{k\times n}(\mathbb{F}_q)$ approximates $\mathcal{K}\mathbf{F}_{\boldsymbol{a}'} = \mathbf{F}_{\boldsymbol{a}}$ to uniform conditioned on Bob's side information. However, the closeness parameter has to be negligible in the security parameter n, thus setting a bound on k (the size of VOLE), i.e. for $\eta > 0$,

$$\begin{aligned} k \log q - \frac{n \log q}{2}\big(1 - h_q(\zeta)\big) &< -n\eta \log q \\ k &< n\Big(\frac{1}{2}\big(1 - h_q(\zeta)\big) - \eta\Big). \end{aligned}$$

Since $k > 0$, we have that $0 < \eta < \frac{1}{2}\big(1 - h_q(\zeta)\big)$. This gives a bound on the proportion of elements that we can extract from n WOLEs and shows how Alice can fix k in the beginning. Note that this bound is not necessarily optimal, and one could try to improve it. We leave this as future work, as it goes beyond the scope of this paper, which is to introduce a quantum protocol for OLE that can be, in turn, adapted accordingly to also achieve VOLE.

Let us denote by $\pi_{\mathbf{QVOLE}}$ the protocol $\pi_{\mathbf{QOLE}}$ with the subprotocol $\pi_{\mathbf{VEXT}}$ instead of $\pi_{\mathbf{EXT}}$. For the security of $\pi_{\mathbf{QVOLE}}$, we have:

Theorem 5 (quantum-UC security of $\pi_{\mathbf{QVOLE}}$)**.** *The protocol $\pi_{\boldsymbol{QOVLE}}$ statistically quantum-UC realizes (see Definition 6) $\mathcal{F}^k_{\boldsymbol{VOLE}}$ in the $\mathcal{F}_{\boldsymbol{COM}}-$hybrid model.*

The proof is much the same as the proof of Theorem 2, therefore we omit it.

5.7 Outlook and further work

OLE is an important primitive for secure two-party computation, and while for stronger primitives such as bit commitment, OT and coin flipping there is a plethora of both theoretical as well as concrete protocol proposals [39, 124**? ? ? ? ? ? ? ? ? ? ? ? ? ? ?**], up until now, there was no OLE protocol based on quantum communication. In this work, we present two protocols for QOLE. The first protocol is secure against semi-honest adversaries in the static corruption setting. The second proposed protocol, $\pi_{\mathbf{QOLE}}$, builds upon the semi-honest version and extends it to the dishonest case, following a commit-and-open approach. We prove this second protocol to be secure in the quantum-UC framework when assuming ideal commitments, making it possible to be composed in any arbitrary way. We also constructed two generalizations of our protocol: the first achieves QOLE in Galois fields of prime-power dimensions and the second is a protocol for quantum vector OLE. Note that our protocol achieves everlasting security, i.e. it remains information-theoretically secure after its execution, even if the dishonest party becomes more powerful in the future.

While the security of $\pi_{\mathbf{QOLE}}$ is thoroughly analyzed, this is done in the noiseless case. Proving security assuming the existence of noise should follow a similar reasoning (see also [29]). Indeed, in the presence of noise, in Step 4 in the quantum phase of $\pi_{\mathbf{QOLE}}$

(Figure 5.5), Alice should abort the protocol if the errors measured (err) exceed some predetermined value ν, that is assumed to be due to noise. This way, the error parameter is $err = \nu + \zeta'$, where ζ' accounts for the activity of a dishonest Bob. Naturally, this will decrease the bound on the min-entropy of Alice's functions $\mathbf{F}$ given Bob's side information, i.e.

$$H_{\min}(\mathbf{F}|\mathbf{Y}E)_{\sigma_{\mathbf{FY}E}} \geq \frac{n\log d}{2}\left(1 - h_d(\nu + \zeta' + \zeta)\right).$$

Although it is possible to generalize the security results to the setting of noisy quantum communication, it is not guaranteed that the proposed protocol retains correct. Therefore, as future work, it would be useful to propose specific implementations where noise is taken into account and its effect on the security properties is studied (see also [**?**]).

Following a different assumption model, the $\pi_{\mathbf{QOLE}}$ protocol can be easily adapted to remain secure in the bounded-quantum-storage model. In this adapted version, the test phase of $\pi^n_{\mathbf{RWOLE}}$ is simply substituted by a waiting time Δt. This ensures that Bob is only able to store a noisy or limited amount of qudits. It would be interesting to explore how different noisy channels affect the security properties. Also, to guarantee the composability of the protocol, an analysis in the bounded-quantum-storage-UC model put forth by Unruh [150] can be performed.

Our protocol is a two-way protocol, i.e. Bob prepares and sends a quantum state, Alice applies some operation to it, and sends it back to Bob who measures the final state. For QOT there exist several proposals for two-way protocols [**? ? ? ? ?**], and, in particular, the one presented by Amiri et al. [**?**] also demonstrates their experimental feasibility. This is further motivation to work on developing realistic practical implementations of our protocol. Furthermore, one could increase the security standards by making our protocol device-independent. The proposal of Kundu et al. [**?**], who extended the work of Chailloux et al. [**?**], could serve as an inspiration. And while the aforementioned works focus on two-way QOT protocols, recently, non-interactive or one-way protocols have also been proposed for device-independent [**?**] and XOR QOT [**?**].

Finally, based on our results, one could construct quantum protocols for oblivious polynomial evaluation, which – as mentioned in the Introduction – is another important primitive facilitating various applications.

Chapter 6

Private phylogenetic trees

Several privacy-enhancing technologies (PET) (differential privacy [11], homomorphic encryption [12] and secure multiparty computation) have already been applied to biomedical data analysis [202–206]. In particular, these classical techniques have been used in the context of genomic private data analysis. As a way to push research and innovation forward, there have been several competitions [207] focused on developing faster and more secure solutions in the field of genomic analysis. Also, in recent surveys [208, 209], the authors describe the role of PETs in four different computational domains of the genomic's field (genomic aggregation, GWASs and statistical analysis, sequence comparison and genetic testing). However, these surveys do not provide any reference covering privacy-preserving methods applied to phylogeny inference.

In contrast to classical technologies, the usage of quantum cryptographic technologies in private computation has not been widely reported. Chan et al. [210] developed real-world private database queries assisted with quantum technologies and in [211] the authors simply suggest that their implementation of quantum OT is suitable to be applied in an SMC environment. Despite its little integration with PETs, quantum cryptographic technologies have already reached a maturity level that enables this integration. Quantum key distribution (QKD) and quantum random number generators (QRNG) are currently being commercialized and applied to critical use cases (e.g. Governmental data storage and communications, Data centres [212]) with in-field deployment (e.g. OpenQKD, https://openqkd.eu/). The quantum oblivious key distribution (QOKD) protocol is based on the same technology as QKD and QRNG, benefiting from its development and allowing to generate the necessary resources to execute OT [124, 133, 134].

In this chapter, we present a feasible modular private phylogenetic tree protocol that leverages quantum communications. It provides enhanced security against quantum computer attacks and decreases the complexity of the computation phase when compared to a state-of-the-art classical-only system. The system is built on top of Libscapi [213] implementation of Yao protocol and PHYLIP phylogeny package [183]. It integrates three crucial quantum primitives: quantum oblivious transfer, quantum key distribution and quantum random number generator.

This chapter follows a top-down approach. In section 6.1, we start by explaining the concept of phylogenetic trees and the distance-based algorithms used to generate these trees. In section 6.2, we set down the security definitions that will be used to analyse and prove the system's security. In sections 6.3 and 6.4, we describe the quantum cryptographic tools and the software tools that are integrated into the protocol, respectively. In section 6.5, we describe the proposed SMC system for phylogenetic trees. In section 6.6 we explain how the quantum cryptographic tools are integrated into the system. Section 6.7 is devoted to the theoretical security analysis of the protocol and in section 6.8 we perform a complexity analysis. In the last section we present a performance comparison of the system between a classical-only and a quantum-assisted implementation.

6.1 Phylogenetic trees

Phylogenetic trees are diagrams that depict the evolutionary ties between groups of organisms [214] and are composed of several nodes and branches. The nodes represent genome sequences and each branch connects two nodes. It is important to note that the terminal nodes (also called leaves) represent known data sequences, whether internal nodes are ancestral sequences inferred from the known sequences [215, 216]. The length of the branches connecting two nodes represents the number of substitutions that have occurred between them. However, this quantity must be estimated because it cannot be computed directly using the sequences. In fact, by simply counting the number of sites where two nodes have different base elements (Hamming distance), we underestimate the number of substitutions that have occurred between them.

The best way to compute a correct phylogenetic tree depends on the type of species and sequences under analysis and the assumptions made by the sequences substitution model. By a correct tree, we mean a tree that depicts as approximate as possible the real phylogeny of the sequences, i.e. the real ties between known sequences and inferred ancestors. These assumptions lead to different algorithms which can be divided into two categories:

1. Distance-based methods: they base their analysis on the evolutionary distance matrix which contains the evolutionary distances between every pair of sequences. The evolutionary distance used also depends on the substitution model considered. These methods are computationally less expensive when compared to character-based methods.

2. Character-based methods: they base their analysis on comparing every site (character) of the known data sequences and do not reduce the comparison of sequences to a single value (evolutionary distance).

We only consider the distance-based algorithms that are part of the PHYLIP [217] distance matrix models, namely: Fitch-Margoliash (`fitch` and `kitsch`), Neighbour Joining (`neighbor`) and UPGMA (`neighbor`). Also, we only consider the evolutionary distances developed in PHYLIP `dnadist` program: Jukes-Cantor (JC) [218], Kimura 2-parameter (K2P) [219], F84 [220] and LogDet [221]. We refer interested readers on this topic to some textbooks about phylogenetic analysis [215, 216].

Next, we give an overview of these distance-based methods to build some intuition on how to tailor them to a private setting. We start by looking at the different evolutionary distances and then at the distance-based algorithms.

6.1.1 Evolutionary distances

The evolutionary distance depends on the number of estimated substitutions between two sequences, which is governed by the substitution model used. So, before defining a suitable distance, it is important to have a model that describes the substitution probability of each nucleotide across the sequence at a given time.

The distances considered in this work can be divided into two groups by their assumptions. JC, K2P and F84 assume that the substitution probabilities remain constant throughout the tree, (i.e. stationary probabilities), whether the LogDet distance assumes that the probabilities are not stationary.

Also, the first three evolutionary distances (JC, K2P and F84) assume an evolutionary model that can be described by a *time-homogeneous stationary Markov* process. This Markov process is based on a probability matrix $\mathbf{P}(t)$ that defines the transition probabilities from one state to the other after a certain time period t. It can be shown [222] that this probability is given by

$$\mathbf{P}(t) = e^{\mathbf{Q}t} \tag{6.1}$$

where the rate matrix $\mathbf{Q}$ is of the form given by (6.2).

$$\mathbf{Q} = \begin{pmatrix} -\mu(a\pi_C + b\pi_G + c\pi_T) & a\mu\pi_C & b\mu\pi_G & c\mu\pi_T \\ g\mu\pi_A & -\mu(g\pi_A + d\pi_G + c\pi_T) & d\mu\pi_G & e\mu\pi_T \\ h\mu\pi_A & i\mu\pi_C & -\mu(h\pi_A + j\pi_C + f\pi_T) & f\mu\pi_T \\ j\mu\pi_A & k\mu\pi_C & l\mu\pi_G & -\mu(i\pi_A + k\pi_C + l\pi_G) \end{pmatrix} \tag{6.2}$$

In $\mathbf{Q}$, each entry $\mathbf{Q}_{ij}$ represents the substitution rate from nucleotide i to j and both its columns and rows follow the order A, C, G, T. μ is the total number of substitutions per unit time and we can define the evolutionary distance, d, to be given by $d = \mu t$. The parameters $a, b, c, ..., l$ represent the relative rate of each nucleotide substitution to any other. Finally, $\pi_A, \pi_C, \pi_G, \pi_T$ describe the frequency of each nucleotide in the sequences.

From expression (6.1), it is possible to define a likelihood function on the distance d and use the maximum likelihood approach to get an estimation of the evolutionary distance. The likelihood function defines the probability of observing two particular sequences, x and y, given the distance d:

$$L(d) = \prod_{i=1}^{n} \pi_{x_i} P_{x_i,y_i}\left(\frac{d}{\mu}\right)$$

The parameters of $\mathbf{Q}$ are defined differently depending on the evolutionary model used and the maximum likelihood solution leads to different evolutionary distances.

Jukes-Cantor

The Jukes-Cantor model [218] is the simplest possible model based on $\mathbf{Q}$ as given in (6.2). It assumes the frequencies of the nucleotide to be the same, i.e. $\pi_A = \pi_C = \pi_G = \pi_T = \frac{1}{4}$ and sets the relative rates $a = b = ... = l = 1$. This model renders an evolutionary distance between two sequences x and y given by:

$$d_{xy} = -\frac{3}{4} \ln\left(1 - \frac{4}{3}\frac{h_{xy}}{n}\right) \tag{6.3}$$

where h_{xy} is the uncorrected hamming distance and n the length of the sequences.

Kimura 2-parameter

This model [219] distinguishes between two different nucleotide mutations:

1. Type I (transition): $A \leftrightarrow G$, i.e. from purine to purine, or $C \leftrightarrow T$, i.e. from pyrimidine to pyrimidine.

2. Type II (transversion): from purine to pyrimidine or vice versa.

These two different types of transformation lead to different probability distributions denoted by P and Q, where P is the probability of homologous sites showing a type I difference, while Q is that of these sites showing a type II difference. So, the Kimura [219] metric between x and y is given by the following:

$$d_{xy} = -\frac{1}{2}\ln\left(\left(1-2P-Q\right)\sqrt{1-2Q}\right) \tag{6.4}$$

where $P = \frac{n_1}{n}$, $Q = \frac{n_2}{n}$ and n_1 and n_2 are respectively the number of sites for which two sequences differ from each other with respect to type I ("transition" type) and type II ("transversion" type) substitutions.

F84

This model [220] also distinguishes different nucleotide transitions but do not assume the nucleotide frequencies to be the same. This leads to a more general distance which can be estimated in closed form:

$$d_{xy} = -2A\ln\left(1-\frac{P}{2A}-\frac{(A-B)Q}{2AC}\right)+2(A-B-C)\ln\left(1-\frac{Q}{2C}\right) \tag{6.5}$$

where $A = \frac{\pi_C\pi_T}{\pi_Y} + \frac{\pi_A\pi_G}{\pi_R}$, $B = \pi_C\pi_T + \pi_A\pi_G$ and $C = \pi_R\pi_Y$ for $\pi_Y = \pi_C + \pi_T$ and $\pi_R = \pi_A + \pi_G$, and P and Q are defined as in the Kimura 2-parameter model above.

Although more complex models can be considered with different combinations of parameters in $\mathbf{Q}$, not all of them produce a distance function that can be estimated in closed form.

LogDet

As mentioned before, the models based on matrix $\mathbf{Q}$ assume that the probability matrix $\mathbf{P}(t)$ is stationary, i.e. remains constant throughout the tree. However, there are evolutionary scenarios where this assumption does not give a correct description of reality. The LogDet evolutionary distance [221] suits a wider set of models and considers the case where $\mathbf{P}(t)$ is different at each branch in the tree. This is given by

$$d_{xy} = -\frac{1}{4}\ln\left(\frac{\det F_{xy}}{\sqrt{\det \prod_x \prod_y}}\right) \tag{6.6}$$

where the divergence matrix F_{xy} is a 4×4 matrix such that the $ij-$th entry gives the proportion of sites with nucleotide i in sequence x and j in sequence y. Also, $\prod_x$ and

$\prod_y$ are diagonal matrices where its $i-$th component correspond to the proportion of i nucleotide in the sequence x and y, respectively.

6.1.2 Distance-based algorithms

All distance-based methods make use of evolutionary distances to compare different genomic sequences. Although it may lead to less accurate phylogenetic trees, these methods are highly popular among researchers who have to handle large number of sequences. All methods assume the following:

1. The evolutionary distance computed between each pair is independent of all other sequences;
2. The estimated distance between each pair of sequences is given by the sum of the size of the branches that connect both of them.

These algorithms are thus divided into two phase:

1. Distance computation phase: all the pairwise evolutionary distances are computed according to the selected model. This step is common to all distance-based methods;
2. Iterative clustering: aggregate the sequences in clusters iteratively. This step is specific to each method.

Let us briefly describe three of the most common distance-based methods [215].

UPGMA

The Unweighted Pair Group Method with Arithmetic mean (UPGMA) method produces a rooted phylogenetic tree and assumes the data to be ultrametric, i.e. assumes that

$$d_{xy} \leq \max(d_{xz}, d_{yz})$$

for sequences x, y and z. These two assumptions imply that all the sequences are equidistant to the inferred root sequence.

It starts by considering every sequence as a single-valued cluster. Then, it goes on merging the clusters according to the smallest difference between them and recomputes the distance matrix through a simple average of distances. In summary, we have the following steps:

1. Merge clusters, $C_i = \{c_i\}$ and $C_j = \{c_j\}$ for sets c_i and c_j, with the smallest distance present in the distance matrix, i.e. $d_{i,j} \leq d_{k,l}\, \forall k, l$. Create a new cluster $C_{i/j} = \{\{c_i, c_j\}\}$. This new cluster represents a branch between clusters C_i and C_j;

2. Recompute the distance matrix according to the following formula:

$$d_{i/j,l} = \frac{d_{i,l} + d_{j,l}}{2}$$

for all other clusters l;

3. Eliminate clusters C_i and C_j from the distance matrix and add cluster $C_{i/j}$ with the distances computed as in the previous step;

4. Repeat steps $1 - 3$ until there is only one cluster left.

Neighbour-Joining

As we have seen, the UPGMA joins the clusters with the minimum distance between them. Now, the Neighbour-Joining method considers not only how close two clusters are, but it also considers how far these two clusters are from the others. Thus, the clusters to be merged should minimize the following quantity:

$$q(C_i, C_j) = (r-2)d(C_i, C_j) - u(C_i) - u(C_j)$$

where r is the number of clusters in the current iteration and $u(C_i) = \sum_j d(C_i, C_j)$.

As opposed to the UPGMA algorithm, this method produces an unrooted tree and it can be summarized in the following steps:

1. Consider every sequence as a single-valued cluster and connect it to a central point;

2. Compute a matrix $\mathcal{Q}$ where its entries are given by the quantity above, i.e. $\mathcal{Q}_{ij} = q(C_i, C_j)$;

3. Identify clusters C_i and C_j with the smallest value in the matrix $\mathcal{Q}$. Create a new node $C_{i,j}$ and join both clusters C_i and C_j to it;

4. Assign to the branch $C_iC_{i/j}$ a distance given by:

$$\frac{1}{2}d(C_i, C_j) - \frac{1}{2}\frac{(u_i - u_j)}{r-2}$$

and to the branch $C_jC_{i/j}$ a distance given by:

$$\frac{1}{2}d(C_i, C_j) - \frac{1}{2}\frac{(u_j - u_i)}{r-2};$$

5. Eliminate clusters C_i and C_j from the distance matrix and add cluster $C_{i/j}$ with the distances to the other clusters computed as follows:

$$d(C_l, C_{i/j}) = \frac{1}{2}(d(C_l, C_i) + d(C_l, C_j) - d(C_i, C_j))$$

for all other nodes C_l;

6. Repeat steps $2 - 5$ until there is only one cluster left.

Fitch-Margoliash

This method renders an unrooted tree and also assumes that the distances are additive. It analyses iteratively three-leaf trees and computes the distance between three known nodes and one created internal node. This is based on the following observation. Given three clusters C_i, C_j and C_l, and one internal node a that is connected to all these three clusters, the distances between the clusters are given by:

$$\begin{aligned} d(C_i, C_j) &= d(C_i, a) + d(a, C_j) \\ d(C_i, C_l) &= d(C_i, a) + d(a, C_l) \\ d(C_l, C_j) &= d(C_l, a) + d(a, C_j) \end{aligned}$$

from which we can easily see that

$$\begin{aligned} d(a, C_i) &= \tfrac{1}{2}\Big(d(C_i, C_j) + d(C_i, C_l) - d(C_l, C_j)\Big) \\ d(a, C_j) &= \tfrac{1}{2}\Big(d(C_i, C_j) + d(C_l, C_j) - d(C_i, C_l)\Big) \\ d(a, C_l) &= \tfrac{1}{2}\Big(d(C_i, C_l) + d(C_l, C_j) - d(C_i, C_j)\Big) \end{aligned} \tag{6.7}$$

Thus, we can estimate the distances from the known clusters to the new internal node using the distances between the clusters as given in (6.7). Based on this, the Fitch-Margoliash algorithm goes as follows:

1. Consider every sequence as a single-valued cluster;

2. Identify the two clusters, C_i and C_j, with the smallest distance in the distance matrix;

3. Consider all the other clusters as a single cluster C_l and recompute the distance matrix with just three clusters. The distances between the identified clusters and the new cluster is given by an average value of the distances between the identified clusters and the elements inside the cluster C_l, i.e.

$$d(C_i, C_l) = \frac{1}{|C_l|} \sum_{c \in C_l} d(C_i, c)$$

and similarly for C_j;

4. Using expressions (6.7), we compute the distances from the three clusters and the central node;

5. Merge clusters, C_i and C_j, into a new one $C_{i/j}$ and recompute the distance matrix between $C_{i/j}$ and all the other clusters $c \in C_l$ by a simple average expression:

$$d(c, C_{i/j}) = \frac{d(c, C_i) + d(c, C_j)}{2};$$

6. Repeat steps $2 - 4$ until there is only one cluster left.

All these methods output a tree with some topology, $\mathcal{T}$ along with the distances between the branches.

6.2 Security definition

In this chapter, we consider a multiparty computation scenario that is secure against *semi-honest* parties. This means that all the parties strictly follow the protocol but can use their inputs, received messages and outputs to deduce any additional information. As such, these are also commonly called *honest-but-curious* parties. Nevertheless, we can extend the protocol to the malicious setting, by simply implementing a two-party secure computation protocol that is secure against malicious adversaries [69]. Our security will follow the simulation paradigm and we start with the definition of security in a multi-party setting. The formal definition is taken from [69].

Notation.

- $\mathcal{F}$ denotes the ideal functionality to be computed in the SMC session, i.e. $\mathcal{F} : \mathcal{X}^n \to \mathcal{Y}^n$ where n is the number of parties participating in the SMC and $\mathcal{X}$ and $\mathcal{Y}$ are the input and output space of each party, respectively. $X^i \in \mathcal{X}$ and $Y^i \in \mathcal{Y}$ denote the sets of input and output of party P^i, respectively. Also, for short, $X = (X^1, ..., X^n)$ and $Y = (Y^1, ..., Y^n)$;

- π denotes the protocol that implements the ideal functionality $\mathcal{F}$;
- C is the set of corrupted parties;
- $\texttt{view}^i_\pi(X) := (X^i, r^i; m^i_1, ..., m^i_t)$. This tuple is called the view of party P^i and it contains its inputs (X^i), its random-tape value (r^i) and the messages m^i_j received during the SMC execution;
- $\texttt{output}_\pi(X) = (\texttt{output}^1_\pi(X), ..., \texttt{output}^n_\pi(X))$, where $\texttt{output}^i_\pi(X)$ is the output of party i computed from its view $\texttt{view}^i_\pi(X)$;
- Sim is a probabilistic polynomial-time simulator in the ideal-world;
- The distribution on inputs X given by a real-world execution of the protocol π:

$$\texttt{Real}_\pi(C; X) := \big\{\{\texttt{view}^i_\pi(X) : i \in C\}, \texttt{output}_\pi(X)\big\}_X.$$

- The distribution on inputs X given by the ideal-world simulation of the parties' view:

$$\texttt{Ideal}_{Sim,\mathcal{F}}(C; X) := \big\{Sim\big(\{(X^i, \mathcal{F}(X^i)) : i \in C\}\big), \mathcal{F}(X)\big\}_X.$$

Definition 10 (Semi-honest security). *A protocol securely realizes $\mathcal{F}$ in the presence of semi-honest adversaries if there exists a simulator Sim such that, for every subset of corrupted parties C and all inputs X, we have*

$$\texttt{Real}_\pi(C; X) \stackrel{c}{\equiv} \texttt{Ideal}_{Sim,\mathcal{F}}(C; X), \tag{6.8}$$

where $\stackrel{c}{\equiv}$ denotes computational indistinguishability.

This definition conveys the notion that whatever can be computed by a party during the execution of the protocol is only based on his inputs and outputs, i.e. the execution of the protocol do not provide any further information. This is equivalent to expression (6.8), which states that the distribution of the view and outputs in a real-world execution is computationally indistinguishable from the distribution generated by a simulator and the functionality output. It is also worth noting that, as it is proved in [223], for deterministic $\mathcal{F}$ we have that definition III.1 is equivalent to the simpler case where the `Real` and `Ideal` distributions do not take into account the output of the real protocol execution and the output of the functionality, respectively, i.e.

$$\texttt{Real}_\pi(C; X) = \{\texttt{view}^i_\pi(X) : i \in C\}_X$$

and

$$\texttt{Ideal}_{Sim,\mathcal{F}}(C;X) = \big\{Sim\big(\{(X^i, \mathcal{F}(X^i)) : i \in C\}\big)\big\}_X.$$

Therefore, we just need to build a simulator that satisfies expression (6.8) for the $\texttt{Real}_\pi(C;X)$ and $\texttt{Ideal}_{Sim,\mathcal{F}}(C;X)$ given as above in order to prove security.

6.2.1 Distance matrix functionality

For our private phylogenetic tree problem, the ideal functionality $\mathcal{F}$ outputs the distance matrix according to the selected evolution model (Jukes-Cantor, Kimura 2-parameter, F84 or LogDet). We denote by $\texttt{DM}_d$, $d \in \{\mathrm{JC}, \mathrm{K2P}, \mathrm{F84}, \mathrm{LD}\}$ such a functionality. Note that this functionality is deterministic and, as we pointed before, we just have to prove expression (6.8) to hold for the simpler definition of `Real` and `Ideal`.

The protocol that privately computes the distance matrix $\texttt{DM}_d$ is built up by many invocations of a two-party distance functionality, denoted by $\texttt{D}_d$ for $d \in \{\mathrm{JC}, \mathrm{K2P}, \mathrm{F84}, \mathrm{LD}\}$. Consequently, we can reduce the the security of $\texttt{DM}_d$ to that of $\texttt{D}_d$ and use the composition theorem proved in [224] to prove $\texttt{DM}_d$ security.

Before presenting the composition theorem, we provide some informal definitions. We have that an *oracle-aided* protocol using the *oracle-functionality* f is a protocol where the parties can interact with an oracle which outputs to each party according to f. Also, when an oracle-aided protocol privately computes some g in the sense of (6.8) using the oracle-functionality f, we say that it *privately reduces* g to f. For a more detailed discussion on this topic, we refer the interested reader to [224]. The composition theorem for the semi-honest model can therefore be stated as follows:

Theorem 6. *(Composition theorem) Suppose that g is privately reducible to f and that there exists a protocol for privately computing f. Then, there exists a protocol for privately computing g.*

In other words, there exists a private protocol of g when the oracle-functionality f is substituted by its real private protocol in the corresponding oracle-aided protocol g.

6.3 Quantum tools

In this section, we present three quantum primitives used in the private computation of phylogenetic trees, rendering a full quantum-proof solution.

6.3.1 Quantum oblivious key distribution

We explored the concept of oblivious keys in chapter 3. To generate these oblivious keys, we saw that we can follow the prepare-and-measure quantum approach developed by Bennet [30] along with some commitment functionality. As an example, Lemus et al. [134] proposed to use the Halevi and Micali classical bit commitments based on universal and cryptographic hashing [225]. Thus, its security is based on the laws of physics and on the fact that there is no significant quantum speed-up in finding collisions on the hash-based bit commitments [134, 226, 227]. Also, as discussed in [40, 134], this protocol has an important security feature: it is resistant against *intercept now - decipher later* attacks. It this chapter, we call quantum oblivious key distribution (QOKD) the subprotocol of BBCS-based QOT protocols that comprises all the phases excluding the transfer phase. For illustration, the QOKD protocol in the bounded-quantum-storage model, $\Pi^{\mathbf{QOKD}}_{\mathbf{bqs}}$, is summarized in Figure 6.1.

6.3.2 Quantum random number generator

A random number generator (RNG) is another very important tool in the realm of secure multiparty computation. The SMC security can be compromised and the parties' privacy can be broken if the RNG used is predictable. An attack of this kind was reported in [3] where the authors exploited the Java weak random number generator used in v0.1.1 FastGC [228]. This attack allowed them to disclose the inputs of both parties in an SMC scenario. It also highlights the fact that it is not possible to use any kind of RNG for cryptographic purposes.

In the case of cryptographically secure pseudorandom number generators (CSRNG), it is crucial that it provides both forward and backward security. The former means that an attacker should not be able to predict the next generated number even when he knows all the generated sequence. The latter means that an attacker should not be able to predict all the generated sequence from a small set of generated elements. These two properties are not present in common RNGs. For example, linear congruential generators do not fit for cryptographic tasks since they can be easily predicted as reported in [229]. Also, Krawczk found that a large class of general congruential generators do not provide forward security even for obscured parameters [230]. So, in order to produce some CSRNG, instead of using linear operations, the research community decided to rely on the computational intractability of computing the discrete logarithm. Both [231] and [232] use modular exponentiation as an intermediate step in order to generate some pseudorandom bit. As mentioned above, all the cryptographic protocols with their security based on the discrete logarithm problem are threatened by quantum computers and these CSRNG

$\Pi^{\mathbf{QOKD}}_{\mathbf{bqs}}$ protocol

Parameters: n, security parameter.
Alice's input: $(m_0, m_1) \in \{0,1\}^l$ (two messages).
Bob's input: $b \in \{0,1\}$ (bit choice).

BB84 phase:

1. Alice generates random bits $\boldsymbol{x}^{\mathsf{A}} \leftarrow_{\$} \{0,1\}^n$ and random bases $\boldsymbol{\theta}^{\mathsf{A}} \leftarrow_{\$} \{+,\times\}^n$. Sends the state $\left|\boldsymbol{x}^{\mathsf{A}}\right\rangle_{\boldsymbol{\theta}^{\mathsf{A}}}$ to Bob.
2. Bob randomly chooses bases $\boldsymbol{\theta}^{\mathsf{B}} \leftarrow_{\$} \{+,\times\}^n$ to measure the received qubits. We denote by $\boldsymbol{x}^{\mathsf{B}}$ his output bits.

Waiting time phase:

3. Both parties wait time Δt.

Oblivious key phase:

4. Alice reveals to Bob the bases $\boldsymbol{\theta}^{\mathsf{A}}$ used during the *BB84 phase* and sets his oblivious key to $\mathsf{ok}^{\mathsf{A}} := \boldsymbol{x}^{\mathsf{A}}$.
5. Bob computes $\mathsf{e}^{\mathsf{B}} = \boldsymbol{\theta}^{\mathsf{B}} \oplus \boldsymbol{\theta}^{\mathsf{A}}$ and sets $\mathsf{ok}^{\mathsf{B}} := \boldsymbol{x}^{\mathsf{B}}$.

Alice's output: ok^{A}.
Bob's output: $(\mathsf{ok}^{\mathsf{B}}, \mathsf{e}^{\mathsf{B}})$.

Figure 6.1: QOKD protocol in the bounded-quantum-storage model.

protocols are not an exception. Besides this technique, one could use either AES or DES as cryptographically random generator.

Although these techniques are used to provide unpredictability and backward secrecy, all the randomness relies on some initial seed. This seed is used because all the process is based on deterministic algorithms. So, a pseudo RNG can be viewed as a randomness extractor from some initial random value. For this reason, it is crucial to use an initial random value that is as close as possible to a truly random value. This can be generated from different sources and usually, the best randomness comes from physical devices (e.g. atomic decay [233] or thermal noise [234]). So, a potentially good source of true RNG comes from natural phenomena where some part of the system is used as the source of entropy. In the case of classical natural phenomena, the entropy is frequently taken from some unknown or chaotic subsystem which can ultimately be described by a deterministic theory. In this case, the unpredictability drawn from the system's entropy comes from our lack of knowledge and inability to fully grasp the underlying complex natural mechanisms. Also, some classical phenomena (e.g. mouse pointers) may not have enough entropy to generate good quality random numbers. However, quantum natural phenomena have their roots in quantum mechanics which is intrinsically related to probability theory. For this reason, quantum systems can be potential sources of entropy even assuming complete knowledge of the system. This comes from the fact that, in quantum mechanics, we only have access to the probability distribution of the system's state and we can only know it after measuring it [235].

Within the scope of SMC, the generation of the circuit's wire keys must be guaranteed to be unpredictable and efficient. All these features can be achieved with a quantum RNG (QRNG) [236].

6.3.3 Quantum key distribution

As we will explain in the last section, part of the communication between the parties should be kept encrypted. Message encryption is commonly achieved with symmetric cryptographic tools, such as AES (Advanced Encryption Scheme) or the perfect cypher one-time pad. These symmetric tools are used to encrypt the communication content through a common key assumed to be only known by both communicating parties. However, the techniques used to distribute a common key cannot be realized using just symmetric cryptography and one needs asymmetric cryptography. Unfortunately, most of the commonly used techniques in asymmetric cryptography (RSA, Elliptic Curves or Diffie-Hellman) rely on computational assumptions that can be broken by a quantum computer through the already mentioned Shor's algorithm [21].

So, to render a quantum-resistant privacy-preserving solution, we make use of quantum key distribution (QKD) protocol to share symmetric keys to be used along with symmetric cryptography [163, 237–239]. Its security relies on the laws of quantum physics and it is proven to be resistant against computationally unbounded adversaries [74, 240]. This level of security comes from one very important quantum property known as the No-Cloning theorem. This property ensures that it is not possible to measure a quantum state without introducing a measurable perturbation in the system. Thus, both parties enrolling in the QKD protocol will be able to detect a potential eavesdropper in case some adversary tries to intercept and read the quantum signals.

6.4 Software tools

Next, we present the open-source tools used to implement the system presented in the subsequent sections.

6.4.1 CBMC-GC

The CBMC-GC compiler [241] is used in step 1) of Yao protocol to generate the boolean circuit representation of the desired function. It translates C-like code into boolean circuits based on a model checking tool called CBMC and it optimizes circuits for size and depth [242, 243]. HyCC [244] is also a potential candidate for this step as it builds upon CBMC-GC. However, it aims to build circuits for hybrid MPC protocols in which our system is not based.

6.4.2 Libscapi

The Libscapi library [213] implements several important cryptographic primitives for two-party and multi-party protocols. It is extensively used to implement steps $2-5$ of the Yao protocol in the repository MPC-Benchmark [245]. This implementation has integrated one of the most efficient OT extension protocols [246] along with the base OTs proposed by Chou and Orlandi [247].

6.4.3 PHYLIP

The PHYLIP package [217] is a C++ open-source project that provides a set of programs to infer phylogenies. Among other programs, it implements distance-based methods (UPGMA, Neighbour-Joining, Fitch-Margoliash) and computes the evolutionary distances described previously in section 6.1.1 (JK, K2P, F84, LD). Due to its modularity, we integrate PHYLIP distance methods with Yao protocol for evolutionary distances assisted with quantum technologies.

6.5 Secure multiparty computation of phylogenetic trees

The proposed system allows to securely compute a suite of algorithms that perform phylogeny analysis through the computation of phylogenetic trees. Based on the modular nature of distance-based algorithms, the system combines different evolution models with different phylogenetic algorithms. In this section, we describe how to integrate the tools presented in previews sections 6.3-6.4 to develop this modular private system.

6.5.1 Functionality definition

As already mentioned in section 6.1, all distance-based methods are divided into two phases: distance matrix computation and distance matrix processing. Apart from the metric used, the first phase is similar among all methods whereas the second phase is specific to each one while depending only on the distance matrix. Therefore, each phase corresponds to a particular functionality that can be formalized as follows:

- Functionality `DM`: receives some distance metric $d \in \{\mathrm{JC}, \mathrm{K2P}, \mathrm{F84}, \mathrm{LD}\}$ and all input sequences, and outputs a matrix with the pairwise distances between every sequence, i.e.

$$\mathtt{DM}(d; s_1, ..., s_m) = \begin{pmatrix} 0 & d_{1,2} & \cdots & d_{1,m} \\ d_{2,1} & 0 & & d_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ d_{m,1} & d_{m,2} & \cdots & 0 \end{pmatrix}$$

  where $d_{i,j} = d(s_i, s_j)$ for short.

- Functionality `A`: receives a distance matrix M and an algorithm type

$$a \in \{\mathrm{UPGMA}, \mathrm{NJ}, \mathrm{FM}\},$$

  and outputs the structure of the tree in newick tree format, i.e.

$$\mathtt{A}(M, a) = (\mathrm{subtree}_1 : l_1, \, \mathrm{subtree}_2 : l_2),$$

  where each l_1 and l_2 denotes the distance to its parent node, subtree is built up by other subtrees and the leaves are given by $(\mathrm{subtree}_{k-1} : l_{k-1}, \, s_{i_k} : l_k)$. For consistency, leaves are also considered as a subtrees. Note that this representation is not unique, e.g. $(s_1 : 0.7, (s_2 : 0.3, s_3 : 0.5) : 0.5)$ and $((s_3 : 0.5, s_2 : 0.3) : 0.5, s_1 : 0.7)$ represent the same rooted tree depicted in Figure 6.2.

  Therefore, if we consider the equivalence relation, $\sim$, given by

$$(\mathrm{subtree}_1 : l_1, \, \mathrm{subtree}_2 : l_2) \sim (\mathrm{subtree}_2 : l_2, \, \mathrm{subtree}_1 : l_1),$$

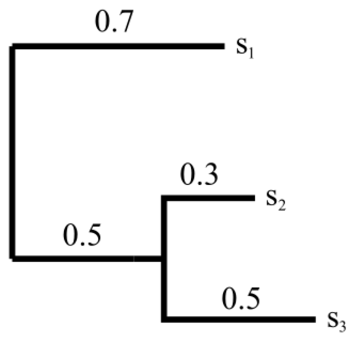

Figure 6.2: Example of rooted phylogenetic tree.

we have that the quotient set of the trees by $\sim$ satisfy the uniqueness property from an evolutionary point of view.

For simplicity, denote by A_d^a the private protocol that implements sequentially both functionalities described above, i.e. $A_d^a(s_1, ..., s_m) = \mathtt{A}(\mathtt{DM}(d; s_1, ..., s_m), a)$. This leads to twelve possible combinations of algorithms A_d^a for $d \in \{\mathrm{JC}, \mathrm{K2P}, \mathrm{F84}, \mathrm{LD}\}$ and $a \in \{\mathrm{UPGMA}, \mathrm{NJ}, \mathrm{FM}\}$.

6.5.2 Private protocol

During the distance matrix computation phase (DM) of the private A_d^a, each party has to compute the distance between his sequences and the other parties' sequences privately, i.e. without revealing his sequences to the other participating parties. Since this corresponds to several instances of a two-party secure computation, we make use of the Yao protocol described in section 2.2.1. This means that each party has to generate the boolean circuit representation of the elected distance d, which is accomplished by the CBMC-GC software tool before the beginning of the protocol. In section 6.7.1, we analyse how to generate these circuits.

Now, since the Yao protocol is executed only between two different parties P^i and P^j for $i, j \in [n]$, the other participating parties P^t, $t \in [n] \setminus \{i, j\}$, do not have access to the distances computed between theses two parties. For this reason, P^t has to receive the result of the Yao protocol execution from both P^j and P^i. After this, each party outputs the distance matrix that is used as the input of PHYLIP programs: `fitch`, `kitsch` or `neighbor`.

In the second phase of the protocol (A), the parties do not need to communicate because this phase only depends on the quantities computed during the first phase. For this reason, this phase is executed internally by each party, who then compute the phylogenetic tree. This phase is carried out by the PHYLIP programs mentioned in the previous paragraph.

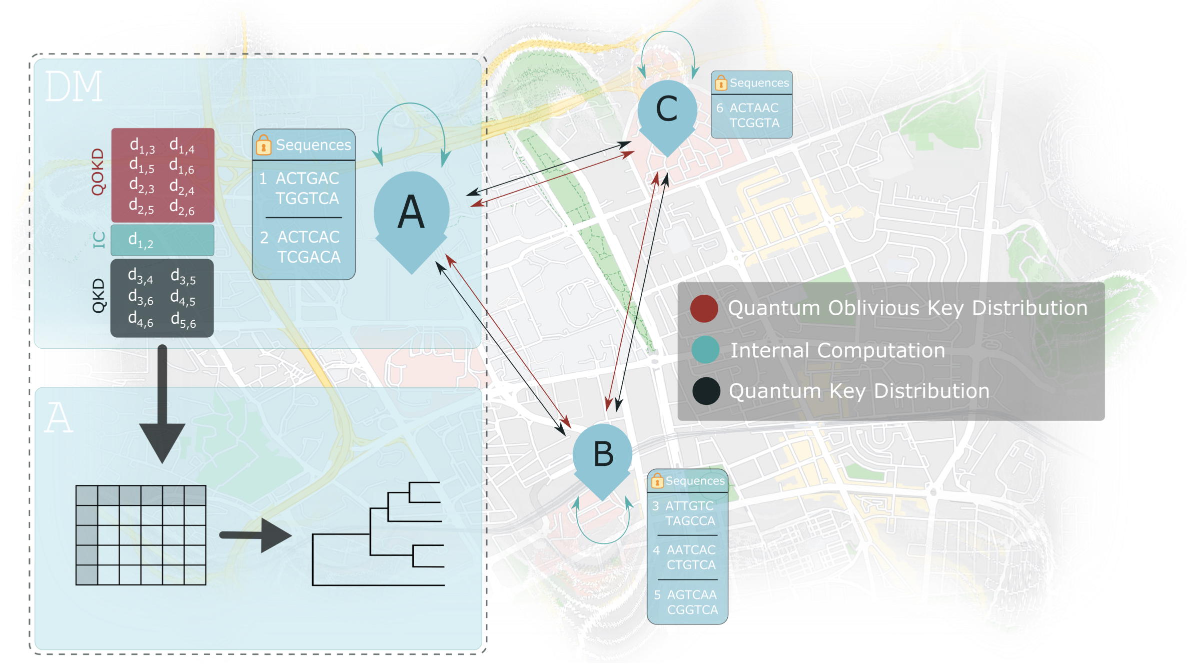

Figure 6.3: Overview of the A_d^a network structure.

These two phases are shown in Figure 6.3 and we give more details about the protocol assisted with quantum technologies in the next section.

6.5.3 Quantum private protocol

Let us specify the private A_d^a protocol with the quantum cryptographic tools. Following the scenario depicted in Figure 6.3, we define $S_i = \{s_{i,1}, ..., s_{i,l}\}$ to be the set of sequences owned by party P^i. Also, we denote by $d_{(i,l),(j,k)}$ the distance between the l-th sequence of party P^i and the k-th sequence of party P^j, i.e. $d_{(i,l),(j,k)} = d(s_{i,l}, s_{j,k})$.

As briefly described before, the private A_d^a protocol has two phases. The first phase requires different types of interactions between the parties to compute the desired distance matrix and the second phase is computed internally. Since the second phase is carried out internally, there is no need for communication between the parties. Therefore, the quantum cryptographic tools will only be used during the first private phase. In summary, each pair of parties require two quantum channels as depicted in Figure 6.3: one to generate oblivious keys for oblivious transfer and the other to generate symmetric keys for encryption.

Consider the case where P_t has to compute the distance matrix entry corresponding to distance $d_{(i,l),(j,k)}$. Depending on whether P_t owns both sequences, one of the sequences or none of the sequences $(s_{(i,l)}, s_{(j,k)})$, P_t proceed as follows:

1. If $i = j = t$ (i.e. both sequences are owned by P_t), $d_{(i,l),(j,k)}$ is computed internally

by P_t (blue arrow in Figure 6.3);

2. If $i = t$ and $j \neq t$ (i.e. one of the sequences is owned by P_t), $d_{(i,l),(j,k)}$ is computed privately with Yao protocol assisted with QOKD system (red arrow in Figure 6.3);

3. If $i \neq t$ and $j \neq t$ (i.e. none of the sequences is owned by P_t), both parties P_i and P_j (or just party P_i in case $i = j$) must send to P_t the distance $d_{(i,j),(k,l)}$ encrypted with the symmetric key generated through the QKD system (black arrow in Figure 6.3).

6.6 Quantum technologies integration

Now, let us see the role of quantum technologies in this private system and its integration with quantum networks.

6.6.1 Quantum oblivious transfer

Libscapi implementation of Yao protocol combines a very efficient base OT protocol with one of the fastest OT extension protocols. It uses the base OT (SimpleOT) proposed by Chou and Orlandi [87] integrated with the OT Extension (KOS15 [4]) presented in chapter 4. In this setting, the $\Pi_{\mathbf{O}}^{\mathbf{BBCS}}$ protocol can be implemented in two different ways depending on the number of oblivious keys generated between the two parties: as a base OT protocol integrated within OT extension protocol or as a stand-alone method substituting all Libscapi OT implementation. If the number of oblivious keys generated is scarce compared to the number of OT required, then one should integrate $\Pi_{\mathbf{O}}^{\mathbf{BBCS}}$ in the OT extension. Otherwise, one could directly use the $\Pi_{\mathbf{O}}^{\mathbf{BBCS}}$ protocol. A scheme of the integration of the quantum oblivious key distribution (QOKD) system is depicted in Figure 6.4.

It is important to note that the base OTs executed during the pre-computation phase of the OT extension have the parties' roles reversed. This means that the OT extension sender is the base OT receiver and vice-versa. This should be taken into consideration in case the $\Pi_{\mathbf{O}}^{\mathbf{BBCS}}$ is integrated within OT extension because $\Pi_{\mathbf{O}}^{\mathbf{BBCS}}$ is not symmetric in the sense that the apparatus used by the sender is different from that of the receiver. However, since it is known that OT is symmetric, we can use the reduction proposed in [248] without having to swap the quantum technological material.

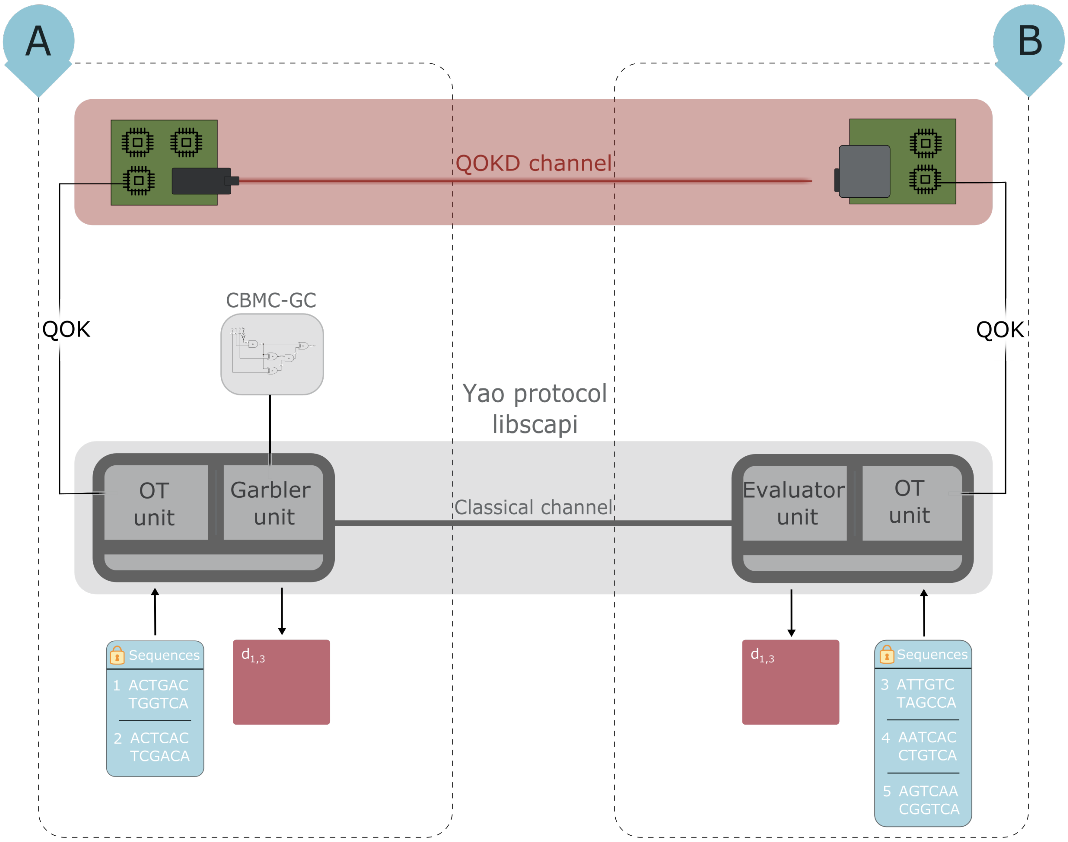

Figure 6.4: Overview of the integration of the QOKD service and the CBMC-GC tool in the Yao protocol.

6.6.2 Quantum random number generation

As previously described, the Yao protocol needs to generate random numbers for the keys in the *Wire encryption* step. This is crucial for the security of the protocol because its predictability allows deducing the parties' input as reported in [3].

Libscapi implementation makes use of OpenSSL library function `RAND_bytes` to randomly generate a seed from which it computes new numbers. In this private system, we substitute this function to a call of QRNG.

6.6.3 Quantum key distribution

The QKD system allows the participating parties to receive the distance elements of the sequences they do not own, while preserving the security of the system. We use the keys generated by the QKD system along with the perfect cipher: one-time pad.

6.6.4 Quantum network integration

Technological equipment

Both QKD and QOKD protocols rely on the same physical processes. They can both be realized either with continuous or discrete variables [134, 163, 238, 249]. Also, the technological equipment used by the receiver (Bob) and transmitter (Alice) is the same in both quantum services (QKD and QOKD). As for the case of the prepare-and-measure setting, the first quantum step is the same in both protocols: Alice randomly sends quantum states in two different bases and Bob measures these states on random bases. The difference relies on the classical post-processing phase. So, we can conclude that both services share the same technological equipment (fibre, receiver and transmitter). Moreover, as proposed by Pinto et al. [250] in a similar setting, both QKD and QOKD services can coexist with classical signals in the same fibre.

Network topology

The quantum private protocol explained above in section 6.5.3 assumes that every two parties have a direct quantum channel between them that is used to generate oblivious keys and symmetric keys, i.e. a fully connected quantum network. This approach follows from the fact that the first QKD and quantum OT (QOT) protocols were based on prepare-and-measure techniques [30, 130]. However, as discussed in chapter 3, there are also protocols that implement device-independent QOT (DI-QOT) [170, 178] (under some constraints) and DI-QKD [163]. In addition to the advantages from a security point of view, these DI protocols can also be implemented within a star-structured quantum network having an untrusted party as the middle point. This increases the implementation flexibility of the proposed quantum private protocol of phylogenetic trees (section 6.5.3).

As analysed by Joshi et al. [251], existing networks fall into three possible types: trusted node networks, actively switched and fully connected quantum networks based on entanglement sharing and wavelength multiplexing. Using the two types of protocols just mentioned (prepare-and-measure and device-independent), it is possible to implement our proposed system in all three existing quantum network implementation types.

Moreover, Kumaresann et al. [252] analyses possible SMC infrastructure topologies that can be created based on a set of OT channels shared between some pairs of parties in the network. They developed "secure protocols that allow additional pairs of parties to establish secure OT correlations using the help of other parties in the network in the presence of a dishonest majority" (Abstract, [252]). Since they work in the information-theoretical setting, there is no security loss in combining Kumaresann protocol with quantum approaches. This integration increases the range of configurations allowed. However, further efficiency analysis has to be done to understand the impact of this approach in practice.

6.7 System security

In this section, we analyse the security of the proposed system. We start by describing the methods used to privately compute the distance between two sequences and then we prove the security of the private protocol proposed in section 6.5.3 which implements the functionality described in section 6.5.1.

6.7.1 Private computation of distances

The private computation of the distance between sequences is an important building block in the security of the system. We have that the privacy of the sequences directly relies on this step. Here, we go through the methods used to compute the distances used by the PHYLIP program: Jukes-Cantor, Kimura 2-parameter, F84 and LogDet.

A common building block to all these four distance metrics is the computation of the Hamming distance between two sequences x and y, h_{xy}. We start by looking at an adapted divide-and-conquer way to compute the Hamming distance between two sequences and then we see how to apply it to the private computation of distance metrics.

Hamming distance

We are interested in the boolean representation of the Hamming distance and, as mentioned above, we use the CBMC-GC tool to translate ANSI-C code into this representation. Usually, to compute the Hamming distance between two binary strings, x and y, we start by applying the XOR operation, $z = x \oplus y$. Then, we just have to count the number of 1's in z. This operation is commonly known as population count or popcount(z) for short. So, the binary Hamming distance is given by $h_{xy} = \text{popcount}(x \oplus y)$.

We use an adapted divide-and-conquer technique for the computation of popcount(z) [253]. Originally, this divide-and-conquer technique starts by dividing the sequence into 2-bit blocks and then counts the number of 1's inside each 2-bit block. After that, it allocates the result of each block in a new 2-bit block. Then, we can sum the values inside these 2-bit blocks iteratively.

We follow the approach described above but we have to tailor it for the computation of the Hamming distance between two four-based sequences (A, C, G, T). Since we are using a boolean circuit representation, the nucleotide sequences must be represented in binary. So, by convention, we use the following 2-bit encoding: $A = 00$, $C = 01$, $G = 10$

and $T = 11$. If we follow directly the approach described above, we would have that the Hamming distance between the single-valued sequences "A" and "C" is smaller than the single-valued sequence between "A" and "T":

$$\begin{aligned} h(A, C) &= \text{popcount}(00 \oplus 01) \\ &= \text{popcount}(01) = 1, \end{aligned}$$

$$\begin{aligned} h(A, T) &= \text{popcount}(00 \oplus 11) \\ &= \text{popcount}(11) = 2. \end{aligned}$$

This issue comes from the fact that we are counting the number of 1's inside every 2-bit blocks. Instead, we are just interested in knowing if there is at least one element 1 inside each 2-bit block because it indicates that the bases at that site are different. Therefore, before counting the number of 1's in the `XOR`ed sequence, we apply an `OR` operation to the bits inside every 2-bit blocks. We call this operation $\text{popcount}^t(z)$. For simplicity, hereafter we denote by h_{xy} the tailored Hamming distance between sequences x and y. Now, we have that the tailored Hamming distance between "A" and "T" gives the desired result:

$$\begin{aligned} h(A, T) &= \text{popcount}^t(00 \oplus 11) \\ &= \text{popcount}^t(11) \\ &= \text{popcount}\big(\texttt{OR}(1, 1)\big) = 1. \end{aligned}$$

In Figure 6.5, we show an example on how to compute the Hamming distance between two-valued sequences "AG" and "GC".

Jukes-Cantor

As described in section 6.1.1, the Jukes-Cantor distance between two sequences is given by:

$$d_{xy} = -\frac{3}{4} \ln\left(1 - \frac{4}{3}\frac{h_{xy}}{N}\right),$$

where h_{xy} is the hamming distance between sequence x and sequence y.

Now, note that the function $f(x) = -\frac{3}{4} \ln\left(1 - \frac{4}{3}\frac{x}{N}\right)$ is one-to-one. This means that, from a privacy point of view, $f(x)$ carries the same amount of information than x. Therefore, we could simply proceed as follows:

1. Privately compute the Hamming distance, h_{xy}, using the tailored Hamming distance method described above and the Yao protocol assisted with quantum oblivious keys;

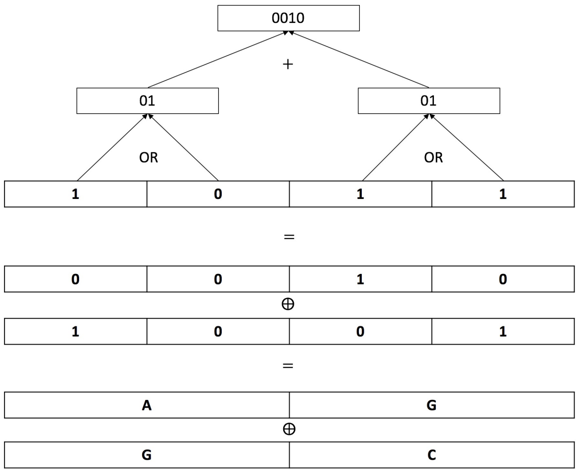

Figure 6.5: Overview of the tailored divide-and-conquer technique. This corresponds to lines `12-19` in Figure B.1 in Appendix B.

2. Internally compute $d_{xy} = f(h_{xy})$ (no need of quantum SMC).

This way, we just have to generate the boolean circuit for h_{xy} rather than generating for the full expression d_{xy}.

Kimura

In section 6.1.1, we saw that the Kimura 2-parameter model leads to the following distance:

$$d_{xy} = -\frac{1}{2} \ln \left((1 - 2P - Q)\sqrt{1 - 2Q} \right),$$

where $P = \frac{n_1}{N}$, $Q = \frac{n_2}{N}$ and n_1 and n_2 are respectively the number of sites for which two sequences differ from each other with respect to type I ("transition" type) and type II ("transversion" type) substitutions.

Similar to the case of Jukes-Cantor metric, note that $h(x) = -\frac{1}{2}\ln\left(\sqrt{\frac{x}{N^3}}\right)$ is one-to-one and only defined for $x > 0$. Thus, we can proceed as follows:

1. Privately compute the expression $c = (N - 2n_1 - n_2)^2(N - 2n_2)$ using the tailored Hamming distance method described above and the Yao protocol assisted with quantum oblivious keys;

2. Internally computes $d_{xy} = h(c)$ (no need of quantum SMC).

More precisely, the ANSI-C code that privately computes expression $c = (N - 2n_1 - n_2)^2(N - 2n_2)$ proceeds as follows. It uses the function $\text{popcount}_t(z)$ described above to compute the quantities n_1 and n_2. Observe that a transition type ($A \leftrightarrow G$ or $C \leftrightarrow T$)

renders the same XOR value:

$$\begin{aligned} A \oplus G &= 00 \oplus 10 = 10 \\ T \oplus C &= 11 \oplus 01 = 10. \end{aligned}$$

Therefore, using a four-sized sequence, the quantities n_1 and n_2 are given by:

$$\begin{aligned} n_1 &= 4 - \text{popcount}_t(x \oplus y \oplus 10101010) \\ n_2 &= \text{popcount}_t(x \oplus y) - n_1. \end{aligned}$$

F84 and LogDet

Recall from section 6.1.1, that the F84 (F_{xy}) and LogDet (L_{xy}) distances are given, respectively, by:

$$F_{xy} = -2A \ln\left(1 - \frac{P}{2A} - \frac{(A-B)Q}{2AC}\right) + 2(A-B-C)\ln\left(1 - \frac{Q}{2C}\right), \tag{6.9}$$

$$L_{xy} = -\frac{1}{4}\ln\left(\frac{\det F_{xy}}{\sqrt{\det \prod_x \prod_y}}\right), \tag{6.10}$$

where $A = \frac{\pi_C\pi_T}{\pi_Y} + \frac{\pi_A\pi_G}{\pi_R}$, $B = \pi_C\pi_T + \pi_A\pi_G$ and $C = \pi_R\pi_Y$ for $\pi_Y = \pi_C + \pi_T$ and $\pi_R = \pi_A + \pi_G$, and P and Q are defined as in the Kimura 2-parameter mode above. Also, the divergence matrix F_{xy} is a 4×4 matrix such that the $ij-$th entry gives the proportion of sites in sequence x and y with nucleotide i and j, respectively. Also, $\prod_x$ and $\prod_y$ are diagonal matrices where its $i-$th component correspond to the proportion of i nucleotide in the sequence x and y, respectively.

As before, we want to split the private computation of both F_{xy} and L_{xy} in two steps. Note that, in this case, there is no clear way to define two bijective functions, $g()$ and $q()$, on some simple parameters, d and e, such that $F_{xy} = g(d)$ and $L_{xy} = p(e)$. By simple parameters, we mean parameters that do not depend on complex operations such as logarithm or square root. Instead, one can use the CORDIC algorithm [254, 255] for square-roots and logarithm functions and translate an approximation of both F_{xy} and L_{xy} into boolean circuits.

6.7.2 Private computation of phylogenetic trees

In this section, we prove that the protocol A^a_d described in section 6.5.3 securely implements functionality $\mathtt{A} \circ \mathtt{DM}$ described in section 6.5.1 according to the security definition 10. So, we want to prove the following theorem:

Theorem 7. *The protocol A_d^a securely realizes $\mathtt{A} \circ \mathtt{DM}$ in the presence of semi-honest adversaries.*

We start by noting that the ideal functionality outputs the distance matrix to the parties and that during $\mathtt{A}$ computation there is no interaction between the parties. Therefore, the security of the system is independent of the distance-based algorithm used (UPGMA, Neighbour-Joining or Fitch-Margoliash) and we can only focus on the computation of $\mathtt{DM}$ functionality.

As already mentioned, the protocol that implements the functionality $\mathtt{DM}$ is built up by many invocations of a two-party distance functionality, denoted by $\mathtt{D}_d$ for $d \in \{\mathrm{JC}, \mathrm{K2P}, \mathrm{F84}, \mathrm{LD}\}$. So, in order to prove the above theorem, we will need to following two lemmas:

Lemma 15. *A_d^a privately reduces $\mathtt{DM}$ to $\mathtt{D}_d$, i.e. an oracle-aided A_d^a protocol privately computes $\mathtt{DM}$ using the oracle-functionality $\mathtt{D}_d$.*

Proof. In order to prove this lemma, we have to develop a simulator Sim that simulates the view of a set of corrupted parties C. The Sim starts from receiving all the input sequences from the corrupted parties. It then proceeds as follows:

1. Generates random sequences of the honest parties, H.
2. Invokes the oracle-functionality $\mathtt{D}_d$ on these sequences.
3. Sends to all corrupted parties C the results of distances computed from honest parties sequences
4. Invokes the oracle-functionality $\mathtt{D}_d$ on the sequences owned by the corrupted parties.
5. Invokes the oracle-functionality $\mathtt{D}_d(s_i, s_j)$ for $s_i \in H$ and $s_j \in C$.

In a real execution, the corrupted parties will only receive the distances computed by $\mathtt{D}_d$ on the honest parties sequences (as in step 2.), on their sequences (as in step 4.) and between corrupt and honest parties. Therefore, we have that the oracle-aided A_d^a protocol privately computes $\mathtt{DM}$ using the oracle-functionality $\mathtt{D}_d$.

□

Lemma 16. *Yao protocol with the OT primitive instantiated by $\Pi_{\boldsymbol{O}}^{\boldsymbol{BBCS}}$ protocol (Figure 4.3) privately computes $\mathtt{D}_d$.*

Proof. In [81] it was developed a framework that allows quantum protocols to be composed in a classical environment. They also mention that a general secure function evaluation remains secure when instantiating the OT primitive by a secure quantum version. In [256], it was proved that $\Pi_{\mathbf{O}}^{\mathbf{BBCS}}$ protocol is secure according to the security definition given in [81]. Therefore, we can compose the $\Pi_{\mathbf{O}}^{\mathbf{BBCS}}$ protocol with a Yao protocol [257] while preserving the overall security.

□

So, from Lemma 15 and 16 we can use the composition theorem 6 and conclude that the protocol A_d^a is secure.

We have proved that our system is well designed and secure against quantum computer attacks under the semi-honest model. In order to extend the protocol to the malicious setting, we just have to implement a two-party secure computation protocol that is secure in the malicious adversary model [69].

6.8 Complexity analysis

In this section, we start by analysing the complexity of the protocol A_d^a presented before. We assume there are n parties, $P^1, \ldots, P^n$, with $M_1, \ldots, M_n$ sequences, respectively. Also, we assume that the sequences are aligned and that they have the same number of nucleotides, s.

6.8.1 Protocol complexity analysis

Now, let us analyse the complexity of the protocol presented in section 6.5.3.

Yao protocol executions

Regarding the number of Yao protocol executions, we have that each party P^j owning M_j sequences has to perform $N^j_{\text{Yao}} = M_j \sum_{i \neq j} M_i$ secure distance computations. So, the total number of Yao protocol executions is given by

$$N_{\text{Yao}} = \sum_j N^j_{\text{Yao}} = \sum_{j, i \neq j} M_j M_i.$$

If we assume the number of sequences per party to be the same, i.e. $M_j = M \,\forall j \in [n]$, then we can simplify the expression above and conclude that $N_{\text{Yao}} = M^2 n(n-1)$. This means that the number of Yao protocol executions is quadratic in the number of sequences per party ($\mathcal{O}(n^2)$) and also in the number of parties ($\mathcal{O}(M^2)$).

OT executions

From N_{Yao} we can deduce the number of OT executions. In the Yao protocol, we need to execute one OT for each of the evaluator's input wires. For a sequence with s nucleotides and using a two-bit representation of each nucleotide, the boolean circuit that computes the distance between two sequences will have $2s$ input wires for each party input. Therefore, each party executes the following number of OT executions ($\forall j$):

$$\begin{aligned} N_{\text{OT}}^j &= N_{\text{Yao}}^j \cdot 2s \\ &= 2sM^2(n-1). \end{aligned}$$

It is important to note that N_{OT}^j is independent of the size of the boolean circuit used, i.e. it is independent of the distance metric d used in the protocol. This is a consequence of using the Yao protocol where the number of OT only depends on the input size. In case we were using GMW [15] protocol, the number of OT per party would depend on the size of the circuit.

As mentioned in section 6.6.1, in case the number of oblivious keys generated is scarce compared to the number of OT required, we can use the $\Pi_{\mathbf{O}}^{\mathbf{BBCS}}$ protocol to generate the base OT used within OT extension protocol. In this case, we just have to generate κ $\Pi_{\mathbf{O}}^{\mathbf{BBCS}}$ protocols per Yao execution: $L_{\text{bOT}}^j = N_{\text{Yao}}^j \cdot \kappa = \kappa M^2(n-1)$.

Oblivious keys

At this point, we can easily deduce the size of oblivious keys that each pair of parties have to generate when using messages of size l.

In case we use $\Pi_{\mathbf{O}}^{\mathbf{BBCS}}$ protocol to generate the final OT:

$$\begin{aligned} L_{\text{ok}}^j &= N_{\text{OT}}^j \cdot 2l \\ &= 4slM^2(n-1). \end{aligned}$$

Also, we can use the number of OT executions per party and the analysis from Table 4.5 and [40] to compute the computational and communication complexity (in bits) of $\Pi_{\mathbf{O}}^{\mathbf{BBCS}}$:

$$\begin{aligned} \mathcal{C}_{\text{comp}}^j &= N_{\text{OT}}^j \cdot 8l \\ &= 16slM^2(n-1), \end{aligned}$$

$$\begin{aligned} \mathcal{C}_{\text{comm}}^j &= N_{\text{OT}}^j \cdot 3l \\ &= 6slM^2(n-1). \end{aligned}$$

In case we use $\Pi_{\mathbf{O}}^{\mathbf{BBCS}}$ protocol to generate the base OT, the total size of oblivious key required is:

$$\begin{aligned} L^j_{\text{bok}} &= N^j_{\text{bOT}} \cdot 2l \\ &= 2\kappa l M^2(n-1). \end{aligned}$$

QRNG

The QRNG has to generate twice the total length of oblivious keys, i.e. $L_{\text{QRNG}} = 2L_{\text{ok}}$.

Internal computation

Number of internal computations per party:

$$N^j_{\text{int}} = \binom{M}{2} = \frac{M!}{2!(M-2)!}.$$

Encryption keys

As discussed before, for every party P^j, P^t $(t \neq j)$ has to receive from P^j the distances known by P^j that P^t does not have access. So, P^j has to send $M^2(n-2) + N^j_{\text{int}}$ distance values to P^t. Consequently, the length of the QKD key used to send these distances to P^t is:

$$32(M^2(n-2) + N^j_{\text{int}}),$$

for a 32−bit number representation. Therefore, the total size of key shared between two parties P^j and P^t must be:

$$L^{jt}_{qkd} = 64(M^2(n-2) + N_{\text{int}}).$$

Also, each party must have an overall shared key of $L^j_{qkd} = \sum_{i\neq j} L^i_{qkd} = 64(n-1)(M^2(n-2) + N^j_{\text{int}})$.

6.8.2 Use case

We now present the scenario used to test and compare both quantum-assisted and classical-only approaches. We start by exploring the complexity analysis and the OT comparison carried out in previous sections. We extend this analysis in the next section with a testbed implementation.

We consider a scenario where three parties $n = 3$ have M SARS-CoV-2 genome sequences (with length $s = 32\,000$) and want to privately compute a phylogenetic tree from them. In the next section we consider a varying number of sequences, but, for now, we set

Parameter	Formula	Amount	Generation Time
L^j_{ok}	$4slM^2(n-1)$	3.3×10^9 bit	5m30s
L^j_{bok}	$2\kappa lM^2(n-1)$	6.6×10^6 bit	0.64s
L^j_{QRNG}	$8slM^2(n-1)$	6.6×10^9 bit	28s
L^j_{qkd}	$64(n-1)(M^2(n-2)+\binom{M}{2})$	18.6×10^3 bit	1.9×10^{-3}s
N^j_{Yao}	$M^2(n-1)$	200	
N^j_{OT}	$2sM^2(n-1)$	12.8×10^6	
N^j_{bOT}	$\kappa M^2(n-1)$	25.6×10^3	
N^j_{int}	$\binom{M}{2}$	45	

Table 6.1: Complexity analysis where $n = 3$, $M = 10$, $s = 32\,000$ and $l, \kappa = 128$. L^j_{ok}: size of total oblivious key. L^j_{bok}: total size of oblivious key for base OT. L^j_{QRNG}: random bits generated by QRNG. L^j_{qkd}: total size of QKD keys. N^j_{Yao}: number of Yao protocol executions. N^j_{OT}: number of OT executions. N^j_{bOT}: number of base OT executions. N^j_{int}: number of internal computations.

$M = 10$. Following a standard choice [3], we consider garbled circuit keys with $l = 128$ bits, computational security parameter with $\kappa = 128$ bits and statistical security parameter with $w = 64$ bits. For these parameter values, we can instantiate the expressions deduced in the complexity analysis (section 6.8.1). This information is summarized in Table 6.1. As expected, the total size of oblivious keys (L^j_{ok}) required for a scenario where $\Pi^{\mathbf{BBCS}}_{\mathbf{O}}$ is the main OT protocol is three orders of magnitude higher than the case where $\Pi^{\mathbf{BBCS}}_{\mathbf{O}}$ serves as a base OT protocol in KOS15 (L^j_{bok}). Also, we note that the total size of symmetric keys required in the protocol (L^j_{qkd}) is much smaller than that of oblivious keys (L^j_{ok} and L^j_{bok}), pointing to the fact that its management should be less expensive than the oblivious keys management system. This will be discussed further in the next section.

We can also estimate the time required to generate the keys based on their size. If we consider state-of-the-art rates of 10 Mbit/s for both QKD and QOKD systems [258] and a rate of 240 Mbit/s for QRNG (ID Quantique QRNG PCIe cards [259]), we would need around 5 minutes for L^j_{ok}, 0.64s for L^j_{bok}, 28s for L^j_{QRNG} and 1.9×10^{-3}s for L^j_{qkd}. Note that we can significantly reduce the time of the precomputation phase in case we integrate $\Pi^{\mathbf{BBCS}}_{\mathbf{O}}$ with KOS15 OT extension protocol.

6.9 Performance evaluation

In this section, we set out to explore and compare the performance of two implementations of the proposed secure phylogenetic tree computation (A_d^a): classical-only and quantum-assisted. The quantum-assisted system replaces Libscapi base OT (SimpleOT [247]) implementation with the $\Pi_{\mathbf{O}}^{\mathbf{BBCS}}$ protocol presented before (Figure 4.4). It also uses symmetric keys along with one-time pad to encrypt distance values as described in section 6.5. More specifically, we benchmark our implementation for the duration of its main components: circuit generation, communication, (internal) computation and SMC operation.

Here, we do not assess the generation performance of both symmetric keys and oblivious keys. We precompute these keys using a simulator that mimics the structure of the quantum generated keys and we do not include their generation time in the performance analysis. The reason for this is twofold: performance in quantum cryptography is an active field of research with no clear way on how to be compared with classical approaches; quantum generation of both keys (symmetric and oblivious) can be precomputed without depending on the parties' inputs and used later as a resource in the execution of the system.

6.9.1 Setup

We leverage a testbed on a virtual environment composed of three Ubuntu (64-bit) 16.04.3 Virtual Machines (VM) with 3GB of RAM. The virtual environment was created using VirtualBox and the VMs were running on a 2.6 GHz Intel Core i7 processor.

The performance of the implementation was measured on the VMs with the clock type `CLOCK_REALTIME` from the C++ library `time`. Although the values might differ for different host machines, this method is certainly adequate to use as a comparison between a classical-only and a quantum-assisted system.

We follow the scenario presented in section 6.8.2, where we have three parties ($n = 3$) owning at most ten sequences ($M \leq 10$) with 32 000 nucleotides. For the sake of comparison, we use the Jukes-Cantor phylogenetic distance along with PHYLIP implementation of UPGMA algorithm, i.e. $(d, a) = (\mathrm{JC}, \mathrm{UPGMA})$.

Sequences preprocessing

The 30 sequences used in this testbed were taken from GISAID database [260] which collects SARS-CoV-2 genome sequences. These sequences were then aligned using the Clustal Omega API [261]. After alignment, the sequences (4-based) were translated to bits according to the following rule: $A \rightarrow 00$, $C \rightarrow 01$, $G \rightarrow 10$ and $T \rightarrow 11$. Note

Min. Time	Time	Nº of gates	Depth
0s	1m42.7s	2 489 218	29 771
100s	3m30.7s	2 205 372	21 711
200s	5m9.3s	2 205 372	21 711

Table 6.2: Generation of Jukes-Cantor boolean circuit. Min. Time: Minimization Time.

that this alignment procedure is not privacy-preserving and was only used for testing purposes. A privacy-preserving alignment can be easily executed if all parties agree on a public reference sequence and align locally their sequences against this reference.

6.9.2 Circuit generation

As mentioned above, the CBMC-GC tool can generate a boolean circuit description of the phylogenetic distance from its corresponding ANSI-C code. In Table 6.2, we present the generation time of the Jukes-Cantor boolean circuit for three different minimization time values (CBMG-GC parameter). The minimization time is a parameter of the CBMC-GC tool that regulates the time spent to minimize the size of the boolean circuit. We note that the generation of the circuit only has to be carried out once. From Table 6.2, we can see that the minimization time for values above 100s does not have a great impact on the minimization of both the number of gates and circuit depth. The C code describing the Jukes-Cantor distance is shown in Appendix B.

6.9.3 System execution time

We start by recalling that the proposed secure algorithm is divided into the following parts:

1. Distance Matrix, `DM`:
   (a) Pairwise SMC computation of distances, `SMC`;
   (b) Pairwise internal computation of distances, `IC`;
   (c) Sending/Receiving other sequences, `Com`;
2. Phylogenetic computation, `A`.

We join the internal computation of sequences and PHYLIP phylogenetic computation into the same category and assess three different components for both classical and quantum runs: Communication (`Com`), SMC (`SMC`) and Computation (`IC`, `A`). In Tables 6.3

Nº of Seq.	2	4	6	8	10
Comm.	3,95%	0,98%	0,44%	0,25%	0,16%
SMC	95,95%	98,94%	99,48%	99,68%	99,77%
Comp.	0,10%	0,08%	0,07%	0,07%	0,07%

Table 6.3: Percentage weight of each component in the classical-only system.

Nº of Seq.	2	4	6	8	10
Comm.	3,75%	0,93%	0,39%	0,22%	0,14%
SMC	96,15%	98,99%	99,55%	99,72%	99,81%
Comp.	0,10%	0,07%	0,06%	0,06%	0,05%

Table 6.4: Percentage weight of each component in the quantum-assisted system.

and 6.4, we show the proportion of each component. As expected, in both systems the pairwise SMC computation of distances represents the greatest portion, accounting for more than 95% of the time for all different numbers of sequences. However, the weight of SMC in the quantum-assisted system is consistently higher than the classical-only system for all cases. This can be explained by the fact that the quantum-assisted SMC takes longer than the classical-only SMC.

Figure 6.6 present us with the average duration of both systems with standard deviation as error bars. Here, we see that the quantum-assisted approach has a higher cost than the classical-only implementation. As discussed in section 6.6.1, we can either use the $\Pi_{\mathbf{O}}^{\mathbf{BBCS}}$ protocol as the main OT in the Libscapi implementation or we can use it as a base OT in the KOS15 OT Extension used by Libscapi. Since we have implemented the latter, our $\Pi_{\mathbf{O}}^{\mathbf{BBCS}}$ is competing against the SimpleOT [247] base OT implementation. As analysed by the authors (section 4 [40]), the $\Pi_{\mathbf{O}}^{\mathbf{BBCS}}$ transfer phase is expected to outperform base OT implementations and to have comparable performance to OT Extension protocols. However, these analyses only compared cryptographic and computational operations and did not take into account implementation constraints and memory complexity.

In the quantum-assisted implementation, we separate the precomputation phase (generation of symmetric and oblivious keys) from the secure computation phase of the proposed protocol, A_d^a. For this reason, it is necessary to develop a key management system to save and keep key synchronization between parties. Consequently, the key management system becomes the bottleneck as the number of sequences increases. In particular, the key management system of oblivious keys is responsible for most of the overhead

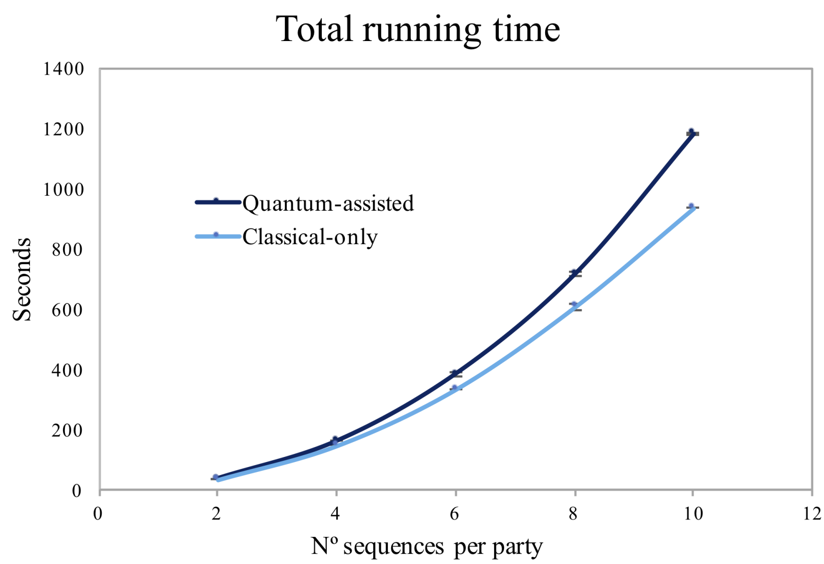

Figure 6.6: Total running time of both quantum-assisted and classical-only systems.

(Figure 6.7).

The reason for oblivious keys management to be more expensive than symmetric management and to be the main cause of overhead is twofold: the total size of oblivious keys used is three orders of magnitude higher than that of symmetric keys (compare L^j_{qkd} and L^j_{bok} from Table 4.5); oblivious keys are loaded into ROM memory (slower access) whereas symmetric keys are loaded into RAM memory (faster access). The main reason for oblivious keys to be managed from a file system is that it allows to use Libscapi implementation of Yao protocol in a modular way, i.e. we only have to change the type of base OT used by Libscapi implementation without tailoring any other module.

As the management of files is time-sensitive to their size, the proportion of time spent on the overhead due to the oblivious key management system (OKM) increases with the number of shared keys per party. This can be confirmed by Figure 6.8 which shows the proportion of time spent by the oblivious key management system in the difference between the quantum-assisted and the classical-only system.

Future work is required to develop more efficient oblivious key management systems. Despite this difference, we stress that the quantum-assisted system has a significantly higher degree of security against quantum computer attacks.

6.10 Conclusion

In this chapter, we presented an SMC protocol assisted with quantum technologies tailored for distance-based algorithms of phylogenetic trees. It is a modular protocol that uses one distance metric taken from four possible evolutionary models (Jukes-Cantor, Kimura

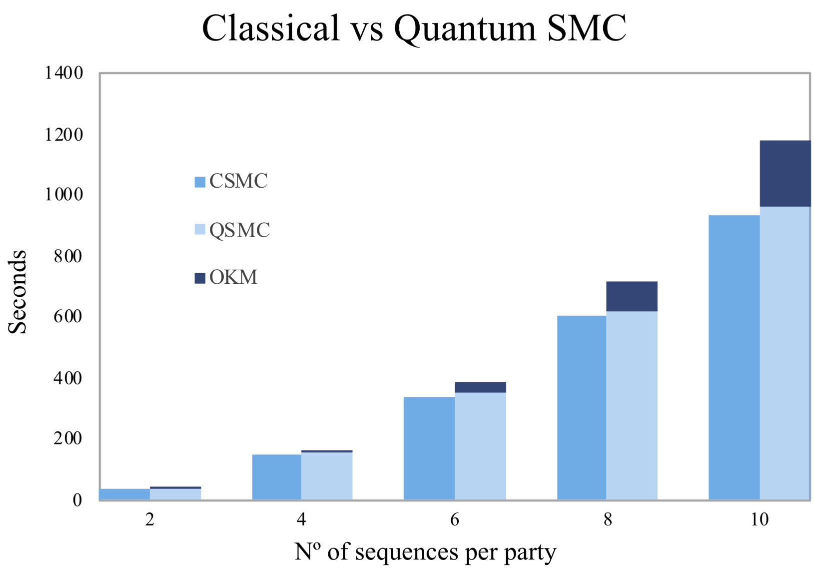

Figure 6.7: Total running time of the pairwise SMC computation of distances for both quantum-assisted and classical-only systems. CSMC: classical-only SMC; QSMC: quantum-assisted SMC; OKM: oblivious key management system.

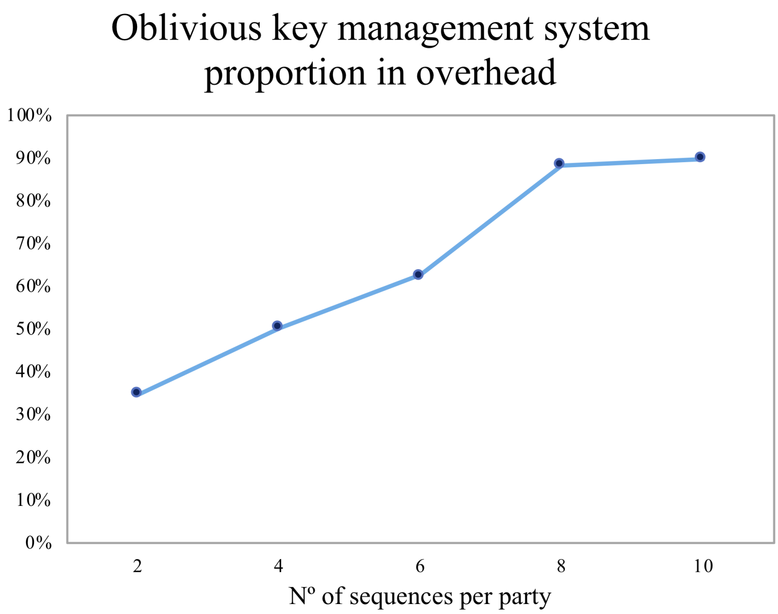

Figure 6.8: Oblivious key management system proportion in the overhead of quantum-assisted system.

2-parameter, F84 and LogDet) and three different protocols (UPGMA, Neighbour-Joining and Fitch-Margoliash). In total, we can implement twelve different combinations of protocols.

The proposed system is based on ready to use libraries (CBMC-GC, Libscapi and PHYLIP) that are integrated with quantum technologies to provide a full quantum-proof solution. We use the quantum version of primitives that play a central role in the security of the system: oblivious transfer, encryption and random number generation.

We compare the performance of a classical-only and a quantum-assisted system based on simulated symmetric and oblivious keys. Previous analyses on the computation and communication complexity point to a scenario where the quantum-assisted version does not add an extra efficiency cost. This is confirmed by comparing the running times of both approaches without considering the overhead created by the oblivious key management system that increases with the number of shared keys. Further work is required to develop more efficient key management systems. Despite this extra cost, the quantum-assisted version significantly improves the system security when compared with the classical-only as it renders a protocol with enhanced security against quantum computers.

Chapter 7

Conclusion

7.1 Future work

Future work is required to develop more efficient oblivious key management systems. Despite this difference, we stress that the quantum-assisted system has a significantly higher degree of security against quantum computer attacks. This can be done by including a field that receives a pointer to the oblivious key.

Appendix A

Proof of Lemma 10 (Dishonest Bob)

As we mentioned in the main text this proof is a combination and adaptation to our case of results from [29] and [73].

To simplify the notation, in this proof we drop the subscript 0 that refers to the RWOLE phase, e.g. we write Alice's function vector $\mathbf{F}_0$, simply as $\mathbf{F}$.

Let the values that Bob commits be fixed as $(x_i, r_i)\, \forall i \in [m]$, where $m = (1+t)n$ and tn the number of qudits $\left|e_{r_i}^{x_i}\right\rangle$ used in the *Test Phase* to check whether he is honest or not. Throughout the proof we denote by $\boldsymbol{x} = (x_1, \ldots, x_m)$ and $\boldsymbol{r} = (r_1, \ldots, r_m)$ the vectors in $\mathbb{Z}_d^m$ whose components contain Bob's commitments $\forall i \in [m]$, and by $X^m = (X_1, \ldots, X_m)$ the vector of the random variables associated to $x_i, i \in [m]$. For each pair (x_i, r_i), the corresponding qudit $\left|e_{r_i}^{x_i}\right\rangle$ belongs in the Hilbert space $\mathcal{H}_{X_i}$, and the quantum system including all the qudits is in $\mathcal{H}_{X^m} = \bigotimes_{i\in[m]} \mathcal{H}_{X_i}$. For simplicity, we refer to the quantum systems in terms of the corresponding random variables X_i, instead of the Hilbert spaces $\mathcal{H}_{X_i}$.

Recall that the set $T \subset [m]$ contains the tn indices of the test qudits, and by $\bar{T}$ we denote its complement $[m]\backslash T$. For $i \in T$, $\boldsymbol{x}_{|T}$ is the vector whose components are the bases x_i in which Alice will measure the test qudits, and $\boldsymbol{r}'_{|T}$ is the vector whose components are Alice's measurement results. The corresponding quantum system is in the Hilbert space $\bigotimes_{i\in T} \mathcal{H}_{X_i}$, which for simplicity we denote as $X_{|T}^m$ in terms of the associated random variables. Finally, $r_H(\cdot,\cdot) = d_H(\cdot,\cdot)/n$ is the relative Hamming distance between two vectors of size n, with $d_H(\cdot,\cdot)$ being their Hamming distance.

Proof. Let us start by proving property 1. of Lemma 10. After the first step of the protocol π_{RWOLE}^n (Figure 5.2), the generated state is ρ_{X^mE}, where E is an auxiliary quantum system that Bob holds. Without loss of generality we assume that $\rho_{X^mE} = |\phi_{X^mE}\rangle\langle\phi_{X^mE}|$, i.e, it is a pure state[1]. If Bob is honest, we have that $|\phi_{X^mE}\rangle = |e_{\boldsymbol{r}}^{\boldsymbol{x}}\rangle \otimes |\psi_E\rangle$, i.e., Bob's auxiliary quantum system E is not entangled to the states that he sends to Alice.

[1]Otherwise, we purify it and carry the purification system along with E.

The *Test Phase* of the protocol is used to guarantee that the real state is close to an ideal state that satisfies the properties 1. and 2. of Lemma 10. Let $\boldsymbol{r'}$ be the vector whose components are Alice's outcomes when measuring the state of X^m in the committed bases $\boldsymbol{x}$, and let $\mathcal{T}$ be the random variable associated to the set of indexes T of size tn. We can consider the state:

$$\rho_{\mathcal{T}X^mE} = \rho_{\mathcal{T}} \otimes |\phi_{X^mE}\rangle\langle\phi_{X^mE}| = \sum_T P_{\mathcal{T}}(T)\, |T\rangle\langle T| \otimes |\phi_{X^mE}\rangle\langle\phi_{X^mE}|\,, \tag{A.1}$$

to be the state resulting from the real execution of the protocol, and prove that it is close to some state, $\sigma_{\mathcal{T}X^mE}$, that fulfills the following property: for any choice of T and for any outcome $\boldsymbol{r'}_{|T}$ when measuring the state of $X^m{}_{|T}$ in the bases $\boldsymbol{x}_{|T}$, the relative error $r_H(\boldsymbol{r'}_{|T}, \boldsymbol{r}_{|T})$ is an upper bound on the relative error $r_H(\boldsymbol{r'}_{|\bar{T}}, \boldsymbol{r}_{|\bar{T}})$, which one would obtain by measuring the remaining subsystems $X^m{}_{|\bar{T}}$ in the bases $\boldsymbol{x}_{|\bar{T}}$. This state, $\sigma_{\mathcal{T}X^mE}$, can be written as:

$$\sigma_{\mathcal{T}X^mE} = \sum_T P_{\mathcal{T}}(T)\, |T\rangle\langle T| \otimes \left|\tilde{\phi}^T_{X^mE}\right\rangle\left\langle\tilde{\phi}^T_{X^mE}\right|, \tag{A.2}$$

where $\forall T$,

$$\left|\tilde{\phi}^T_{X^mE}\right\rangle = \sum_{\boldsymbol{r'} \in \mathcal{B}_T} \alpha^T_{\boldsymbol{r'}}\, |e^{\boldsymbol{x}}_{\boldsymbol{r'}}\rangle \otimes \left|\psi^{\boldsymbol{r'}}_E\right\rangle, \tag{A.3}$$

for $\mathcal{B}_T = \{\boldsymbol{r'} \in \mathbb{Z}_d^m : r_H(\boldsymbol{r'}_{|\bar{T}}, \boldsymbol{r}_{|\bar{T}}) \leq r_H(\boldsymbol{r'}_{|T}, \boldsymbol{r}_{|T}) + \zeta\}$ for $\zeta > 0$ and arbitrary coefficients $\alpha^T_{\boldsymbol{r'}}$. The state $\left|\psi^{\boldsymbol{r'}}_E\right\rangle$ is an arbitrary state on E subsystem that possibly depends on $\boldsymbol{r'}$. Then we have:

Lemma 17. *Let the quantum states $\rho_{\mathcal{T}X^mE}$ and $\sigma_{\mathcal{T}X^mE}$ be given by* (A.1) *and* (A.2), *respectively. Then, $\forall \zeta > 0$ and fixed strings $\boldsymbol{x}, \boldsymbol{r} \in \mathbb{Z}_d^m$, we have that*

$$\rho_{\mathcal{T}X^mE} \approx_\epsilon \sigma_{\mathcal{T}X^mE},$$

where $\epsilon(\zeta, n) = \epsilon(\zeta, n) = \exp\Big(-\frac{2\zeta^2 t^2 n^2}{(nt+1)(t+1)}\Big)$. That is, the real state $\rho_{\mathcal{T}X^mE}$ is exponentially close, with respect to n, to the ideal state $\sigma_{\mathcal{T}X^mE}$.

Proof. This proof is an adaptation of the proof of Lemma 4.3 from [29] to our case.

For any T, let $\left|\tilde{\phi}^T_{X^mE}\right\rangle$ be the renormalized projection of $|\phi_{X^mE}\rangle$ into the subspace

$$\text{Span}\,\{|e^{\boldsymbol{x}}_{\boldsymbol{r'}}\rangle : \boldsymbol{r'} \in \mathcal{B}_T\} \otimes \mathcal{H}_E,$$

and let $\left|\tilde{\phi}^{T\perp}_{X^mE}\right\rangle$ be the renormalized projection of $|\phi_{X^mE}\rangle$ into its orthogonal complement.

We can, then, write

$$|\phi_{X^mE}\rangle = \epsilon_T \left|\tilde{\phi}^T_{X^mE}\right\rangle + \epsilon^\perp_T \left|\tilde{\phi}^{T\perp}_{X^mE}\right\rangle,$$

with $\epsilon_T = \left\langle \tilde{\phi}^T_{X^m E} \middle| \phi_{X^m E} \right\rangle$ and $\epsilon^\perp_T = \left\langle \tilde{\phi}^{T\perp}_{X^m E} \middle| \phi_{X^m E} \right\rangle$. By construction, this state satisfies (A.3).

Furthermore, we can calculate the distance:

$$\delta \left(|\phi_{X^m E}\rangle\langle\phi_{X^m E}| \, , \left|\tilde{\phi}^T_{X^m E}\right\rangle\left\langle\tilde{\phi}^T_{X^m E}\right| \right) = \sqrt{1 - \left|\left\langle \tilde{\phi}^T_{X^m E} \middle| \phi_{X^m E} \right\rangle\right|^2} = \sqrt{1-|\epsilon_T|^2} = |\epsilon^\perp_T|,$$

where, given T, $|\epsilon^\perp_T|$ is the probability amplitude for getting outcome $\boldsymbol{r'} \notin \mathcal{B}_T$ when measuring the state of X^m in bases $\boldsymbol{x}$. We continue to derive an upper bound on the distance between the real and the ideal state:

$$\begin{aligned} \delta\big(\rho_{\mathcal{T}X^m E}, \sigma_{\mathcal{T}X^m E}\big) &= \left(\sum_T P_{\mathcal{T}}(T) \delta \left(|\phi_{X^m E}\rangle\langle\phi_{X^m E}| \, , \left|\tilde{\phi}^T_{X^m E}\right\rangle\left\langle\tilde{\phi}^T_{X^m E}\right| \right) \right)^2 \\ &\leq \sum_T P_{\mathcal{T}}(T) |\epsilon^\perp_T|^2, \end{aligned}$$

where we used Jensen's inequality and properties of the trace norm. The last term is the probability that, when choosing T according to $P_{\mathcal{T}}$ and measuring the state of X^m in bases $\boldsymbol{x}$ we get an outcome $\boldsymbol{r'} \notin \mathcal{B}_T$. We write

$$\sum_T P_{\mathcal{T}}(T)|\epsilon^\perp_T|^2 = \Pr_{\mathcal{T}}[\boldsymbol{r'} \notin \mathcal{B}_T] = \Pr_{\mathcal{T}}[r_H(\boldsymbol{r'}_{|\bar{T}}, \boldsymbol{r}_{|\bar{T}}) - r_H(\boldsymbol{r'}_{|T}, \boldsymbol{r}_{|T}) > \zeta].$$

Then, we can use Lemma 2 which states that the above probability is negligible in n and gives us an upper bound for $\delta\big(\rho_{\mathcal{T}X^m E}, \sigma_{\mathcal{T}X^m E}\big)$. In particular, given the set $[m]$ with $(1+t)n$ elements, we apply the aforementioned corollary for a random subset $\mathcal{T}$ of size tn and its complement $\bar{\mathcal{T}}$ of size n. Denoting by $\mu_{\mathcal{T}}$ and $\mu_{\bar{\mathcal{T}}}$, respectively, the averages of these subsets, we obtain

$$\Pr[\mu_{\bar{\mathcal{T}}} - \mu_{\mathcal{T}} \geq \zeta] \leq \exp\left(-\frac{2\zeta^2 t^2 n^2}{(nt+1)(t+1)} \right).$$

Hence, we have:

$$\delta\big(\rho_{\mathcal{T}X^m E}, \sigma_{\mathcal{T}X^m E}\big) \leq \sum_T P_{\mathcal{T}}(T)|\epsilon^\perp_T|^2 \leq \exp\left(-\frac{2\zeta^2 t^2 n^2}{(nt+1)(t+1)} \right) =: \epsilon, \tag{A.4}$$

concluding the proof of Lemma 17, i.e. that the real state (A.1) generated by the protocol until the *Computation Phase* is $\epsilon-$close to the ideal state (A.2). □

It is now straightforward to complete the proof of property 1. of Lemma 10.

To obtain the states $\sigma_{\mathbf{F}B'}$ and $\rho_{\mathbf{F}B'}$ from the states $\sigma_{\mathcal{T}X^m E}$ and $\rho_{\mathcal{T}X^m E}$, respectively, we need to apply the operator $V^{\boldsymbol{b}}_{\boldsymbol{a}} \operatorname{Tr}_{X^m_{|T} E_{|T}}[\,\cdot\,] V^{\boldsymbol{b}\dagger}_{\boldsymbol{a}}$. This operator is a CPTP map, being the composition of the two CPTP maps, $V_{\boldsymbol{a}}^{\boldsymbol{b}}$ and Tr. Since the trace distance between two density matrices does not increase under CPTP maps (see Lemma 7 in [70]), the final states indeed satisfy property 1. of Lemma 10, namely

$$\sigma_{\mathbf{F}B'} \approx_\epsilon \rho_{\mathbf{F}B'}.$$

For the rest of this proof dishonest Bob's system B' is identified with $\mathbf{Y}\,E$, where $\mathbf{Y}$ corresponds to the classical information leaking to him through the output of the WROLE and E is, in general, a quantum auxiliary system that he might also hold. Consequently, from now on we write $\sigma_{\mathbf{F}B'}$ as $\sigma_{\mathbf{FY}E}$. Now, we have to prove property 2. of Lemma 10, i.e., obtain the corresponding lower bound on min-entropy with respect to $\sigma_{\mathbf{FY}E}$. We start with the following Lemma 18:

Lemma 18 (Corollary 4.4 in [29])**.** *Let $err := r_H(\boldsymbol{r'}_{|T}, \boldsymbol{r}_{|T}) \leq 1 - \frac{1}{d}$ be the error measured by Alice while measuring the state of $X^m_{|T}$ according to her choice of T, and let $\sigma_{\mathbf{X}E} := |\psi\rangle\langle\psi|_{\mathbf{X}E}$ be the state to which the ideal state $\sigma_{\mathcal{T}X^mE}$ collapses after this measurement. Following* (A.2) *and* (A.3)*, we write $|\psi\rangle_{\mathbf{X}E} = \sum_{\boldsymbol{z}\in\mathcal{B}} \alpha_{\boldsymbol{z}} |e^{\boldsymbol{x}}_{\boldsymbol{z}}\rangle |\psi^{\boldsymbol{z}}_E\rangle$ for some $|\psi^{\boldsymbol{z}}_E\rangle$ and $\mathcal{B} = \{\boldsymbol{z} \in \mathbb{Z}^n_d : r_H(\boldsymbol{z}, \boldsymbol{r}_{|\bar{T}}) \leq err + \zeta\}$ with $\zeta > 0$. Then, we have:*

$$H_{\min}(\mathbf{X}|E)_{\sigma_{\mathbf{X}E}} \geq -h_d(err + \zeta) n \log d, \tag{A.5}$$

where $h_d(x)$ is given in Definition 1.

Proof. We start by defining the state $\tilde{\sigma}_{\mathbf{X}E} := \sum_{\boldsymbol{z}\in\mathcal{B}} |\alpha_{\boldsymbol{z}}|^2 |e^{\boldsymbol{x}}_{\boldsymbol{z}}\rangle\langle e^{\boldsymbol{x}}_{\boldsymbol{z}}| \otimes |\psi^{\boldsymbol{z}}_E\rangle\langle\psi^{\boldsymbol{z}}_E|$. Then, by applying Lemma 5, we obtain

$$H_{\min}(\mathbf{X}|E)_{\sigma_{\mathbf{X}E}} \geq H_{\min}(\mathbf{X}|E)_{\tilde{\sigma}_{\mathbf{X}E}} - \log|\mathcal{B}|.$$

Since $\tilde{\sigma}_{\mathbf{X}E}$ is a classical-quantum state, its min-entropy cannot be negative, i.e.

$$H_{\min}(\mathbf{X}|E)_{\tilde{\sigma}_{\mathbf{X}E}} \geq 0,$$

thus $H_{\min}(\mathbf{X}|E)_{\sigma_{\mathbf{X}E}} \geq -\log|\mathcal{B}|$.

Finally, to get the lower bound shown in (A.5), we apply Lemma 3 to our case, namely for the Hamming ball around $\boldsymbol{r}_{|\bar{T}}$ with radius $n(err + \zeta)$.

□

To complete the proof of property 2. of Lemma 10 and find a lower bound on $H_{\min}(\mathbf{F}|\mathbf{Y}\,E)\sigma_{\mathbf{FY}E}$ we need to relate it with $H_{\min}(\mathbf{X}|E)_{\sigma_{\mathbf{X}E}}$, for which we just derived a lower bound. In what follows, we adapt the notation from [73]:

- $\Phi_{\mathbf{X}\bar{\mathbf{X}}} = |\Phi\rangle\langle\Phi|_{\mathbf{X}\bar{\mathbf{X}}}$, where $|\Phi\rangle_{\mathbf{X}\bar{\mathbf{X}}} = \sum_{\boldsymbol{s}} |e_{\boldsymbol{s}}^{\boldsymbol{x}}\rangle_{\mathbf{X}} \otimes |e_{\boldsymbol{s}}^{\boldsymbol{x}}\rangle_{\bar{\mathbf{X}}}$ for all basis choices $\boldsymbol{x} \in \mathbb{Z}_d^n$, and

- $\Phi_{(\boldsymbol{a},\boldsymbol{b})} = |\Phi_{(\boldsymbol{a},\boldsymbol{b})}\rangle\langle\Phi_{(\boldsymbol{a},\boldsymbol{b})}| = \sum_{\boldsymbol{s}\boldsymbol{s'}} V_{\boldsymbol{a}}^{\boldsymbol{b}} |e_{\boldsymbol{s}}^{\boldsymbol{x}}\rangle\langle e_{\boldsymbol{s'}}^{\boldsymbol{x}}| V_{\boldsymbol{a}}^{\boldsymbol{b}\dagger} \otimes |e_{\boldsymbol{s}}^{\boldsymbol{x}}\rangle\langle e_{\boldsymbol{s'}}^{\boldsymbol{x}}|$, with $|\Phi_{(\boldsymbol{a},\boldsymbol{b})}\rangle = (V_{\boldsymbol{a}}^{\boldsymbol{b}} \otimes \mathbb{1}) |\Phi\rangle_{\mathbf{X}\bar{\mathbf{X}}}$, for $(\boldsymbol{a},\boldsymbol{b}) \in \mathbb{Z}_d^2$

and we use the following properties of the operators V_a^b, which can be derived from (5.2):

1. $V_a^b |e_i^x\rangle\langle e_i^x| V_a^{b\dagger} = |e_{ax-b+i}^x\rangle\langle e_{ax-b+i}^x|$,
2. $V_a^b = \sum_j c_{a,b,j,x} |e_{ax-b+j}^x\rangle\langle e_j^x|$,
3. $V_a^{b\dagger} = \sum_j c_{a,b,j,x}^* |e_j^x\rangle\langle e_{ax-b+j}^x|$.

The following lemma, which is an adaptation of Theorem 12 in [73] to our case, provides a lower bound for $H_{\min}(\mathbf{F}|\mathbf{Y}\,E)_{\sigma_{\mathbf{FY}E}}$ in terms of $H_{\min}(\mathbf{X}|E)_{\sigma_{\mathbf{X}E}}$:

Lemma 19. *Let* $\mathbf{X}$ *denote our* n*-qudit system and* $\sigma_{\mathbf{X}E}$ *be the ideal quantum state to which the system collapsed after Alice's test measurements, as introduced before. Then, we have:*

$$H_{min}(\mathbf{F}|\mathbf{Y}\,E)_{\sigma_{\mathbf{FY}E}} \geq \frac{1}{2}\big(n\log d + H_{min}(\mathbf{X}|E)_{\sigma_{\mathbf{X}E}}\big),$$

where

$$\sigma_{\mathbf{FY}E} = \frac{1}{d^{2n}} \sum_{(\boldsymbol{a},\boldsymbol{b})\in\mathbb{Z}_d^{2n}} |e_{\boldsymbol{a}}^{\boldsymbol{x}}, e_{\boldsymbol{b}}^{\boldsymbol{x}}\rangle\langle e_{\boldsymbol{a}}^{\boldsymbol{x}}, e_{\boldsymbol{b}}^{\boldsymbol{x}}| \otimes V_{\boldsymbol{a}}^{\boldsymbol{b}}\,\sigma_{\mathbf{X}E}\,V_{\boldsymbol{a}}^{\boldsymbol{b}\dagger}, \tag{A.6}$$

is the state obtained when $V_{\boldsymbol{a}}^{\boldsymbol{b}}$ *is applied to the system* $\mathbf{X}$ *according to* $\mathbf{F}$.

Proof. This proof is an adaptation of the proof of Theorem 12 in [73] to our case. Let us fix $x \in \mathbb{Z}_d$ and write $|i\rangle = |e_i^x\rangle$ for short. V_a^b is a CPTP map, and it is known that the min-entropy does not decrease whenever a CPTP map is applied. However, this is not enough to prove the security of the protocol and determine a lower bound. We need a more refined expression relating $H_{\min}(\mathcal{M}(\mathbf{X})|E)$ and $H_{\min}(\mathbf{X}|E)$ for some CPTP map $\mathcal{M}$. This is given by the following Lemma 20, which is an adaptation of Theorem 1 in [73] to our case:

Lemma 20 (Theorem 1, [73])**.** *Let* $\mathbf{X}$ *denote our system of* n *qudits, and* $\mathcal{M}_{\mathbf{X}\to\mathbf{FY}}$ *be a CP map such that* $((\mathcal{M}^\dagger \circ \mathcal{M})_{\mathbf{X}} \otimes id_{\bar{\mathbf{X}}})(\Phi_{\mathbf{X}\bar{\mathbf{X}}}) = \sum_{(\boldsymbol{a},\boldsymbol{b})\in\mathbb{Z}_d^{2n}} \lambda_{(\boldsymbol{a},\boldsymbol{b})}\Phi_{(\boldsymbol{a},\boldsymbol{b})}$*. As above, let* $\sigma_{\mathbf{X}E}$ *be the ideal quantum state to which the system collapsed after Alice's test measurements. Then, for any partition of* $\mathbb{Z}_d^{2n} = \mathfrak{S}_+ \cup \mathfrak{S}_-$ *into subsets* $\mathfrak{S}_+$ *and* $\mathfrak{S}_-$*, we have*

$$2^{-H_2(\mathbf{FY}|E)_{\sigma_{\mathbf{FY}E}|\sigma_{\mathbf{X}E}}} \leq \sum_{(\boldsymbol{a},\boldsymbol{b})\in\mathfrak{S}_+} \lambda_{(\boldsymbol{a},\boldsymbol{b})} 2^{-H_2(\mathbf{X}|E)_{\sigma_{\mathbf{X}E}}} + (\max_{(\boldsymbol{a},\boldsymbol{b})\in\mathfrak{S}_-} \lambda_{(\boldsymbol{a},\boldsymbol{b})}) d^n, \tag{A.7}$$

where, in general, for a (not necessarily normalized) quantum state $\rho_{AB} \in \mathcal{P}(\mathcal{H}_A \otimes \mathcal{H}_B)$, $H_2(A|B)$ is the so-called collision entropy [74], given as

$$H_2(A|B)_{\rho_{AB}} = -\log\left(\mathrm{Tr}\left\{\left(\rho_B^{-1/4}\rho_{AB}\rho_B^{-1/4}\right)^2\right\}\right).$$

If we further condition on a general quantum state $\sigma_B \in \mathcal{P}(\mathcal{H}_B)$, we have

$$H_2(A|B)_{\rho_{AB}|\sigma_B} = -\log\left(\mathrm{Tr}\left\{\left(\sigma_B^{-1/4}\rho_{AB}\sigma_B^{-1/4}\right)^2\right\}\right).$$

We can apply the above Lemma 20 with $\mathcal{M}_{\mathbf{X}\to\mathbf{FY}} = \mathcal{N}^{\otimes n}$, where $\mathcal{N}(\sigma_X) = \frac{1}{d^2}\sum_{(a,b)\in\mathbb{Z}_d^2}\mathcal{N}_{a,b}\sigma_X\mathcal{N}_{a,b}^\dagger = \frac{1}{d^2}\sum_{(a,b)\in\mathbb{Z}_d^2}\left(|a,b\rangle\otimes V_a^b\right)\sigma_X\left(\langle a,b|\otimes V_a^{b\dagger}\right)$. The operator $\mathcal{N}$ applies the operator V_a^b to the single system X and saves its choice (a,b) to a new record in the F space.

To continue, we want to write $((\mathcal{M}^\dagger\circ\mathcal{M})_{\mathbf{X}}\otimes\mathrm{id}_{\bar{\mathbf{X}}})(\Phi_{\mathbf{X}\bar{\mathbf{X}}})$ as a linear combination of $\Phi_{(\boldsymbol{a},\boldsymbol{b})}$, i.e.

$$((\mathcal{M}^\dagger\circ\mathcal{M})_{\mathbf{X}}\otimes\mathrm{id}_{\bar{\mathbf{X}}})(\Phi_{\mathbf{X}\bar{\mathbf{X}}}) = \sum_{(\boldsymbol{a},\boldsymbol{b})\in\mathbb{Z}^{2n}}\lambda_{(\boldsymbol{a},\boldsymbol{b})}\Phi_{(\boldsymbol{a},\boldsymbol{b})}.$$

First, note that

$$\begin{aligned}\mathcal{N}(|i\rangle\langle j|) &= \frac{1}{d^2}\sum_{(a,b)\in\mathbb{Z}_d^2}\left(|a,b\rangle\otimes V_a^b\right)|i\rangle\langle j|\left(\langle a,b|\otimes V_a^{b\dagger}\right)\\ &\overset{1}{=} \frac{1}{d^2}\sum_{(a,b)\in\mathbb{Z}_d^2}|a,b\rangle\langle a,b|\otimes|ax-b+i\rangle\langle ax-b+j|,\end{aligned}\tag{A.8}$$

where we used the property 1. We proceed to compute $\mathcal{N}^\dagger\circ\mathcal{N}\,|i\rangle\langle j|$:

$$\begin{aligned}\mathcal{N}^\dagger\circ\mathcal{N}\,|i\rangle\langle j| &= \frac{1}{d^2}\sum_{(a',b')\in\mathbb{Z}_d^2}\mathcal{N}_{a'b'}^\dagger(\mathcal{N}\,|i\rangle\langle j|)\mathcal{N}_{a'b'}\\ &\overset{\text{eq.(A.8)}}{=} \frac{1}{d^4}\sum_{(a',b'),(a,b)\in\mathbb{Z}_d^2}\left(\langle a',b'|\otimes V_{a'}^{b'\dagger}\right)|a,b\rangle\langle a,b|\\ &\qquad\qquad\otimes|ax-b+i\rangle\langle ax-b+j|\left(|a',b'\rangle\otimes V_{a'}^{b'}\right)\\ &\overset{2,3}{=} \frac{1}{d^4}\sum_{(a,b)\in\mathbb{Z}_d^2}V_a^{b\dagger}\,|ax-b+i\rangle\langle ax-b+j|\,V_a^b\\ &= \frac{1}{d^4}\sum_{(a,b)\in\mathbb{Z}_d^2}|i\rangle\langle j|\\ &= \frac{1}{d^2}\,|i\rangle\langle j|,\end{aligned}\tag{A.9}$$

and

$$
\begin{aligned}
((\mathcal{N}^\dagger \circ \mathcal{N})_X \otimes \mathrm{id}_{\bar{X}})(\Phi_{X\bar{X}}) &= \sum_{i,j} \mathcal{N}^\dagger \circ \mathcal{N}(|i\rangle\langle j|) \otimes |i\rangle\langle j| \\
&\overset{\text{eq.}(A.9)}{=} \sum_{i,j} \left(\frac{1}{d^2} |i\rangle\langle j| \right) \otimes |i\rangle\langle j| \\
&= \frac{1}{d^2} \sum_{i,j} |i\rangle\langle j| \otimes |i\rangle\langle j| \\
&= \frac{1}{d^2} \Phi_{X\bar{X}}. \qquad \text{(A.10)}
\end{aligned}
$$

Therefore, we have that

$$
\big((\mathcal{N}^\dagger \circ \mathcal{N})_X \otimes \mathrm{id}_{\bar{X}}\big)(\Phi_{X\bar{X}}) = \frac{1}{d^2} \Phi_{(0,0)}, \qquad \text{(A.11)}
$$

from which we easily see that

$$
\big((\mathcal{M}^\dagger \circ \mathcal{M})_{\mathbf{X}} \otimes \mathrm{id}_{\bar{\mathbf{X}}}\big)(\Phi_{\mathbf{X}\bar{\mathbf{X}}}) = \frac{1}{d^{2n}} \Phi_{(\mathbf{0},\mathbf{0})}. \qquad \text{(A.12)}
$$

Consequently,

$$
\lambda_{(\boldsymbol{a},\boldsymbol{b})} = \begin{cases} \frac{1}{d^{2n}} & \text{if } (\boldsymbol{a},\boldsymbol{b}) = (\mathbf{0},\mathbf{0}) \\ 0 & \text{otherwise.} \end{cases}
$$

Now, we want to choose the partition that gives us the best lower bound on the collision entropy, i.e. decreases the r.h.s of the following relation:

$$
2^{-\mathrm{H}_2(\mathbf{FY}|E)_{\sigma_{\mathbf{FY}E}}} \leq \sum_{(\boldsymbol{a},\boldsymbol{b})\in\mathfrak{S}_+} \lambda_{(\boldsymbol{a},\boldsymbol{b})} 2^{-\mathrm{H}_2(\mathbf{X}|E)_{\sigma_{\mathbf{X}E}}} + (\max_{(\boldsymbol{a},\boldsymbol{b})\in\mathfrak{S}_-} \lambda_{(\boldsymbol{a},\boldsymbol{b})}) d^n.
$$

Note that we dropped the conditioning on the state $\sigma_{\mathbf{X}E}$ at $\mathrm{H}_2(\mathbf{FY}|E)_{\sigma_{\mathbf{FY}E}}$. This is because the map $\mathcal{M}$ is trace-preserving (for a detailed explanation see [73] below Theorem 1). For our case there are just two types of partitions: the case where $0 \in \mathfrak{S}_+$ and the case where $0 \in \mathfrak{S}_-$. If $0 \in \mathfrak{S}_+$:

$$
\text{r.h.s} = \sum_{(\boldsymbol{a},\boldsymbol{b})\in\mathfrak{S}_+} \lambda_s 2^{-\mathrm{H}_2(\mathbf{X}|E)_{\sigma_{\mathbf{X}E}}} = \frac{1}{d^{2n}} 2^{-\mathrm{H}_2(\mathbf{X}|E)_{\sigma_{\mathbf{X}E}}} \leq \frac{1}{d^n}.
$$

The last inequality holds because $-n\log d \leq \mathrm{H}_2(\mathbf{X}|E)_{\sigma_{\mathbf{X}E}} \leq n\log d$. In fact,

$$
\frac{1}{d^{2n}} 2^{-\mathrm{H}_2(\mathbf{X}|E)_{\sigma_{\mathbf{X}E}}} \leq \frac{1}{d^n} \iff 2^{-\mathrm{H}_2(\mathbf{X}|E)_{\sigma_{\mathbf{X}E}}} \leq d^n \iff \mathrm{H}_2(\mathbf{X}|E)_{\sigma_{\mathbf{X}E}} \geq -n\log d.
$$

If $0 \in \mathfrak{S}_-$, r.h.s $= \frac{1}{d^n}$ which, as we have just seen by the previous inequality, does not provide a better lower bound on the collision entropy whenever $\mathrm{H}_2(\mathbf{X}|E)_{\sigma_{\mathbf{X}E}} \neq -n\log d$.

So, choosing any partition such that $0 \in \mathfrak{S}_+$, we get

$$2^{-\mathrm{H}_2(\mathbf{FY}|E)_{\sigma_{\mathbf{FY}E}}} \leq \sum_{(\boldsymbol{a},\boldsymbol{b})\in\mathfrak{S}_+} \lambda_{(\boldsymbol{a},\boldsymbol{b})} 2^{-\mathrm{H}_2(\mathbf{X}|E)_{\sigma_{\mathbf{X}E}}} + (\max_{(\boldsymbol{a},\boldsymbol{b})\in\mathfrak{S}_-} \lambda_{(\boldsymbol{a},\boldsymbol{b})}) d^n = \frac{1}{d^{2n}} 2^{-\mathrm{H}_2(\mathbf{X}|E)_{\sigma_{\mathbf{X}E}}},$$

from which we conclude that

$$\mathrm{H}_2(\mathbf{FY}|E)_{\sigma_{\mathbf{FY}E}} \geq 2n\log d + \mathrm{H}_2(\mathbf{X}|E)_{\sigma_{\mathbf{X}E}}. \tag{A.13}$$

In order to relate $\mathrm{H}_2(\mathbf{FY}|E)_{\sigma_{\mathbf{FY}E}}$ with $\mathrm{H}_2(\mathbf{F}|\mathbf{Y}E)_{\sigma_{\mathbf{FY}E}}$, we use the chain rule proved in Proposition 8 of [?]:

$$\mathrm{H}_2(\mathbf{F}|\mathbf{Y}E)_{\sigma_{\mathbf{FY}E}} \geq \mathrm{H}_2(\mathbf{FY}|E)_{\sigma_{\mathbf{FY}E}} - \log\mathrm{rank}(\sigma_{\mathbf{Y}}) \geq \mathrm{H}_2(\mathbf{FY}|E)_{\sigma_{\mathbf{FY}E}} - n\log d, \tag{A.14}$$

since $\log\mathrm{rank}(\sigma_{\mathbf{Y}}) \leq n\log d$. Combining (A.13) and (A.14), we get the desired result:

$$\mathrm{H}_2(\mathbf{F}|\mathbf{Y}\,E)_{\sigma_{\mathbf{FY}E}} \geq n\log d + \mathrm{H}_2(\mathbf{X}|E)_{\sigma_{\mathbf{X}E}}.$$

To express the above relation in terms of the min-entropy instead of the collision entropy, we start by noticing that the state given in (A.6) can be written as

$$\sigma_{\mathbf{FY}E} = \frac{1}{d^{2n}} \sum_{(\boldsymbol{a},\boldsymbol{b})\in\mathbb{Z}_d^{2n}} |\boldsymbol{a},\boldsymbol{b}\rangle\langle\boldsymbol{a},\boldsymbol{b}| \otimes V_{\boldsymbol{a}}^{\boldsymbol{b}}\,\sigma_{\mathbf{X}E}\,V_{\boldsymbol{a}}^{\boldsymbol{b}\dagger} = \frac{1}{d^{2n}} \sum_{(\boldsymbol{a},\boldsymbol{b})\in\mathbb{Z}_d^{2n}} |\boldsymbol{a},\boldsymbol{b}\rangle\langle\boldsymbol{a},\boldsymbol{b}| \otimes \sigma_{\mathbf{X}E}^{\boldsymbol{a},\boldsymbol{b}}, \tag{A.15}$$

which is a classical-quantum state. Therefore, we can use Lemma 18 from [73] to obtain

$$\mathrm{H}_{\min}(\mathbf{F}|\mathbf{Y}\,E)_{\sigma_{\mathbf{FY}E}} \geq \frac{1}{2}\left(n\log d + \mathrm{H}_2(\mathbf{X}|E)_{\sigma_{\mathbf{X}E}}\right).$$

Furthermore, $\sigma_{\mathbf{X}E}$ is a general quantum state, and from Lemma 17 in [73] we have

$$\mathrm{H}_{\min}(\mathbf{F}|\mathbf{Y}\,E)_{\sigma_{\mathbf{FY}E}} \geq \frac{1}{2}\left(n\log d + \mathrm{H}_{\min}(\mathbf{X}|E)_{\sigma_{\mathbf{X}E}}\right).$$

□

To complete the proof of property 2. of Lemma 10, we combine Lemma 18 and Lemma 19 and obtain:

$$H_{\min}(\mathbf{F}|\mathbf{Y}E)_{\sigma_{\mathbf{FY}E}} \geq \frac{1}{2}(n\log d - h_d(\zeta)n\log d) = \frac{n\log d}{2}(1 - h_d(\zeta)),$$

for $err = 0$.

□

Appendix B

Jukes-Cantor distance for CBMC-GC

The boolean circuit that represents the Jukes-Cantor distance receives as inputs two four-based sequences (A, C, G, T) with size 32 000. Since we are using a boolean circuit representation, the nucleotide sequences must be represented in binary. So, by convention, we use the following 2-bit encoding: $A = 00$, $C = 01$, $G = 10$ and $T = 11$. As a result, we start by defining a sequence type of size 4 000 with the `unsigned short` type elements (Figure B.1, lines $1-5$). In fact, the type `Array_Seq` saves $4\,000 \times 16 = 64\,000$ bits. Each element of `Array_Seq` represents a small sequence of eight elements. This is an implementation choice that renders a good compromise between accuracy level and circuit size.

As we saw in the main text, the hamming distance between two binary strings can then be easily computed by XORing them and counting the number of 1's. This last operation is commonly called `popcount`. We cannot directly apply this approach because our sequences are in fact four-based sequences. In fact, our version of the `popcount` function is only interested in computing the number of 2-bit elements that are different between both sequences.

We follow a tailored divide-and-conquer technique. The original technique is described by Henry Warren in his book "Hacker's Delight", chapter 5 [253]. In summary, the original technique starts by counting in parallel the number of 1's inside each 2-bit block and saves it in 2-bit blocks. Then, it adds two 2-bit blocks and saves the result in a 4-bit block. It continues until we get the final sum. If we follow directly this approach we might run into wrong results as described in the main text. For our case, instead of counting the number of 1's inside every 2-bit block, we only care if there is one element 1 inside each 2-bit block. This simply indicates that the elements at that site are different. This is achieved by applying an `OR` operation (represented by `|`) to the bits inside each 2-bit block (Figure B.1, line 14). For 4-bit blocks and above we follow the same recipe of the original divide-and-conquer technique.

The main function that computes the Hamming weight between two nucleotide sequences `INPUT_A` and `INPUT_B` is defined by the function `mpc_main`, line 21. It outputs the inverse of the hamming weight: `total/distance` line 37. Since we know the hamming weight lies between 0 and 1, it renders smaller circuits to use the native integer division operator, /, from the CBMC-GC tool and then invert the output after the Yao computation. Otherwise, we would need a fixed precision representation to output decimal numbers.

Below we describe the variables used in the `mpc_main` function:

- `INPUT_A` and `INPUT_B`: the binary input sequences of Alice and Bob, respectively. Following the CBMC-GC convention, the input elements must start with the identifier `INPUT_`.

- `OUTPUT_distance`: the inverse of the hamming weight. Following the CBMC-GC convention, the output element must start with the identifier `OUTPUT_`.

- `total`: keeps track of the number of elements that can be compared between aligned sequences.

- `distance`: keeps track of the hamming distance between both sequences.

- `count_axorb`: saves the number of elements that are different in a 16-bit block sequence (i.e. in `INPUT_A.el[i]^INPUT_B.el[i]`).

```c
#define LEN_SEQ 4000

typedef struct {
  unsigned short el[LEN_SEQ];
} Array_Seq;

const unsigned int m1  = 0x55555555; //binary: 0101...
const unsigned int m2  = 0x33333333; //binary: 00110011..
const unsigned int m4  = 0x0f0f0f0f; //binary: 4 zeros, 4 ones ...
const unsigned int m8  = 0x00ff00ff; //binary: 8 zeros, 8 ones ...

unsigned int popcount(unsigned short INPUT_B_x) {
    unsigned int x = INPUT_B_x;
    x = (x & m1 ) | ((x >>  1) & m1 ); // changed step
    x = (x & m2 ) + ((x >>  2) & m2 );
    x = (x & m4 ) + ((x >>  4) & m4 );
    x = (x & m8 ) + ((x >>  8) & m8 );
    return x;
}

void mpc_main(Array_Seq INPUT_A, Array_Seq INPUT_B){
  unsigned int distance = 0;
  int total = 0;
  for(int i=0; i<LEN_SEQ; i++){
    int count_a = popcount(INPUT_A.el[i]);
    int count_b = popcount(INPUT_B.el[i]);
    if(count_a > 0 && count_b > 0){
      int count_axorb = popcount(INPUT_A.el[i]^INPUT_B.el[i]);
      if(count_axorb == 1){
        distance = distance + 1;
      }
      total = total + 8;
    }
  }
  unsigned int OUTPUT_distance;
  if(distance > 0){
    OUTPUT_distance = total/distance;
  } else {
    OUTPUT_distance = 0;
  }
}
```

Figure B.1: Jukes-Cantor distance C code for CBMC-GC boolean circuit generation.

Bibliography

[1] Vladimir Kolesnikov, Ahmad-Reza Sadeghi, and Thomas Schneider. Improved garbled circuit building blocks and applications to auctions and computing minima. In *Cryptology and Network Security*, pages 1–20. Springer Berlin Heidelberg, 2009.

[2] Mihir Bellare and Silvio Micali. Non-interactive oblivious transfer and applications. In *Proceedings on Advances in Cryptology*, CRYPTO -89, pages 547–557, Berlin, Heidelberg, 1989. Springer-Verlag.

[3] Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. More efficient oblivious transfer and extensions for faster secure computation. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, pages 535–548, New York, NY, USA, 2013. Association for Computing Machinery.

[4] Marcel Keller, Emmanuela Orsini, and Peter Scholl. Actively secure ot extension with optimal overhead. In *Advances in Cryptology - CRYPTO 2015*, volume 9215 of *Lecture Notes in Computer Science*, pages 724–741. Springer, August 2015. Date of Acceptance: 08/05/2015.

[5] Jun Wang. Personal genomes: For one and for all. *Science*, 331(6018):690–690, 2011.

[6] Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large sparse datasets. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pages 111–125, 2008.

[7] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.

[8] Nils Homer, Szabolcs Szelinger, Margot Redman, David Duggan, Waibhav Tembe, Jill Muehling, John V. Pearson, Dietrich A. Stephan, Stanley F. Nelson, and David W. Craig. Resolving individuals contributing trace amounts of DNA to highly

complex mixtures using high-density SNP genotyping microarrays. *PLoS Genetics*, 4(8):e1000167, August 2008.

[9] M. Gymrek, A. L. McGuire, D. Golan, E. Halperin, and Y. Erlich. Identifying personal genomes by surname inference. *Science*, 339(6117):321–324, January 2013.

[10] 2016 reform of eu data protection rules. https://eur-lex.europa.eu/eli/reg/2016/679/oj, 2016.

[11] Ninghui Li, Min Lyu, Dong Su, and Weining Yang. Differential privacy: From theory to practice. *Synthesis Lectures on Information Security, Privacy, and Trust*, 8(4):1–138, October 2016.

[12] Frederik Armknecht, C. Boyd, Christopher Carr, K. Gjøsteen, Angela Jäschke, Christian A. Reuter, and Martin Strand. A guide to fully homomorphic encryption. *IACR Cryptol. ePrint Arch.*, 2015:1192, 2015.

[13] A. C. Yao. Protocols for secure computations. In *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*, pages 160–164, 1982.

[14] Andrew Chi-Chih Yao. How to generate and exchange secrets. In *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*. IEEE, October 1986.

[15] O. Goldreich, S. Micali, and A. Wigderson. How to play ANY mental game. In *Proceedings of the nineteenth annual ACM conference on Theory of computing - STOC '87*. ACM Press, 1987.

[16] Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. Semi-homomorphic encryption and multiparty computation. In *Advances in Cryptology – EUROCRYPT 2011*, pages 169–188. Springer Berlin Heidelberg, 2011.

[17] Ivan Damgård, Valerio Pastro, Nigel Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In *Lecture Notes in Computer Science*, pages 643–662. Springer Berlin Heidelberg, 2012.

[18] Joe Kilian. Founding cryptography on oblivious transfer. In *Proceedings of the twentieth annual ACM symposium on Theory of computing - STOC '88*. ACM Press, 1988.

[19] Ivan Damgård, Valerio Pastro, Nigel Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In *Lecture Notes in Computer Science*, pages 643–662. Springer Berlin Heidelberg, 2012.

[20] R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, STOC '89, page 44–61, New York, NY, USA, 1989. Association for Computing Machinery.

[21] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.

[22] Anne Broadbent and Christian Schaffner. Quantum cryptography beyond quantum key distribution. *Designs, Codes and Cryptography*, 78(1):351–382, December 2015.

[23] A. N. Pinto, N. A. Silva, A. Almeida, and N. J. Muga. Using quantum technologies to improve fiber optic communication systems. *IEEE Communications Magazine*, 8(51):42–48, August 2013.

[24] Andre Chailloux and Iordanis Kerenidis. Optimal bounds for quantum bit commitment. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*. IEEE, October 2011.

[25] André Chailloux and Iordanis Kerenidis. Optimal quantum strong coin flipping. In *2009 50th Annual IEEE Symposium on Foundations of Computer Science*. IEEE, October 2009.

[26] Anne Broadbent, Joseph Fitzsimons, and Elham Kashefi. Universal blind quantum computation. In *2009 50th Annual IEEE Symposium on Foundations of Computer Science*. IEEE, October 2009.

[27] Dominique Unruh. Quantum position verification in the random oracle model. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014*, pages 1–18, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[28] Ivan Damgård, Serge Fehr, Louis Salvail, and Christian Schaffner. Secure identification and QKD in the bounded-quantum-storage model. *Theoretical Computer Science*, 560:12–26, December 2014.

[29] Ivan Damgård, Serge Fehr, Carolin Lunemann, Louis Salvail, and Christian Schaffner. Improving the security of quantum protocols via commit-and-open. In Shai Halevi, editor, *Advances in Cryptology - CRYPTO 2009*, pages 408–427, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[30] Charles H. Bennett, Gilles Brassard, Claude Crépeau, and Marie-Hélène Skubiszewska. Practical quantum oblivious transfer. In Joan Feigenbaum, editor, *Advances in Cryptology — CRYPTO '91*, pages 351–366, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg.

[31] Alex B. Grilo, Huijia Lin, Fang Song, and Vinod Vaikuntanathan. Oblivious transfer is in miniqcrypt. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021*, pages 531–561, Cham, 2021. Springer International Publishing.

[32] James Bartusek, Andrea Coladangelo, Dakshita Khurana, and Fermi Ma. One-way functions imply secure computation in a quantum world. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021*, pages 467–496, Cham, 2021. Springer International Publishing.

[33] Marcel Keller, Emmanuela Orsini, and Peter Scholl. MASCOT. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, October 2016.

[34] H. Bechmann-Pasquinucci and W. Tittel. Quantum cryptography using larger alphabets. *Phys. Rev. A*, 61:062308, May 2000.

[35] Thomas Durt, Berthold-Georg Englert, Ingemar Bengtsson, and Karol Zyczkowski. On mutually unbiased bases. *International Journal of Quantum Information*, 08(04):535–640, 2010.

[36] Tian Zhong, Hongchao Zhou, Robert D Horansky, Catherine Lee, Varun B Verma, Adriana E Lita, Alessandro Restelli, Joshua C Bienfang, Richard P Mirin, Thomas Gerrits, Sae Woo Nam, Francesco Marsili, Matthew D Shaw, Zheshen Zhang, Ligong Wang, Dirk Englund, Gregory W Wornell, Jeffrey H Shapiro, and Franco N C Wong. Photon-efficient quantum key distribution using time–energy entanglement with high-dimensional encoding. *New Journal of Physics*, 17(2):022002, 2015.

[37] Frédéric Bouchard, Natalia Herrera Valencia, Florian Brandt, Robert Fickler, Marcus Huber, and Mehul Malik. Measuring azimuthal and radial modes of photons. *Opt. Express*, 26(24):31925–31941, Nov 2018.

[38] Mirdit Doda, Marcus Huber, Gláucia Murta, Matej Pivoluska, Martin Plesch, and Chrysoula Vlachou. Quantum key distribution overcoming extreme noise: Simultaneous subspace coding using high-dimensional entanglement. *Phys. Rev. Applied*, 15:034003, Mar 2021.

[39] Manuel B. Santos, Paulo Mateus, and Armando N. Pinto. Quantum oblivious transfer: A short review. *Entropy*, 24(7):945, July 2022.

[40] Manuel B. Santos, Armando N. Pinto, and Paulo Mateus. Quantum and classical oblivious transfer: A comparative analysis. *IET Quantum Communication*, 2(2):42–53, May 2021.

[41] Manuel B. Santos, Paulo Mateus, and Chrysoula Vlachou. Quantum universally composable oblivious linear evaluation, 2022.

[42] 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.

[43] 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 *2021 Telecoms Conference (ConfTELE)*. IEEE, February 2021.

[44] 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 *2020 22nd International Conference on Transparent Optical Networks (ICTON)*. IEEE, July 2020.

[45] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, page 1–10, New York, NY, USA, 1988. Association for Computing Machinery.

[46] David Chaum, Claude Crépeau, and Ivan Damgard. Multiparty unconditionally secure protocols. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, page 11–19, New York, NY, USA, 1988. Association for Computing Machinery.

[47] Rushi Patel, Pierre-Francois Wolfe, Robert Munafo, Mayank Varia, and Martin Herbordt. Arithmetic and boolean secret sharing MPC on FPGAs in the data center. In *2020 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, September 2020.

[48] Michael O. Rabin. How to exchange secrets with oblivious transfer. Technical Report TR-81, Aiken Computation Laboratory, Harvard University, 1981.

[49] Yan-Cheng Chang. Single database private information retrieval with logarithmic communication. In Huaxiong Wang, Josef Pieprzyk, and Vijay Varadharajan, editors, *Information Security and Privacy*, pages 50–61, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[50] Michele Orrù, Emmanuela Orsini, and Peter Scholl. Actively secure 1-out-of-n ot extension with application to private set intersection. In Helena Handschuh, editor, *Topics in Cryptology – CT-RSA 2017*, pages 381–396, Cham, 2017. Springer International Publishing.

[51] Bo Bi, Darong Huang, Bo Mi, Zhenping Deng, and Hongyang Pan. Efficient LBS security-preserving based on NTRU oblivious transfer. *Wireless Personal Communications*, 108(4):2663–2674, May 2019.

[52] D. Beaver, S. Micali, and P. Rogaway. The round complexity of secure protocols. In *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing*, STOC '90, page 503–513, New York, NY, USA, 1990. Association for Computing Machinery.

[53] Moni Naor, Benny Pinkas, and Reuban Sumner. Privacy preserving auctions and mechanism design. In *Proceedings of the 1st ACM Conference on Electronic Commerce*, EC '99, page 129–139, New York, NY, USA, 1999. Association for Computing Machinery.

[54] Benny Pinkas, Thomas Schneider, Nigel P. Smart, and Stephen C. Williams. Secure two-party computation is practical. In *Advances in Cryptology – ASIACRYPT 2009*, pages 250–267. Springer Berlin Heidelberg, 2009.

[55] Vladimir Kolesnikov. Gate evaluation secret sharing and secure one-round two-party computation. In *Lecture Notes in Computer Science*, pages 136–155. Springer Berlin Heidelberg, 2005.

[56] Samee Zahur, Mike Rosulek, and David Evans. Two halves make a whole. In *Advances in Cryptology - EUROCRYPT 2015*, pages 220–250. Springer Berlin Heidelberg, 2015.

[57] Russel Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, STOC '89, page 44–61, New York, NY, USA, 1989. Association for Computing Machinery.

[58] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. How to garble arithmetic circuits. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 120–129, 2011.

[59] Nico Döttling, Daniel Kraschewski, and Jörn Müller-Quade. David and goliath oblivious affine function evaluation - asymptotically optimal building blocks for universally composable two-party computation from a single untrusted stateful tamper-proof hardware token. Cryptology ePrint Archive, Report 2012/135, 2012. `https://ia.cr/2012/135`.

[60] Satrajit Ghosh, Jesper Buus Nielsen, and Tobias Nilges. Maliciously secure oblivious linear function evaluation with constant overhead. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017*, pages 629–659, Cham, 2017. Springer International Publishing.

[61] Nico Döttling, Satrajit Ghosh, Jesper Buus Nielsen, Tobias Nilges, and Roberto Trifiletti. Tinyole: Efficient actively secure two-party computation from oblivious linear function evaluation. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, page 2263–2276, New York, NY, USA, 2017. Association for Computing Machinery.

[62] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, STOC '87, page 218–229, New York, NY, USA, 1987. Association for Computing Machinery.

[63] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Secure arithmetic computation with no honest majority. In Omer Reingold, editor, *Theory of Cryptography*, pages 294–314, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[64] Benny Applebaum, Ivan Damgård, Yuval Ishai, Michael Nielsen, and Lior Zichron. Secure arithmetic computation with constant computational overhead. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017*, pages 223–254, Cham, 2017. Springer International Publishing.

[65] Elette Boyle, Geoffroy Couteau, Niv Gilboa, and Yuval Ishai. Compressing vector OLE. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, January 2018.

[66] Carmit Hazay, Yuval Ishai, Antonio Marcedone, and Muthuramakrishnan Venkitasubramaniam. Leviosa: Lightweight secure arithmetic computation. CCS '19, page 327–344, New York, NY, USA, 2019. Association for Computing Machinery.

[67] Melissa Chase, Yevgeniy Dodis, Yuval Ishai, Daniel Kraschewski, Tianren Liu, Rafail Ostrovsky, and Vinod Vaikuntanathan. Reusable non-interactive secure computation. 11694:462–488, 2019.

[68] Satrajit Ghosh and Tobias Nilges. An algebraic approach to maliciously secure private set intersection. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019*, pages 154–185, Cham, 2019. Springer International Publishing.

[69] David Evans, Vladimir Kolesnikov, and Mike Rosulek. A pragmatic introduction to secure multi-party computation. *Foundations and Trends® in Privacy and Security*, 2(2-3):70–246, 2018.

[70] Dominique Unruh. Lecture notes in quantum cryptography, November 2017.

[71] Claude Elwood Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 1948.

[72] Venkatesan Guruswami. Lecture notes in introduction to coding theory, January 2010.

[73] Frederic Dupuis, Omar Fawzi, and Stephanie Wehner. Entanglement sampling and applications. *IEEE Transactions on Information Theory*, 61(2):1093–1112, February 2015.

[74] Renato Renner. Security of quantum key distribution, 2006.

[75] Shai Halevi and Hugo Krawczyk. MMH: Software message authentication in the gbit/second rates. In *Fast Software Encryption*, pages 172–189. Springer Berlin Heidelberg, 1997.

[76] Marco Tomamichel, Christian Schaffner, Adam Smith, and Renato Renner. Leftover hashing against quantum side information. *IEEE Transactions on Information Theory*, 57(8):5524–5535, August 2011.

[77] Ran Canetti. Universally composable security. *Journal of the ACM*, 67(5):1–94, October 2020.

[78] Dominique Unruh. Simulatable security for quantum protocols. *arXiv*, 0409125, 2004.

[79] Michael Ben-Or and Dominic Mayers. General security definition and composability for quantum and classical protocols. *arXiv*, 0409062, 2004.

[80] Dominique Unruh. Universally composable quantum multi-party computation. In *Advances in Cryptology – EUROCRYPT 2010*, pages 486–505, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[81] Serge Fehr and Christian Schaffner. Composing quantum protocols in a classical environment. In Omer Reingold, editor, *Theory of Cryptography*, pages 350–367, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[82] Bruno Costa, Pedro Branco, Manuel Goulão, Mariano Lemus, and Paulo Mateus. Randomized oblivious transfer for secure multiparty computation in the quantum setting. *Entropy*, 23(8):1001, July 2021.

[83] Vijay Kumar Yadav, Nitish Andola, Shekhar Verma, and S Venkatesan. A survey of oblivious transfer protocol. *ACM Computing Surveys*, January 2022.

[84] Y. Gertner, S. Kannan, T. Malkin, O. Reingold, and M. Viswanathan. The relationship between public key encryption and oblivious transfer. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*. IEEE Comput. Soc, 2000.

[85] Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. *Commun. ACM*, 28(6):637–647, June 1985.

[86] M. Naor and Benny Pinkas. Efficient oblivious transfer protocols. In *SODA '01*, 2001.

[87] Tung Chou and Claudio Orlandi. The simplest protocol for oblivious transfer. In *Proceedings of the 4th International Conference on Progress in Cryptology – LATINCRYPT 2015 - Volume 9230*, page 40–58, Berlin, Heidelberg, 2015. Springer-Verlag.

[88] Gilles Brassard and Claude Crépeau. 25 years of quantum cryptography. *ACM SIGACT News*, 27(3):13–24, September 1996.

[89] G. Brassard. Brief history of quantum cryptography: a personal perspective. In *IEEE Information Theory Workshop on Theory and Practice in Information-Theoretic Security.* IEEE, 2005.

[90] Jörn Müller-Quade. Quantum cryptography beyond key exchange. *Informatik - Forschung und Entwicklung*, 21(1-2):39–54, September 2006.

[91] Serge Fehr. Quantum cryptography. *Foundations of Physics*, 40(5):494–531, January 2010.

[92] S. Pirandola, U. L. Andersen, L. Banchi, M. Berta, D. Bunandar, R. Colbeck, D. Englund, T. Gehring, C. Lupo, C. Ottaviani, J. L. Pereira, M. Razavi, J. Shamsul Shaari, M. Tomamichel, V. C. Usenko, G. Vallone, P. Villoresi, and P. Wallden. Advances in quantum cryptography. *Advances in Optics and Photonics*, 12(4):1012, December 2020.

[93] Christopher Portmann and Renato Renner. Security in quantum cryptography, 2021.

[94] Shihai Sun and Anqi Huang. A review of security evaluation of practical quantum key distribution system. *Entropy*, 24(2):260, February 2022.

[95] Louis Salvail. *The Search for the Holy Grail in Quantum Cryptography*, pages 183–216. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999.

[96] Andrew Chi-Chih Yao. Security of quantum protocols against coherent measurements. In *Proceedings of the twenty-seventh annual ACM symposium on Theory of computing - STOC '95*. ACM Press, 1995.

[97] G. Brassard, C. Crepeau, R. Jozsa, and D. Langlois. A quantum bit commitment scheme provably unbreakable by both parties. In *Proceedings of 1993 IEEE 34th Annual Foundations of Computer Science*. IEEE, 1993.

[98] Dominic Mayers. The trouble with quantum bit commitment, 1996.

[99] Hoi-Kwong Lo and H. F. Chau. Is quantum bit commitment really possible? *Physical Review Letters*, 78(17):3410–3413, April 1997.

[100] Dominic Mayers. Unconditionally secure quantum bit commitment is impossible. *Physical Review Letters*, 78(17):3414–3417, April 1997.

[101] Hoi-Kwong Lo. Insecurity of quantum secure computations. *Physical Review A*, 56(2):1154–1162, August 1997.

[102] Roger Colbeck. Impossibility of secure two-party classical computation. *Physical Review A*, 76(6), December 2007.

[103] Harry Buhrman, Matthias Christandl, and Christian Schaffner. Complete insecurity of quantum protocols for classical two-party computation. *Physical Review Letters*, 109(16), October 2012.

[104] Louis Salvail, Christian Schaffner, and Miroslava Sotáková. Quantifying the leakage of quantum protocols for classical two-party cryptography. *International Journal of Quantum Information*, 13(04):1450041, December 2014.

[105] Serge Fehr, Jonathan Katz, Fang Song, Hong-Sheng Zhou, and Vassilis Zikas. Feasibility and completeness of cryptographic tasks in the quantum world. In Amit Sahai, editor, *Theory of Cryptography*, pages 281–296, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[106] Gilles Brassard, Claude Crépeau, Dominic Mayers, and Louis Salvail. A brief review on the impossibility of quantum bit commitment, 1997.

[107] Horace P. Yuen. Unconditionally secure quantum bit commitment is possible, 2000.

[108] Horace P. Yuen. Quantum bit commitment and unconditional security, 2002.

[109] Horace P. Yuen. How to build unconditionally secure quantum bit commitment protocols, 2003.

[110] Chi-Yee Cheung. Quantum bit commitment can be unconditionally secure, 2001.

[111] Jeffrey Bub. The quantum bit commitment theorem. *Foundations of Physics*, 31(5):735–756, 2001.

[112] Chi-Yee Cheung. Secret parameters in quantum bit commitment, 2005.

[113] CHI-YEE CHEUNG. Quantum bit commitment with secret parameters. *International Journal of Modern Physics B*, 21(23n24):4271–4274, September 2007.

[114] Giacomo Mauro D'Ariano, Dennis Kretschmann, Dirk Schlingemann, and Reinhard F. Werner. Reexamination of quantum bit commitment: The possible and the impossible. *Physical Review A*, 76(3), September 2007.

[115] Giulio Chiribella, Giacomo Mauro D'Ariano, and Paolo Perinotti. Probabilistic theories with purification. *Physical Review A*, 81(6), June 2010.

[116] Giulio Chiribella, Giacomo Mauro D'Ariano, Paolo Perinotti, Dirk Schlingemann, and Reinhard Werner. A short impossibility proof of quantum bit commitment. *Physics Letters A*, 377(15):1076–1087, June 2013.

[117] Guang Ping He. Comment on "a short impossibility proof of quantum bit commitment", 2013.

[118] Katriel Cohn-Gordon. Commitment algorithms. Master's thesis, University of Oxford, Oxford, UK, 2012.

[119] Xin Sun, Feifei He, and Quanlong Wang. Impossibility of quantum bit commitment, a categorical perspective. *Axioms*, 9(1):28, March 2020.

[120] Anne Broadbent and Martti Karvonen. Categorical composable cryptography. In Patricia Bouyer and Lutz Schröder, editors, *Foundations of Software Science and Computation Structures*, pages 161–183, Cham, 2022. Springer International Publishing.

[121] Scott Aaronson. Quantum lower bound for the collision problem. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing - STOC '02*. ACM Press, 2002.

[122] I.B. Damgard, S. Fehr, L. Salvail, and C. Schaffner. Cryptography in the bounded quantum-storage model. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*. IEEE, 2005.

[123] Stephanie Wehner, Christian Schaffner, and Barbara M. Terhal. Cryptography from noisy storage. *Physical Review Letters*, 100(22), June 2008.

[124] Robert Konig, Stephanie Wehner, and Jürg Wullschleger. Unconditional security from noisy quantum storage. *IEEE Transactions on Information Theory*, 58(3):1962–1984, March 2012.

[125] Yi-Kai Liu. Building one-time memories from isolated qubits. In *Proceedings of the 5th conference on Innovations in theoretical computer science*. ACM, January 2014.

[126] Damián Pitalúa-García. Spacetime-constrained oblivious transfer. *Physical Review A*, 93(6), June 2016.

[127] Adrian Kent. Location-oblivious data transfer with flying entangled qudits. *Physical Review A*, 84(1), July 2011.

[128] Dominique Unruh. Everlasting multi-party computation. *Journal of Cryptology*, 31(4):965–1011, March 2018.

[129] Stephen Wiesner. Conjugate coding. *ACM SIGACT News*, 15(1):78–88, January 1983.

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

[131] Charles H. Bennett, Gilles Brassard, Seth Breidbart, and Stephen Wiesner. Quantum cryptography, or unforgeable subway tokens. In *Advances in Cryptology*, pages 267–275. Springer US, 1983.

[132] Charles H. Bennett, Gilles Brassard, and Seth Breidbart. Quantum cryptography II: How to re-use a one-time pad safely even if p=NP. *Natural Computing*, 13(4):453–458, October 2014.

[133] Markus Jakobi, Christoph Simon, Nicolas Gisin, Jean-Daniel Bancal, Cyril Branciard, Nino Walenta, and Hugo Zbinden. Practical private database queries based on a quantum-key-distribution protocol. *Physical Review A*, 83(2), February 2011.

[134] 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, June 2020.

[135] C. Crepeau and J. Kilian. Achieving oblivious transfer using weakened security assumptions. In *[Proceedings 1988] 29th Annual Symposium on Foundations of Computer Science*. IEEE, 1988.

[136] D. Mayers and L. Salvail. Quantum oblivious transfer is secure against all individual measurements. In *Proceedings Workshop on Physics and Computation. PhysComp '94*. IEEE Comput. Soc. Press, 1994.

[137] Dominic Mayers. Quantum key distribution and string oblivious transfer in noisy channels. In *Advances in Cryptology — CRYPTO '96*, pages 343–357. Springer Berlin Heidelberg, 1996.

[138] Claude Crépeau, Paul Dumais, Dominic Mayers, and Louis Salvail. Computational collapse of quantum state with application to oblivious transfer. In Moni Naor, editor, *Theory of Cryptography*, pages 374–393, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[139] Dominique Unruh. Universally composable quantum multi-party computation. In *Advances in Cryptology – EUROCRYPT 2010*, pages 486–505. Springer Berlin Heidelberg, 2010.

[140] Niek J. Bouman and Serge Fehr. Sampling in a quantum population, and applications. In *Advances in Cryptology – CRYPTO 2010*, pages 724–741. Springer Berlin Heidelberg, 2010.

[141] Mario Berta, Matthias Christandl, Roger Colbeck, Joseph M. Renes, and Renato Renner. The uncertainty principle in the presence of quantum memory. *Nature Physics*, 6(9):659–662, July 2010.

[142] Marco Tomamichel and Renato Renner. Uncertainty relation for smooth entropies. *Physical Review Letters*, 106(11), March 2011.

[143] Ivan B. Damgård, Serge Fehr, Renato Renner, Louis Salvail, and Christian Schaffner. A tight high-order entropic quantum uncertainty relation with applications. In Alfred Menezes, editor, *Advances in Cryptology - CRYPTO 2007*, pages 360–378, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[144] Renato Renner and Robert König. Universally composable privacy amplification against quantum adversaries. In *Theory of Cryptography*, pages 407–425. Springer Berlin Heidelberg, 2005.

[145] Renato Renner. Security of quantum key distribution, 2005.

[146] Moni Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151–158, January 1991.

[147] Johan HÅstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, January 1999.

[148] Iftach Haitner and Omer Reingold. Statistically-hiding commitment from any one-way function. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing - STOC '07*. ACM Press, 2007.

[149] Ran Canetti and Marc Fischlin. Universally composable commitments. In Joe Kilian, editor, *Advances in Cryptology — CRYPTO 2001*, pages 19–40, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[150] Dominique Unruh. Concurrent composition in the bounded quantum storage model. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, pages 467–486, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[151] Jörn Müller-Quade and Renato Renner. Composability in quantum cryptography. *New Journal of Physics*, 11(8):085006, August 2009.

[152] Michael Ben-Or and Dominic Mayers. General security definition and composability for quantum & classical protocols, 2004.

[153] Dominique Unruh. Simulatable security for quantum protocols, 2004.

[154] Ueli Maurer and Renato Renner. Abstract cryptography. In Bernard Chazelle, editor, *The Second Symposium on Innovations in Computer Science, ICS 2011*, pages 1–21. Tsinghua University Press, 1 2011.

[155] Prabha Mandayam and Stephanie Wehner. Achieving the physical limits of the bounded-storage model. *Physical Review A*, 83(2), February 2011.

[156] Stephanie Wehner and Jürg Wullschleger. Composable security in the bounded-quantum-storage model. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *Automata, Languages and Programming*, pages 604–615, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[157] Christian Schaffner. Simple protocols for oblivious transfer and secure identification in the noisy-quantum-storage model. *Physical Review A*, 82(3), September 2010.

[158] C. Cachin, C. Crepeau, and J. Marcil. Oblivious transfer with a memory-bounded receiver. In *Proceedings 39th Annual Symposium on Foundations of Computer Science (Cat. No.98CB36280)*. IEEE Comput. Soc, 1998.

[159] Yan Zong Ding, Danny Harnik, Alon Rosen, and Ronen Shaltiel. Constant-round oblivious transfer in the bounded storage model. In Moni Naor, editor, *Theory of Cryptography*, pages 446–472, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[160] George. Savvides. *Interactive hashing and reductions between Oblivious Transfer variants*. PhD thesis, McGill University, School of Computer Science, 2007.

[161] Christian Schaffner, Barbara M. Terhal, and Stephanie Wehner. Robust cryptography in the noisy-quantum-storage model. *Quantum Inf. Comput.*, 9(11&12):963–996, 2009.

[162] Hoi-Kwong Lo, Marcos Curty, and Kiyoshi Tamaki. Secure quantum key distribution. *Nature Photonics*, 8(8):595–604, July 2014.

[163] S. Pirandola, U. L. Andersen, L. Banchi, M. Berta, D. Bunandar, R. Colbeck, D. Englund, T. Gehring, C. Lupo, C. Ottaviani, J. L. Pereira, M. Razavi, J. Shamsul Shaari, M. Tomamichel, V. C. Usenko, G. Vallone, P. Villoresi, and P. Wallden. Advances in quantum cryptography. *Advances in Optics and Photonics*, 12(4):1012, December 2020.

[164] Vadim Makarov and Dag R. Hjelme. Faked states attack on quantum cryptosystems. *Journal of Modern Optics*, 52(5):691–705, March 2005.

[165] N. Gisin, S. Fasel, B. Kraus, H. Zbinden, and G. Ribordy. Trojan-horse attacks on quantum-key-distribution systems. *Physical Review A*, 73(2), February 2006.

[166] Nitin Jain, Birgit Stiller, Imran Khan, Dominique Elser, Christoph Marquardt, and Gerd Leuchs. Attacks on practical quantum key distribution systems (and how to prevent them). *Contemporary Physics*, 57(3):366–387, March 2016.

[167] Nitin Jain, Birgit Stiller, Imran Khan, Dominique Elser, Christoph Marquardt, and Gerd Leuchs. Attacks on practical quantum key distribution systems (and how to prevent them). *Contemporary Physics*, 57(3):366–387, March 2016.

[168] Dominic Mayers and Andrew Chi-Chih Yao. Self testing quantum apparatus. *Quantum Inf. Comput.*, 4(4):273–286, 2004.

[169] Artur K. Ekert. Quantum cryptography based on bell's theorem. *Physical Review Letters*, 67(6):661–663, August 1991.

[170] Jedrzej Kaniewski and Stephanie Wehner. Device-independent two-party cryptography secure against sequential attacks. *New Journal of Physics*, 18(5):055004, May 2016.

[171] Jérémy Ribeiro, Le Phuc Thinh, Jedrzej Kaniewski, Jonas Helsen, and Stephanie Wehner. Device independence for two-party cryptography and position verification with memoryless devices. *Physical Review A*, 97(6), June 2018.

[172] Antonio Acín, Nicolas Gisin, and Lluis Masanes. From bell's theorem to secure quantum key distribution. *Physical Review Letters*, 97(12), September 2006.

[173] John F. Clauser, Michael A. Horne, Abner Shimony, and Richard A. Holt. Proposed experiment to test local hidden-variable theories. *Physical Review Letters*, 23(15):880–884, October 1969.

[174] Anne Broadbent and Peter Yuen. Device-independent oblivious transfer from the bounded-quantum-storage-model and computational assumptions, 2021.

[175] Tony Metger, Yfke Dulek, Andrea Coladangelo, and Rotem Arnon-Friedman. Device-independent quantum key distribution from computational assumptions. *New Journal of Physics*, 23(12):123021, December 2021.

[176] Tony Metger and Thomas Vidick. Self-testing of a single quantum device under computational assumptions. *Quantum*, 5:544, September 2021.

[177] Chris Peikert. A decade of lattice cryptography. Cryptology ePrint Archive, Paper 2015/939, 2015. `https://eprint.iacr.org/2015/939`.

[178] Jeremy Ribeiro and Stephanie Wehner. On bit commitment and oblivious transfer in measurement-device independent settings, 2020.

[179] Hoi-Kwong Lo, Marcos Curty, and Bing Qi. Measurement-device-independent quantum key distribution. *Physical Review Letters*, 108(13), March 2012.

[180] G Murta, S B van Dam, J Ribeiro, R Hanson, and S Wehner. Towards a realization of device-independent quantum key distribution. *Quantum Science and Technology*, 4(3):035011, July 2019.

[181] Shihan Sajeed, Igor Radchenko, Sarah Kaiser, Jean-Philippe Bourgoin, Anna Pappa, Laurent Monat, Matthieu Legré, and Vadim Makarov. Attacks exploiting deviation of mean photon number in quantum key distribution and coin tossing. *Physical Review A*, 91(3), March 2015.

[182] Zishuai Zhou, Qisheng Guang, Chaohui Gao, Dong Jiang, and Lijun Chen. Measurement-device-independent two-party cryptography with error estimation. *Sensors*, 20(21):6351, November 2020.

[183] S. Pohlig and M. Hellman. An improved algorithm for computing logarithms over gf(p) and its cryptographic significance (corresp.). *IEEE Transactions on Information Theory*, 24(1):106–110, 1978.

[184] D. Shanks. Class number, a theory of factorization and genera. *In Proc. Symp. Pure Math., Providence, R.I.: American Mathematical Society*, 20:415–440, 1971.

[185] John M. Pollard. Monte Carlo methods for index computation mod p. *Mathematics of Computation*, 32:918–924, 1978.

[186] David Adrian, Karthikeyan Bhargavan, Zakir Durumeric, Pierrick Gaudry, Matthew Green, J. Alex Halderman, Nadia Heninger, Drew Springall, Emmanuel Thomé, Luke Valenta, Benjamin VanderSloot, Eric Wustrow, Santiago Zanella-Béguelin, and Paul Zimmermann. Imperfect forward secrecy: How diffie-hellman fails in practice. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 5–17, New York, NY, USA, 2015. Association for Computing Machinery.

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

[188] S. Pirandola, U. L. Andersen, L. Banchi, M. Berta, D. Bunandar, R. Colbeck, D. Englund, T. Gehring, C. Lupo, C. Ottaviani, J. L. Pereira, M. Razavi, J. Shamsul Shaari, M. Tomamichel, V. C. Usenko, G. Vallone, P. Villoresi, and P. Wallden. Advances in quantum cryptography. *Adv. Opt. Photon.*, 12(4):1012–1236, Dec 2020.

[189] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 2001.

[190] Shay Gueron. Efficient software implementations of modular exponentiation. *IACR Cryptology ePrint Archive*, 2011:239, 01 2011.

[191] Donald Beaver. Correlated pseudorandomness and the complexity of private computations. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, pages 479–488, New York, NY, USA, 1996. Association for Computing Machinery.

[192] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. 06 2003.

[193] Jesper Nielsen. Extending oblivious transfers efficiently - how to get robustness almost for free. *IACR Cryptology ePrint Archive*, 2007:215, 01 2007.

[194] Jesper Nielsen, Peter Nordholt, Claudio Orlandi, and Sai Burra. A new approach to practical active-secure two-party computation. 8, 02 2012.

[195] Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. More efficient oblivious transfer extensions with security for malicious adversaries. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015*, pages 673–701, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[196] Christian Paquin, Douglas Stebila, and Goutam Tamvada. Benchmarking post-quantum cryptography in tls. In *International Conference on Post-Quantum Cryptography*, pages 72–91. Springer, 2020.

[197] Daniel Mansy and Peter Rindal. Endemic oblivious transfer. CCS '19, page 309–326, New York, NY, USA, 2019. Association for Computing Machinery.

[198] Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, John M Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-kyber. *NIST, Tech. Rep*, 2017.

[199] Bo Mi, Darong Huang, Shaohua Wan, Libo Mi, and J. Cao. Oblivious transfer based on ntruencrypt. *IEEE Access*, 6:35283–35291, 2018.

[200] Bo Mi, Darong Huang, Shaohua Wan, Yu Hu, and Kim-Kwang Raymond Choo. A post-quantum light weight 1-out-n oblivious transfer protocol. *Computers & Electrical Engineering*, 75:90–100, 2019.

[201] Philip S. Hirschhorn, Jeffrey Hoffstein, Nick Howgrave-Graham, and William Whyte. Choosing ntruencrypt parameters in light of combined lattice reduction and mitm approaches. In Michel Abdalla, David Pointcheval, Pierre-Alain Fouque, and Damien Vergnaud, editors, *Applied Cryptography and Network Security*, pages 437–455, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[202] Danilo Verhaert, Majid Nateghizad, and Zekeriya Erkin. An efficient privacy-preserving recommender system for e-healthcare systems. In *Proceedings of the 15th International Joint Conference on e-Business and Telecommunications*. SCITEPRESS - Science and Technology Publications, 2018.

[203] Simone Scardapane, Rosa Altilio, Valentina Ciccarelli, Aurelio Uncini, and Massimo Panella. Privacy-preserving data mining for distributed medical scenarios. In *Multidisciplinary Approaches to Neural Computing*, pages 119–128. Springer International Publishing, August 2017.

[204] Christian Maulany, Majid Nateghizad, Bart Mennink, and Zekeriya Erkin. Privacy-preserving distributed access control for medical data. In *Proceedings of the 15th International Joint Conference on e-Business and Telecommunications*. SCITEPRESS - Science and Technology Publications, 2018.

[205] Hiroaki Kikuchi, Xuping Huang, Shigeta Ikuji, and Manami Inoue. Privacy-preserving hypothesis testing for reduced cancer risk on daily physical activity. *Journal of Medical Systems*, 42(5), April 2018.

[206] Ahmed M. Tawfik, Sahar F. Sabbeh, and Tarek EL-Shishtawy. Privacy-preserving secure multiparty computation on electronic medical records for star exchange topology. *Arabian Journal for Science and Engineering*, 43(12):7747–7756, March 2018.

[207] Shuang Wang, Xiaoqian Jiang, Haixu Tang, Xiaofeng Wang, Diyue Bu, Knox Carey, Stephanie OM Dyke, Dov Fox, Chao Jiang, Kristin Lauter, Bradley Malin, Heidi Sofia, Amalio Telenti, Lei Wang, Wenhao Wang, and Lucila Ohno-Machado. A community effort to protect genomic data sharing, collaboration and outsourcing. *npj Genomic Medicine*, 2(1), October 2017.

[208] Abukari Mohammed Yakubu and Yi-Ping Phoebe Chen. Ensuring privacy and security of genomic data and functionalities. *Briefings in Bioinformatics*, 21(2):511–526, February 2019.

[209] Muhammad Naveed, Erman Ayday, Ellen W. Clayton, Jacques Fellay, Carl A. Gunter, Jean-Pierre Hubaux, Bradley A. Malin, and Xiaofeng Wang. Privacy in the genomic era. *ACM Computing Surveys*, 48(1):1–44, September 2015.

[210] Philip Chan, Itzel Lucio-Martinez, Xiaofan Mo, Christoph Simon, and Wolfgang Tittel. Performing private database queries in a real-world environment using a quantum protocol. *Scientific Reports*, 4(1), June 2014.

[211] Tomohiro Ito, Hayato Koizumi, Nobumitsu Suzuki, Izumi Kakesu, Kento Iwakawa, Atsushi Uchida, Takeshi Koshiba, Jun Muramatsu, Kazuyuki Yoshimura, Masanobu Inubushi, and Peter Davis. Physical implementation of oblivious transfer using optical correlated randomness. *Scientific Reports*, 7(1), August 2017.

[212] Adam M. Lewis and Martino Travagnin. A secure quantum communications infrastructure for europe. *JRC116937*, 2019.

[213] libscapi. `https://github.com/cryptobiu/libscapi/tree/master`, 2021.

[214] Deborah A. McLennan. How to Read a Phylogenetic Tree. *Evolution: Education and Outreach*, 3(4):506–519, 2010.

[215] Ziheng Yang. Computational molecular evolution. *Oxford University Press*, 2006.

[216] J. Felsenstein. *Inferring phylogenies*. Sinauer Associates, 2003.

[217] J. Felsenstein. Phylip - phylogeny inference package (version 3.2). *Cladistics*, 5:164–166, 1989.

[218] Thomas Hughes Jukes and Charles R Cantor. Mammalian protein metabolism. *Academic Press, New York*, pages 21–132, 1969.

[219] Motoo Kimura. A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. *Journal of molecular evolution*, 16(2):111–120, 1980.

[220] J Felsenstein and G A Churchill. A Hidden Markov Model approach to variation among sites in rate of evolution. *Molecular Biology and Evolution*, 13(1):93–104, 01 1996.

[221] P J Lockhart, M A Steel, M D Hendy, and D Penny. Recovering evolutionary trees under a more realistic model of sequence evolution. *Molecular Biology and Evolution*, 11(4):605–612, 07 1994.

[222] Philippe Lemey, Marco Salemi, and Anne-Mieke Vandamme, editors. *The Phylogenetic Handbook*. Cambridge University Press, 2009.

[223] Yehuda Lindell. How to simulate it – a tutorial on the simulation proof technique. In *Tutorials on the Foundations of Cryptography*, pages 277–346. Springer International Publishing, 2017.

[224] Oded Goldreich. Secure multi-party computation, 1998.

[225] Shai Halevi and Silvio Micali. Practical and provably-secure commitment schemes from collision-free hashing. In *Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '96, page 201–215, Berlin, Heidelberg, 1996. Springer-Verlag.

[226] Yaoyun Shi. Quantum lower bounds for the collision and the element distinctness problems. In *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.*, pages 513–519, 2002.

[227] Divesh Aggarwal, Gavin K. Brennen, Troy Lee, Miklos Santha, and Marco Tomamichel. Quantum attacks on Bitcoin, and how to protect against them. *arXiv e-prints*, page arXiv:1710.10377, October 2017.

[228] Yan Huang, David Evans, Jonathan Katz, and Lior Malka. Faster secure two-party computation using garbled circuits. In *Proceedings of the 20th USENIX Conference on Security*, SEC'11, page 35, USA, 2011. USENIX Association.

[229] J Hastad and A Shamir. The cryptographic security of truncated linearly related variables. In *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, STOC '85, pages 356–362, New York, NY, USA, 1985. Association for Computing Machinery.

[230] Hugo Krawczyk. How to predict congruential generators. *Journal of Algorithms*, 13(4):527 – 545, 1992.

[231] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. Comput.*, 13(4):850–864, November 1984.

[232] L Blum, M Blum, and M Shub. A simple unpredictable pseudo random number generator. *SIAM J. Comput.*, 15(2):364–383, May 1986.

[233] Ammar Alkassar, Thomas Nicolay, and Markus Rohe. Obtaining true-random binary numbers from a weak radioactive source. In *Computational Science and Its Applications – ICCSA 2005*, pages 634–646. Springer Berlin Heidelberg, 2005.

[234] William A. Gaviria Rojas, Julian J. McMorrow, Michael L. Geier, Qianying Tang, Chris H. Kim, Tobin J. Marks, and Mark C. Hersam. Solution-processed carbon nanotube true random number generator. *Nano Letters*, 17(8):4976–4981, July 2017.

[235] Miguel Herrero-Collantes and Juan Carlos Garcia-Escartin. Quantum random number generators. *Reviews of Modern Physics*, 89(1), February 2017.

[236] Mauricio J. Ferreira, Nuno A. Silva, Armando N. Pinto, and Nelson J. Muga. Homodyne noise characterization in quantum random number generators. In *2021 Telecoms Conference (ConfTELE)*. IEEE, February 2021.

[237] Margarida Almeida, Daniel Pereira, Margarida Facão, Armando N. Pinto, and Nuno A. Silva. Impact of imperfect homodyne detection on measurements of vacuum states shot noise. *Optical and Quantum Electronics*, 52(11), November 2020.

[238] Nuno A. Silva, Margarida Almeida, Daniel Pereira, Margarida Facao, Nelson J. Muga, and Armando N. Pinto. Role of device imperfections on the practical performance of continuous-variable quantum key distribution systems. In *2019 21st International Conference on Transparent Optical Networks (ICTON)*. IEEE, July 2019.

[239] Margarida Almeida, Margarida Facao, Nelson J. Muga, Armando N. Pinto, and Nuno A. Silva. Secret key extraction in direct reconciliation CV-QKD systems. In *2021 Telecoms Conference (ConfTELE)*. IEEE, February 2021.

[240] Marco Tomamichel and Anthony Leverrier. A largely self-contained and complete security proof for quantum key distribution. *Quantum*, 1:14, July 2017.

[241] Martin Franz, Andreas Holzer, Stefan Katzenbeisser, Christian Schallhart, and Helmut Veith. CBMC-GC: An ANSI C Compiler for Secure Two-Party Computations. In Albert Cohen, editor, *Compiler Construction - 23rd International Conference, CC 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014. Proceedings*, volume 8409 of *Lecture Notes in Computer Science*, pages 244–249. Springer, 2014.

[242] Niklas Büscher, Martin Franz, Andreas Holzer, Helmut Veith, and Stefan Katzenbeisser. On compiling boolean circuits optimized for secure multi-party computation. *Formal Methods in System Design*, 51(2):308–331, September 2017.

[243] Niklas Buescher, Andreas Holzer, Alina Weber, and Stefan Katzenbeisser. Compiling low depth circuits for practical secure computation. In *Computer Security – ESORICS 2016*, pages 80–98. Springer International Publishing, 2016.

[244] Niklas Büscher, Daniel Demmler, Stefan Katzenbeisser, David Kretzmer, and Thomas Schneider. HyCC. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, January 2018.

[245] Mpc-benchmark. `https://github.com/cryptobiu/MPC-Benchmark`, 2021.

[246] Marcel Keller, Emmanuela Orsini, and Peter Scholl. Actively secure OT extension with optimal overhead. In *Lecture Notes in Computer Science*, pages 724–741. Springer Berlin Heidelberg, 2015.

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

[248] Stefan Wolf and Jürg Wullschleger. Oblivious transfer is symmetric. In *Advances in Cryptology - EUROCRYPT 2006*, pages 222–232. Springer Berlin Heidelberg, 2006.

[249] Fabian Furrer, Tobias Gehring, Christian Schaffner, Christoph Pacher, Roman Schnabel, and Stephanie Wehner. Continuous-variable protocol for oblivious transfer in the noisy-storage model. *Nature Communications*, 9(1), April 2018.

[250] 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 *2020 22nd International Conference on Transparent Optical Networks (ICTON)*. IEEE, July 2020.

[251] Siddarth Koduru Joshi, Djeylan Aktas, Sören Wengerowsky, Martin Lončarić, Sebastian Philipp Neumann, Bo Liu, Thomas Scheidl, Guillermo Currás Lorenzo, Željko Samec, Laurent Kling, Alex Qiu, Mohsen Razavi, Mario Stipčević, John G. Rarity, and Rupert Ursin. A trusted node–free eight-user metropolitan quantum communication network. *Science Advances*, 6(36), September 2020.

[252] Ranjit Kumaresan, Srinivasan Raghuraman, and Adam Sealfon. Network oblivious transfer. In *Advances in Cryptology – CRYPTO 2016*, pages 366–396. Springer Berlin Heidelberg, 2016.

[253] Henry S. Warren. *Hacker's Delight*. Addison-Wesley Professional, 2nd edition, 2012.

[254] Jack E. Volder. The cordic trigonometric computing technique. *IRE Transactions on Electronic Computers*, EC-8(3):330–334, 1959.

[255] Ebrahim M. Songhori, M. Sadegh Riazi, Siam U. Hussain, Ahmad-Reza Sadeghi, and Farinaz Koushanfar. ARM2gc. In *Proceedings of the 56th Annual Design Automation Conference 2019*. ACM, June 2019.

[256] Ivan Damgård, Serge Fehr, Carolin Lunemann, Louis Salvail, and Christian Schaffner. Improving the security of quantum protocols via commit-and-open. In *Advances in Cryptology - CRYPTO 2009*, pages 408–427. Springer Berlin Heidelberg, 2009.

[257] Yehuda Lindell and Benny Pinkas. A proof of security of yao's protocol for two-party computation. *Journal of Cryptology*, 22(2):161–188, December 2008.

[258] Zhiliang Yuan, Alan Plews, Ririka Takahashi, Kazuaki Doi, Winci Tam, Andrew Sharpe, Alexander Dixon, Evan Lavelle, James Dynes, Akira Murakami, Mamko Kujiraoka, Marco Lucamarini, Yoshimichi Tanizawa, Hideaki Sato, and Andrew J. Shields. 10-mb/s quantum key distribution. *J. Lightwave Technol.*, 36(16):3427–3433, Aug 2018.

[259] Id quantique website. `https://www.idquantique.com/random-number-generation/products/quantis-qrng-pcie/`, 2021.

[260] Gisaid database. `https://www.gisaid.org/`, 2021.

[261] Fábio Madeira, Young Mi Park, Joon Lee, Nicola Buso, Tamer Gur, Nandana Madhusoodanan, Prasad Basutkar, Adrian R N Tivey, Simon C Potter, Robert D Finn, and Rodrigo Lopez. The embl-ebi search and sequence analysis tools apis in 2019. *Nucleic acids research*, 47(W1):W636–W641, July 2019.